Resource Management in Virtualized Data Center

by

Md Rabbani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

Declaration

Md Rabbani

Abstract

As businesses are increasingly relying on the cloud to host their services, cloud providers are striving to offer guaranteed and highly-available resources. To achieve this goal, recent proposals have advocated to offer both computing and networking resources in the form of Virtual Data Centers (VDCs). However, to offer VDCs, cloud providers have to overcome several technical challenges. In this thesis, we focus on two key challenges: (1) the VDC embedding problem: how to efficiently allocate resources to VDCs such that energy costs and bandwidth consumption are minimized, and (2) the availability-aware VDC embedding and backup provisioning problem which aims at allocating resources to VDCs with hard guarantees on their availability.

The first part of this thesis is primarily concerned with the first challenge. The goal of the VDC embedding problem is to allocate resources to VDCs while minimizing the bandwidth usage in the data center and maximizing the cloud provider's revenue. Existing proposals have focused only on the placement of VMs and ignored mapping of other types of resources like switches. Hence, we propose a new VDC embedding solution that explicitly considers the embedding of virtual switches in addition to virtual machines and communication links. Simulations show that our solution results in high acceptance rate of VDC requests, less bandwidth consumption in the data center network, and increased revenue for the cloud provider.

In the second part of this thesis, we study the availability-aware VDC embedding and backup provisioning problem. The goal is to provision virtual backup nodes and links in order to achieve the desired availability for each VDC. Existing solutions addressing this challenge have overlooked the heterogeneity of the data center equipment in terms of failure rates and availability. To address this limitation, we propose a High-availability Virtual Infrastructure (Hi-VI) management framework that jointly allocates resources for VDCs and their backups while minimizing total energy costs. Hi-VI uses a novel technique to compute the availability of a VDC that considers both (1) the heterogeneity of the data center networking and computing equipment, and (2) the number of redundant virtual nodes and links provisioned as backups. Simulations demonstrate the effectiveness of our framework compared to heterogeneity-oblivious solutions in terms of revenue and the number of physical servers used to embed VDCs.

Acknowledgements

First of all, praise be to almighty Allah, the most beneficent, the most merciful, for blessing me with the intellect and ability to undertake this research.

I would like to express my deepest gratitude and appreciation to my supervisor, Professor Raouf Boutaba, for his constant guidance, valuable advice, supportive criticism, and endless patience towards the completion of this thesis. I would also like to thank Dr. Khuzaima Daudjee and Dr. Bernard Wong for their insightful comments and constructive criticisms of this work.

My profound gratitude goes to my parents Md. Jaynal Abedin and Fatema Begum for their unconditional love and utmost sacrifice for my successful endeavour. My true love is for my wonderful wife Sk. Sakila Arobi for her support and care during the completion of this thesis. Special thanks are owed to my elder brother Md. Golam Rasul, only sister Dr. Jobaida Khatun, youngest brother Md. Golam Mowla, parents-in-law Sk. Abdus Salam and Mrs. Nazma Sarker, siblings-in-law Dr. Mohd. Masudur Rahman, Sk. Bony Yamin Khaleque and Sk. Sabbir Hossain for extending a helping hand in times of need.

I would like to express my appreciation to the members of the Smart Applications on Virtual Infrastructure (SAVI) project. Specially, I thank Dr. Mohamed Faten Zhani for his support during the completion of the projects and for his valuable suggestions during the preparation of this thesis. I also thank Rafael Esteves, Professor Gwendal Simon, and Dr. Maxim Podlesny for their support during the SAVI projects. I would also like to thank all the members of the Networking Lab, and in particular Professor Reaz Ahmed, Dr. Qi Zhang, Md. Faizul Bari, Shihabur Rahman Chowdhury, and Arup Roy for their invaluable support. I am also thankful to my colleagues and friends Istiaque Ahmed, and Md. Rakibul Haque.

Finally, I sincerely thank the University of Waterloo for supporting me with generous scholarships as well as an excellent academic and research environment. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under the SAVI Research Network.

Dedication

This is dedicated to my family.

Table of Contents

List of Tables

List of Figures

Publications

Journal Publications:

- **Md. Rabbani**, M. F. Zhani, and R. Boutaba. On Achieving High Survivability in Virtualized Data Centers. *IEICE Transactions on Communications*. Vol. E97-B (1), January 2014.

- MF. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, **M. Rabbani**, Q. Zhang and M. F. Zhani. Data Center Network Virtualization: A Survey. *IEEE Communications Surveys and Tutorials*. IEEE Press, Vol. 15(2), pp. 909-928, April 2013.

Conference Publications:

- **Md. Rabbani**, R. Esteves, M. Podlesny, G. Simon, L. Granville and R. Boutaba. On Tackling Virtual Data Center Embedding Problem. *In Proceedings of the IFIP/IEEE Integrated Network Management Symposium (IM)*. Ghent (Belgium), 27-31 May 2013.

Chapter 1

Introduction

With the increasing popularity of cloud computing, data centers have been widely deployed by companies such as Amazon, Google, Facebook, and Yahoo! to support large-scale applications and to store large volumes of data [1, 2, 20, 24]. Hence, hosting services in data centers has become a multi-billion dollar business that plays a crucial role in the future IT industry.

Traditionally, data centers use dedicated servers to run applications, resulting in poor server utilization and high operational cost. The situation improved with the emergence of server virtualization technologies (*e.g.*, VMware [10], Xen [11]), which allow multiple Virtual Machines (VMs) to be collocated on a single physical server. These technologies are adopted by data center architectures proposed over the recent years [14, 28, 27]. These technologies can provide performance isolation between collocated VMs to improve application performance and security. However, server virtualization alone is insufficient to address all of the limitations of today's data center architectures. In particular, data center networks still suffer from a number of limitations:

- *No performance isolation*: Many of today's cloud applications, like search engines and web services have strict requirements on network performance in terms of latency and throughput. However, traditional networking technologies only provide best-effort delivery service with no performance isolation. Thus, it is difficult to provide predictable Quality of Service (QoS) for these applications. For example, *MapReduce* applications have strict constraints in terms of latency and throughput,that cannot be guaranteed through server virtualization alone [44].

- *Increased security risks*: Traditional data center networks do not restrict the communication pattern and bandwidth usage of each application. As a result, the network is vulnerable to insider attacks such as Denial of Service (DoS) attacks [41].

- *Poor application deployability*: Many enterprise applications use application-specific protocols and address spaces [17]. Migrating these applications to data center environments is a major hurdle because it often requires cumbersome modifications to these protocols and eventually to the applications' source codes.

- *Limited management flexibility*: In a data center environment where both servers and networks are shared among multiple applications, application owners often wish to control and manage the network fabric for a variety of purposes such as load balancing, fault diagnosis, and security protection. However, traditional data center network architectures do not provide the flexibility for different users to manage their communication fabric in a data center.

- *No support for network innovation*: Inflexibility of the traditional data center architecture prohibits network innovation. As a result, it is difficult to introduce changes to traditional data center networks such as upgrading network protocols or introducing new network services. In the long run, it will reduce the effectiveness of the initial capital investment in data center networks.

Motivated by these limitations, there is an emerging trend towards fully virtualized data center architectures [16]. As a result, recent research proposals have advocated offering both computing and networking resources in the form of Virtual Data Centers (VDCs). Basically, a VDC consists of virtual machines, routers and switches connected through virtual links with guaranteed bandwidth. This allows cloud providers to achieve better performance isolation between VDCs and to implement more fine-grained resource allocation schemes. At the same time, VDCs allow service providers to guarantee predictable network performance for their applications.

In typical cloud environments, the Infrastructure Provider (InP) owns the physical infrastructure and leases resources to multiple Service Providers (SPs). Each SP then leverages its dedicated resources to deploy applications and services and offer them to end users over the Internet. So far, InPs mainly roll out computing resources (i.e., virtual machines) with no network performance guarantees [1, 7]. This results in a variable and unpredictable network performance in addition to potential security risks [15, 52, 36, 29]. Introducing VDCs will facilitate the possibility to implement customized network protocol and management policies, to minimize the impact of security threats, and to deploy new applications and services.

1.1 Motivation and Challenges

Allocating resources to VDCs is known as the *VDC embedding* or *mapping* problem. One of the main objectives of InPs is to accommodate as many VDC requests as possible and increase their revenue. Existing embedding solutions [29, 15] have proposed heuristics to cope with the NP-hardness of the problem. However, most of these proposals are heavily topology-dependent and hence cannot be generally applied to arbitrary data center topologies. More importantly, most of them usually focus on VM placement [29] and do not explicitly consider other types of resources (*e.g.*, switches), which can limit their applicability in realistic scenarios. For example, researchers or some companies may want to test their own switching protocol that requires switches in VDC topology. Also, if InPs know in advance that their data centers are going to be affected by natural calamities, they can move their entire data center topology including servers and switches to other data centers. The requested VDC of the affected InP will include virtual switches in addition to VM. Hence, finding an efficient VDC embedding algorithm considering all of these aspects is a challenging task for InPs.

Another primary challenge that has not been adequately addressed so far is how to guarantee high VDC availability. On the one hand, for an SP, a service disruption, even for seconds, may incur high losses in revenue and also significantly impair the SP's reputation. A recent study by an IT analyst firm estimated SP losses due to service downtime from US\$$25,000$ up to US\$$150,000$ per hour [3]. On the other hand, InPs could also incur significant penalties from not delivering the promised availability specified in the service level agreements. For instance, Amazon EC2 pledges to offer a service credit equal to 10% of the bill to any customer whose resources' annual availability falls below 99.95% [1]. As a result, providing high resource availability has been pointed out as one of the primary challenges of InPs to exhort SPs to rely on the cloud.

Although, few proposals have addressed the issue of improving availability of VDCs, they assume physical components in data centers to be homogeneous in terms of failure rate and availability. However, in real environments, these characteristics are highly heterogeneous and vary significantly depending on various parameters such as the types of the equipments and their age [25, 47]. This observation suggests that, in order to achieve the desired VDC availability, the equipment heterogeneity must be considered to (1) determine the placement of VDC nodes and links and to (2) decide more accurately the number and the placement of the backup resources.

1.2 Contributions

The contributions of this thesis are threefold.

- We present a comprehensive background about data centers, the basic concepts of data center virtualization. We also discuss the business model associated in cloud environments. We also provide an extensive literature survey on VDC embedding and survivability schemes in virtualized data centers.
- We propose a new VDC embedding solution that, in addition to virtual machine placement, explicitly considers the embedding of virtual switches and links. Our proposed algorithm significantly increases the acceptance rate of VDCs and reduces the bandwidth usage in the data center network. In addition, unlike previous proposals, our solution does not restrict the type of VDC topology that can be embedded.
- Finally, we address the availability-aware resource allocation problem in virtualized data centers. Hence, we propose Hi-VI, a VDC management framework that allocates resources for VDCs while achieving the desired availability for each of them by provisioning backup resources and taking into account the availability of the underlying physical devices.

1.3 Thesis Organization

The remainder of the thesis is organized as follows.

Chapter 2 provides an overview of conventional data centers, the basic concepts of data center virtualization, and the business model commonly used in cloud environments.

Chapter 3 surveys the existing proposals addressing VDC embedding, presents the formulation of the VDC embedding problem, and proposes a heuristic to solve this problem. Simulations are then conducted to evaluate the performance of the proposed heuristic compared to a baseline algorithm.

Chapter 4 presents previous works on failure characterization in data centers, summarizes the existing availability-aware VDC allocation schemes and their limitations, and proposes a resource allocation framework that takes into account the heterogeneity of data center computing and networking equipment to dynamically provision backup resources and satisfy the desired availability for each VDC. Simulation results are then presented to show the effectiveness of our solution compared to a heterogeneity-oblivious solution.

Chapter 5 concludes the thesis with a summary of our contributions, and outlines potential directions for future research.

Chapter 2

Background

The aim of this chapter is to present the terminology relevant to data center virtualization. Table 2.1 provides a list of abbreviations used throughout the paper. We first provide some definitions related to data centers and we present an overview of conventional data centers in Section 2.1. Then we discuss data center virtualization technologies in Section 2.2. Finally, Section 2.3 describes the main stakeholders in cloud environments.

2.1 Data Center

- A *Data Center* (DC) is a facility consisting of servers (physical machines), storage and network devices (*e.g.*, switches, routers, and cables), power distribution systems, cooling systems. Table 2.1 provides a list of abbreviations used throughout the thesis.

- A *data center network* is the communication infrastructure used in a data center, and is described by the network topology, routing/switching equipment, and the used protocols (*e.g.*, Ethernet and IP). In the following, we present the main topologies used in data centers or recently proposed in the literature:

  - *Conventional data center network topology* [12] (Figure 2.1): In this topology, each Top-of-Rack (ToR) switch in the access layer provides connectivity to the servers mounted in one rack. Each *Aggregation Switch* (AS) in the aggregation layer (sometimes referred to as distribution layer) forwards traffic from multiple ToR switches to the core layer. Every ToR switch is connected to multiple aggregation switches for redundancy. The core layer provides secure connectivity

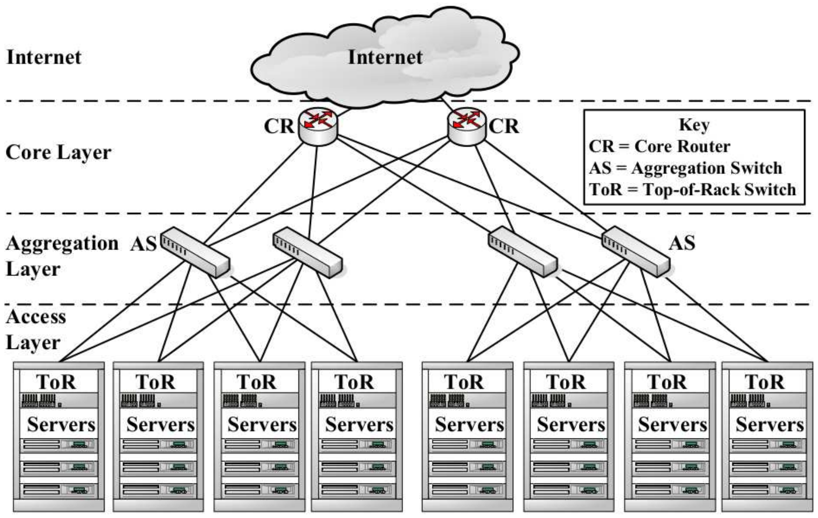


Figure 2.1: Conventional data center network topology.

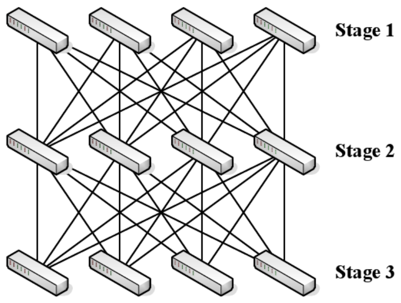


Figure 2.2: Clos topology

between aggregation switches and *core routers* (CR) connected to the Internet. A particular case of the conventional topology is the *flat layer 2 topology*, which uses only layer 2 switches.

– *Clos topology* is a topology built up from multiple stages of switches [23]. Each switch in a stage is connected to all switches in the next stage, which provides extensive path diversity. Figure 2.2 shows an example of a three-stage Clos topology.

– *Fat-tree topology* [31] is a special type of Clos topology that is organized in a tree-like structure, as shown in Figure 2.3. The topology built of k-port switches contains k *pods*; each of them has two layers (*aggregation* and *edge*) of $k/2$

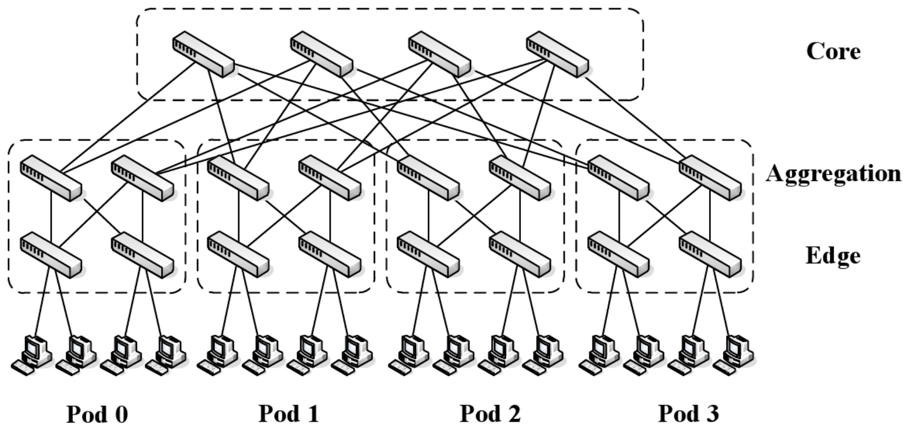


Figure 2.3: Fat-tree topology (k = 4)

switches. Each of $(k/2)^2$ core switches has one port connected to each of k pods. The i-th port of any core switch is connected to pod i so that consecutive ports in the aggregation layer of each pod switch are connected to core switches on $k/2$ strides. Each edge switch is directly connected to $k/2$ end-hosts; each of the remaining $k/2$ ports of an edge switch is connected to $k/2$ ports of an aggregation switch [14].

It is worth noting that data center network topologies are not limited to the topologies presented in this section. The interested reader can find a comparison of recently proposed data center network topologies in [34].

2.2 Data Center Virtualization

- A *Virtualized Data Center* is a data center where some or all of the hardware (*e.g.*, servers, routers, switches, and links) are virtualized. Typically, physical hardware is virtualized using software or firmware called *hypervisor* that divides the equipment into multiple isolated and independent virtual instances. For example, a physical machine (server) is virtualized via a hypervisor that creates virtual machines (VMs) having different capacities (CPU, memory, disk space) and running different operating systems and applications.

- A *Virtual Data Center* (VDC) is a collection of virtual resources (VMs, virtual switches, and virtual routers) connected via *virtual links*. While a Virtualized Data

Center is a physical data center with deployed resource virtualization techniques, a Virtual Data Center is an abstraction of resource allocation. A user can request resources in the form of VDC to the owner of virtualized data center. A *Virtual Network* (VN) is a set of virtual networking resources: virtual nodes (end-hosts, switches, routers) and virtual links; thus, a VN is a part of a VDC. *A network virtualization level* is one of the layers of the network stack (application to physical) in which the virtualization is introduced. In Figure 2.4, we show how several VDCs can be deployed over a virtualized data center.

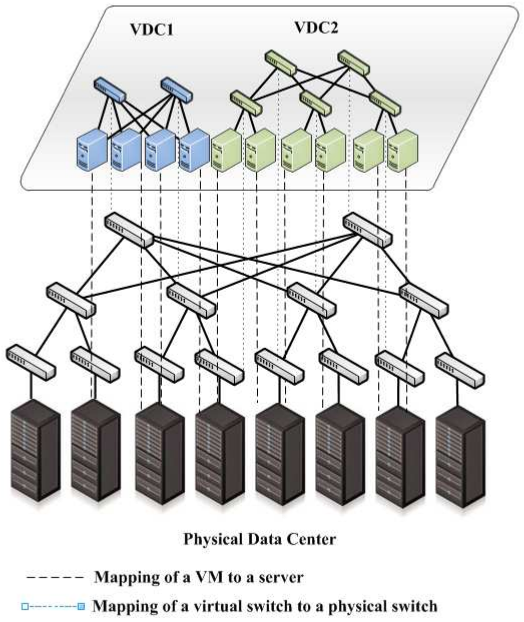

VDC1
VDC2
Physical Data Center
----- Mapping of a VM to a server
Mapping of a virtual switch to a physical switch

Figure 2.4: Virtualization of a data center.

Both network virtualization and data center virtualization rely on virtualization techniques to partition available resources and share them among different users, however they differ in various aspects. While virtualized ISP networks mostly consist of packet forwarding elements (*e.g.*, routers), virtualized data center networks involve different types of nodes including servers, routers, switches, and storage nodes. Hence, unlike a VN, a VDC is composed of different types of virtual nodes (e.g., VMs, virtual switches and virtual routers) with diverse resources (e.g., CPU, memory and disk). In addition, in the context of network virtualization, virtual links are characterized by their bandwidth. Propagation delay is an important metric when nodes are geographically distributed. However, since a data center network covers a small geographic area, the propagation delay between nodes is negligible; hence, it is always ignored when defining VDC virtual links [29, 15]. Low latency requirements are becoming increasingly important.

Another key difference between data center networks and ISP networks is the number of nodes. While the number of nodes in ISP backbones is in order of hundreds (*e.g.*, 471, 487, and 862 nodes in Sprintlink, AT&T, and Verio ISPs, respectively [42]), it can go up to thousands in today's data centers (*e.g.*, around 12000 servers in one Google Compute cluster [51]). Hence, the solutions to embedding problem in the VN domain can potentially raise scalability issues, and increase management complexity.

Furthermore, different from ISP networks, data center networks are built using topologies like the conventional tree, fat-tree, or Clos topologies with well defined properties, that allows to develop embedding algorithms optimized for such particular topologies (*e.g.*, Ballani *et al.* [15] proposed an embedding scheme applicable only to tree topology).

In summary, data center network virtualization is different from ISP network virtualization, because one has to consider different constraints and resources, specific topologies, and degrees of scalability. [21] presents a survey of ISP network virtualization techniques.

Table 2.1: List of abbreviations

Acronym	**Description**
DC	Data Center
VM	Virtual Machine
VN	Virtual Network
VDC	Virtual Data Center
$VLAN$	Virtual Local Area Network
ToR	Top-of-Rack
AS	Aggregation Switch
CR	Core Router
InP	Infrastructure Provider
SP	Service Provider
$IaaS$	Infrastructure-as-a-Service

2.3 Business Model

In this section, we describe main stakeholders in cloud environments. One of the differences between the traditional networking model and network virtualization model is participating players. In particular, whereas the former assumes that there are two players: ISPs and end-users, the latter proposes to separate the role of the traditional ISP into two: an *Infrastructure Provider* (InP) and a *Service Provider* (SP). Decoupling SPs from InPs adds opportunities for network innovation since it separates the role of deploying networking mechanisms, *i.e.*, protocols, services (*i.e.*, SP) from the role of owning and maintaining the physical infrastructure (*i.e.*, InP).

In the context of data center virtualization, an InP is a company that owns and manages the physical infrastructure of a data center. An InP leases virtualized resources to multiple service providers/*tenants*. Each tenant creates a VDC over the physical infrastructure owned by the InP for further deployment of services and applications offered to end-users. Thus, several SPs can deploy their coexisting heterogeneous network architectures required for delivering services and applications over the same physical data center infrastructure. The end-users use services and applications offered by SPs.

2.4 Summary

In this chapter, we provided the terminologies related to data center virtualization. We described the components of a data center, and different network topologies used in data centers. We also provided the definition of virtualized/virtual data centers, and the main stakeholders in a typical cloud environment. In the next chapter, we present a resource allocation scheme that allows InPs to efficiently allocate resources to VDCs with the goal of maximizing the revenue while minimizing bandwidth usage in the data center network.

Chapter 3

Virtual Data Center Embedding

3.1 Introduction

The mapping of virtual resources to physical ones is commonly known as the *VDC embedding problem*. Devising efficient VDC embedding algorithms is crucial to accept a high number of VDC requests, which in turn impacts the revenue of the InP. In this section, we deal with the problem of VDC embedding. We define a VDC in a generic way, where we distinguish multiple resource types. We typically observe that the providers of interactive multimedia applications require GPU in their VMs while other SPs are interested in very fast memory hardware (SRAM) [36]. An InP should be able to differentiate these different hardware in order to serve a vast range of SPs. We also go further by defining virtual switches and virtual links. We believe that an SP running applications with strict network requirements is interested in renting a very precise network infrastructure for the connection between its VMs. Protocols related to Software-Defined Networking (SDN) can provide some guarantees but a full isolation of applications can only be offered by the virtualization of switches. In addition, switch virtualization provides SPs with the possibility of defining their own forwarding protocols.

As the VDC embedding problem is NP-hard, we design a heuristic solution based on three embedding phases: VM mapping, link mapping, and switch mapping. We explicitly include in our heuristic different data center resources such as switches and storage. Moreover, we clearly distinguish physical servers and network nodes (i.e., switches). To increase the chances of successful VDC embedding, our solution includes two features. First, it allows multiple VMs of a single VDC request to be embedded in the same physical server, which is not considered in some previous proposals. Second, we leverage the coordination between switch mapping and link mapping phases.

The rest of this chapter is organized as follows. Existing literature on VDC embedding is described in Section 3.2. Then, in Section 3.3, we formalize the data center network model and the VDC embedding problem. In Section 3.4, we propose our algorithm for VDC embedding based on coordinated switch and link mapping. In Section 3.5, we present the performance evaluation of the proposed algorithm. Finally, we conclude in Section 3.6.

3.2 Literature Survey

The mapping of virtual resources to physical has been the subject of extensive research in the context of network virtualization [50, 22, 38, 19]. Several proposals have been made to cope with the NP-hardness of the embedding problem.

SecondNet [29] proposes Virtual Data Centers (VDCs) as the abstraction for resource allocation in multi-tenant cloud environments. The goal of SecondNet is to provide bandwidth guarantees by designing a scalable and deployable VDC embedding algorithm. SecondNet aims at increasing the utilization of the network.

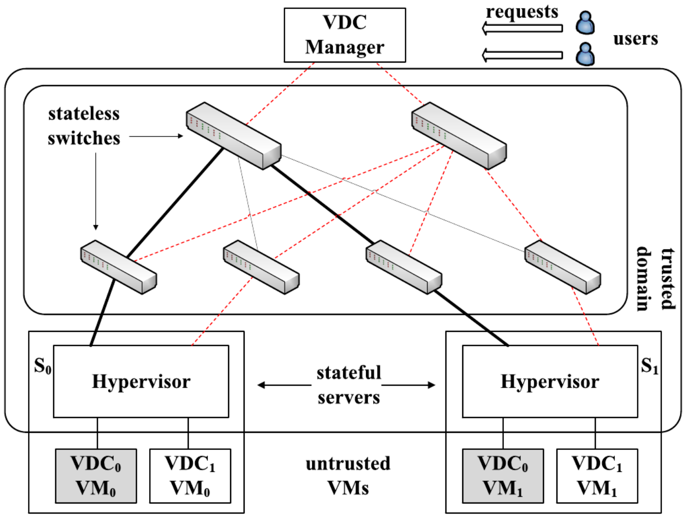
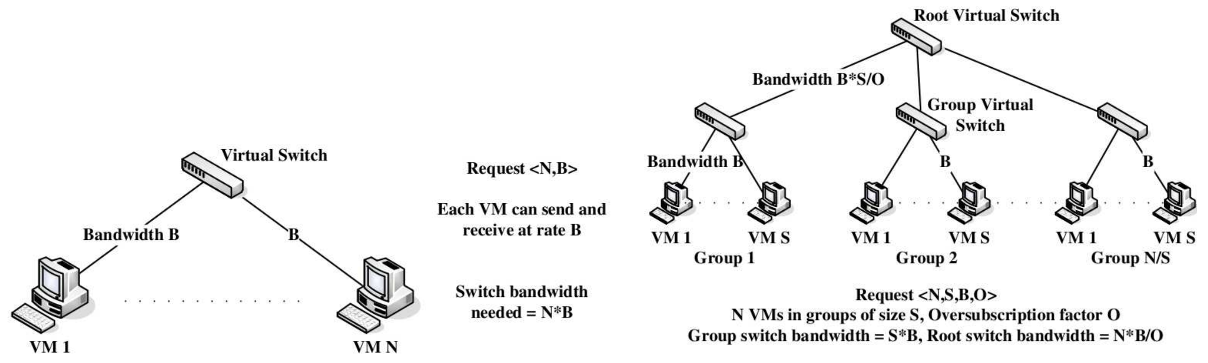


Figure 3.1: SecondNet architecture [29].

The main component of the SecondNet architecture (Figure 3.1) is the VDC manager that creates VDCs based on a requirement matrix that defines the requested bandwidth between VM pairs. SecondNet defines three basic service types: a high priority end-to-end guaranteed service (type 0), a better than best-effort service (type 1) that offers bandwidth guarantees for the first/last hops of a path, and a best-effort service (type 2).

Performance of the proposed allocation algorithm in SecondNet depends on the physical network topology. For instance, the data center network utilization is reported to be high for a BCube topology but low for fat-tree and VL2 topologies. SecondNet does not consider different types of resources (*e.g.*, storage servers, switches, routers, links) found in real data centers. We explained in Section 1.1 about the importance of considering virtual switches in addition to VMs.

Oktopus [15] is a data center network architecture that implements two virtual data center abstractions (i.e., virtual cluster and virtual oversubscribed cluster).


Virtual Switch
Bandwidth B
B
VM 1
VM N
Request <N,B>
Each VM can send and receive at rate B
Switch bandwidth needed = N*B
Root Virtual Switch
Bandwidth B*S/O
Group Virtual Switch
Bandwidth B
B
B
VM 1
VM S
Group 1
VM 1
VM S
Group 2
VM 1
VM S
Group N/S
Request <N,S,B,O>
N VMs in groups of size S, Oversubscription factor O
Group switch bandwidth = S*B, Root switch bandwidth = N*B/O

(a) Virtual cluster (b) Virtual oversubscribed cluster

Figure 3.2: VDC request abstractions in Oktopus [15].

A virtual cluster (Figure 3.2(a)) provides the illusion of having all VMs connected to a single non-oversubscribed virtual switch. However, a virtual oversubscribed cluster (Figure 3.2(b)) emulates an oversubscribed two-tier cluster consisting of a virtual root switch that interconnects a set of virtual clusters. A tenant can choose the abstraction and the degree of the oversubscription of the virtual network based on the communication pattern of the applications to be deployed in the VDC (*e.g.*, user-facing web-applications, data intensive applications). Oktopus uses a greedy algorithm for the resource allocation to the VDCs that aims at finding the best trade-off between the performance guarantees offered to tenants, their costs, and the provider revenue. They try to map the VDC requests to the smallest sub-tree of the data center topology with the least amount of residual bandwidth on the links connecting the sub-tree to the rest of the topology.

Oktopus has some limitations. It supports only two types of VDC requests. Besides, it can be applied only in tree-like physical topologies.

The Traffic-Aware Virtual Machine Placement Problem (TVMPP) [32] focuses on improving the scalability of data center networks by optimizing the VM placement. The main idea is to minimize the overall communication cost of the data center by placing VMs that have heavy traffic among them close to each other, thus avoiding network bottlenecks in the data center. A two-tier algorithm , called Cluster and Cut, is proposed to solve the problem.

One limitation of this proposal consists in the fact that traffic and cost matrices must be known in advance. In a realistic scenario, VMs can belong to different tenants, which in turn run applications with different traffic patterns that may not be known prior to the application deployment. Besides, other types of virtual resources (networking, storage) that exist in a data center are not considered in this work.

NetShare [30] tackles the problem of bandwidth allocation in virtualized data center networks proposing a statistical multiplexing mechanism that does not require any changes in switches or routers. NetShare allocates bandwidth for tenants in a proportional way and achieves high link utilization for infrastructure providers. In NetShare, data center network links are shared among services, applications, or corporate groups, rather than among individual connections. In this way, one service/application/group cannot hog the available bandwidth by opening more connections.

Scalability of NetShare can be an issue because queues have to be configured at each switch port for each service/application. In addition, NetShare aims to achieve fairness in bandwidth allocation, and, thus, does not provide any absolute bandwidth guarantees to services.

Seawall [41] is a bandwidth allocation scheme that allows InPs to define how the bandwidth will be shared in a data center network with multiple tenants. The idea of Seawall is to assign weights to network entities generating traffic (*e.g.*, VMs, process), and to allocate bandwidth according to these weights in a proportional way. Seawall overcomes the scalability problem associated with NetShare, by mapping many VMs onto a small, fixed number of queues.

Both NetShare and Seawall aim to achieve fairness in bandwidth allocation, and, thus, does not provide any absolute bandwidth guarantees to services.

Gatekeeper [39] focuses on providing guaranteed bandwidth among VMs in a multi-tenant data center, and achieving a high bandwidth utilization. Bandwidth allocation is done by defining both the minimum guaranteed rate and maximum allowed rate for each

VM pair. These parameters can be configured to achieve minimum bandwidth guarantee, while ensuring that link capacities are effectively utilized by tenants.

Gatekeeper only works for full bisection-bandwidth networks and focuses on preventing congestion on access links. However, it assumes no congestion at the core links which might be optimistic.

FairCloud [33] tackles the tradeoffs among minimum bandwidth guarantees, network utilization and fairness for sharing networks in multitenant data centers. To address this issue, it proposes three bandwidth allocation policies. However, these weight-based competing mechanisms can not provide bandwidth guarantees for VMs.

VINEYard [22] is a set of online embedding algorithms for virtual networks that coordinates the node and link mapping phases. The motivation behind this coordinated mapping relies on the fact that ignoring the relation between node and link mapping can lead to poor performance. In VINEYard, the virtual network embedding problem is formulated as a mixed integer program that includes both node and link-related constraints. In order to be solved in polynomial time, the MIP formulation is transformed into a linear program through constraint relaxation. The main objective of VINEYard is to minimize the cost of embedding a VN request to a physical substrate and, consequently, increase the InP revenue.

The limitation of VINEYard in the context of data center is that is does not consider explicitly other types of virtual resources that can exist in a data center and that have specific requirements, like ingress and egress bandwidth and storage capacity. Besides, it assumes that the substrate supports path splitting, which is not true for all data center environments.

These are the most prominent proposals of VDC embedding problem and bandwidth sharing mechanism in multitenant data centers. We find that most of these proposals are only applicable to tree-like proposals. Most of them also do not explicitly consider other types of resources (*e.g.*, storage servers, switches, routers, links) and their associated constraints, which can limit their applicability in real environments. Other bandwidth sharing proposals cannot provide guaranteed bandwidth for VDCs. In the following, we propose our VDC embedding algorithm that overcomes the aforementioned limitations.

3.3 Problem Formulation

We provide now our network model and we mathematically formulate the VDC embedding problem. Used notations are provided in Table 3.1 and Table 3.2.

Table 3.1: Notations for physical network infrastructure.

$G^p(S^p, X^p, E^p)$	The physical network infrastructure
S^p	set of physical servers
X^p	set of physical switches
E^p	set of physical links
$c_i(s^p)$	capacity of hardware i of server $s^p \in S^p$
$\bar{c}_i(s^p)$	residual capacity of hardware i of server $s^p \in S^p$
$b(e^p)$	bandwidth capacity of link $e^p \in E^p$
$\bar{b}(e^p)$	residual bandwidth capacity of link $e^p \in E^p$
$c_i(x^p), \bar{c}_i(x^p)$	capacity of hardware i (and residual capacity) of switch x^p

3.3.1 Physical Data Center

The InP owns a physical data center network, which is modeled as a weighted undirected graph, $G^p(S^p, X^p, E^p)$, where S^p is the set of physical servers, X^p is the set of physical switches and E^p is the set of physical links.

We associate with each physical server in S^p a set of featured hardware, which typically includes CPU, data storage memory, Graphical Processing Unit (GPU), and fast SRAM memory. A part of each of these hardware can be reserved by a tenant. To preserve the generality of our model, we associate with each physical server $s^p \in S^p$ a set of featured capacities $c_i(s^p), i \in \{1, \dots, k\}$ where k is the number of different reservable specific hardware capacities. For example, $c_1(s^p)$ and $c_2(s^p)$ can refer to CPU and memory capacities respectively. We also keep track of the residual capacity for each hardware after some parts of these hardware have been rented by a tenant. We note $\bar{c}_i(s^p)$ the residual capacity for hardware $i \in \{1, \dots, k\}$.

We also associate with each physical link e^p in E^p a bandwidth capacity, noted $b(e^p)$, and the residual bandwidth capacity, noted $\bar{b}(e^p)$. Please note that this capacity can be extended with a subscript in the same manner as for the capacity of physical servers.

One of the novelties of our model is that we also introduce resources for switches. We consider each buffer of each switch port and we allow a SP to partially rent each of these buffers. Thus, a SP is able to very precisely adjust packet losses and processing in every port. To simplify, we use the same notations as for the virtual servers, so $c_i(x^p)$ is the capacity of the hardware i in the virtual switch x^p.

3.3.2 VDC Request

The clients of the InP are entities that would like to reserve a subset of InP's physical infrastructure. We call a VDC request what a *tenant* wants to reserve. The VDC request is also modelled as a weighted undirected graph, which we note $G^v(S^v, X^v, E^v)$, where S^v is the set of VMs, X^v is the set of virtual switches and E^v is the set of virtual links.

Table 3.2: Notations for VDC request.

$G^v(S^v, X^v, E^v)$	VDC request
S^v	set of virtual machines
X^v	set of virtual switches
E^v	set of virtual links
$c_i(s^v)$	capacity of hardware i in VM $s^v \in S^v$
$b(e^v)$	bandwidth of virtual link $e^v(i,j) \in E^v$
$c_i(x^v)$	capacity of hardware i in virtual switch $x^v \in X^v$

We keep the same notations as for the physical infrastructure for the specific required capacities of VMs and virtual links. Thus, $c_i(e^p)$ and $b(e^p)$ refer respectively to the requested capacity of hardware i on VM s^v and the requested bandwidth capacity of virtual link e^v respectively.

3.3.3 Objective

The goal of an InP is to maximize its profits through an optimal exploitation of its infrastructure. This objective is related to maximizing the number of VDC requests that are successfully embedded in the physical infrastructure subject to embedding constraints.

Another challenge for InP is that tenants do not coordinate to request VDC. Thus, VDC requests do not arrive at the same time. Hence the problem faced by InPs is an *online* problem where the inputs (here the VDC requests) have to be processed iteratively, without knowledge of the future VDCs. Let consider a VDC request G^v, which is received by the InP while a part of its infrastructure has already been reserved. We express in the following the constraints for the embedding of G^v. We first have to define the output variables.

Let x_{uv} be a set of binary variables for the mapping of equipment $u \in S^v$ (respectively $u \in X^v$) into a physical server $v \in S^p$ (respectively $v \in X^p$).

$$x_{uv} = \begin{cases} 1 & \text{if } u \text{ is embedded into } v \\ 0 & \text{otherwise} \end{cases}$$

Let $y_{(uu')(vv')}$ be a set of binary variables for the mapping of virtual link $(uu') \in E^v$ into physical link $(vv') \in E^p$.

$$y_{(uu')(vv')} = \begin{cases} 1 & \text{if } (uu') \text{ is embedded into } (vv') \\ 0 & \text{otherwise} \end{cases}$$

An embedding can be done if the following constraints are fulfilled for the mapping of equipments.

$$\sum_{v \in S^p \cup X^p} x_{uv} = 1, \quad \forall u \in S^v \cup X^v \tag{3.1}$$

$$\sum_{u \in S^v \cup X^v} x_{uv} \times c_i(u) \leq \bar{c}_i(v), \quad \forall v \in S^p \cup X^p, \forall i \tag{3.2}$$

Constraint (3.1) ensures that a VM is embedded into one and only one physical server. It also makes sure that all VMs are embedded. Constraint (3.2) guarantees that the capacities of physical servers are not overused by the embedded VMs.

For the link mapping, here are the constraints:

$$\sum_{(vv') \in E^p} y_{(uu')(vv')} \geq 1, \quad \forall (uu') \in E^v \tag{3.3}$$

$$\sum_{(uu') \in E^v} y_{(uu')(vv')} \times b((uu')) \leq \bar{b}((vv')), \qquad \forall (vv') \in E^p \tag{3.4}$$

$$x_{uv} = 1, x_{u'v'} = 1, \forall u, u' \in S^v \cup X^v \tag{3.5}$$

Constraint (3.3) ensures that a virtual link can be mapped to several physical links or a physical path. All the links $vv' \in E^p$, that any $uu' \in E^v$ is mapped to, form a path between physical servers where u and u' are mapped. Constraint (3.4) guarantees that physical links have enough bandwidth to host embedded virtual links. Constraint (3.5) indicates that selected links belong to the physical devices x_v and $x_{v'}$ hosting the virtual devices x_u and $x_{u'}$, respectively.

For one given VDC and one infrastructure configuration, several embeddings can fulfill all above constraints. But some of them succeed in preserving “nearby” resources for future VDCs although others barely allow the next VDCs to be embedded. The quality of the online embedding algorithm for VDCs directly relates to the capacity of the embedding algorithm to be friendly for the next VDCs.

3.4 Proposed Solution

Optimization problems for embeddings that include both nodes and links are known to be NP-hard. For practical implementations, heuristics are needed. We discuss in the next sub-section the main principles of an iterative three-steps algorithm for the embedding. We go into the details of our main proposal in sub-section 3.4.3.

3.4.1 Description of the three phase embedding

For the processing of a new VDC G^v, an iterative heuristic consists of the three following rounds.

Phase I: VMs mapping – in this first step, the objective is to map each VM in S^v into one physical server. The mapping should respect the constraints in Equation (3.2), that is, the physical server that host a VM should have enough capacity. Various strategies can be implemented. With regard to the following steps, the proximity of the selected physical servers is critical.

Phase II: Virtual switches mapping – the second step is about mapping each virtual switch in X^v to a physical switch. Besides constraints (3.2), the main idea here is to associate virtual switches to physical ones while taking into account the previous VM mapping. It makes more sense to map a virtual switch between two already embedded VM so that the distance between the chosen physical switch and the physical servers that host the VM is small.

Phase III: Virtual links mapping – the final step is to map the virtual links into physical links, with respect to constraints (3.4). Many paths can exist between two physical servers in the real topology. The goal here is to find the paths between the physical servers that host the virtual equipments that are given in E^v so that all virtual links can be embedded.

In this heuristic, each step critically impacts the following steps because the input of each phase depends on the output of the previous phase. Therefore, we have to implement a mechanism that allows to get back to a previous phase if ever one phase cannot be fulfilled. For example, the impossibility to find a mapping for virtual switches does not mean that the whole VDC cannot be embedded. Another VM mapping may enable a successful switch mapping. On the other hands, a heuristic does not explore all possible solutions. A trade-off should be found between the number of tested solutions allowed by such a *retro-mechanism* and the efficiency of the whole algorithm, which should be fast.

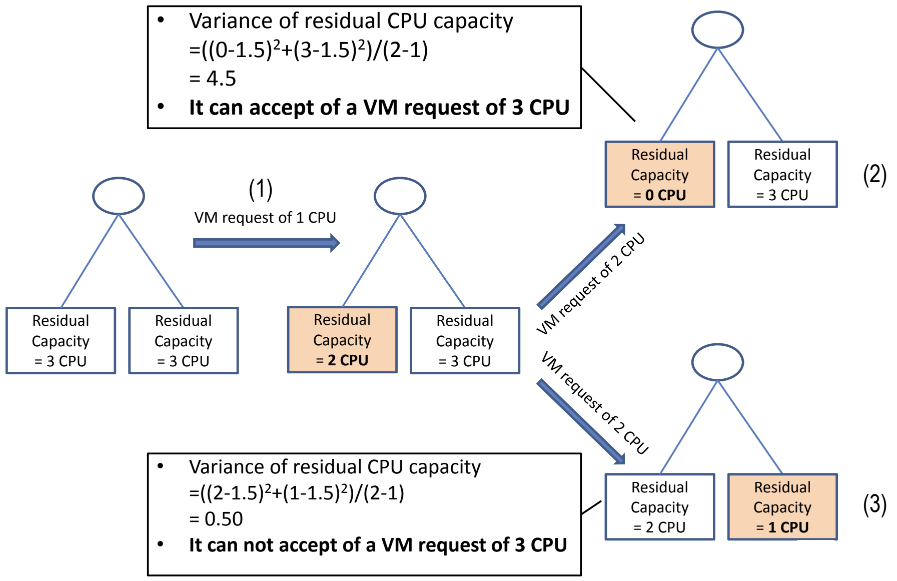
We present now our heuristic. First, we present the rationale of our solutions. Then, we describe our implementations of the three steps.

3.4.2 Rationale for the Objectives

The main idea of our heuristic is to reduce server fragmentation, minimize communication costs between VMs and virtual switches, increase the residual bandwidth of the network, and offer load balancing capabilities to the network. For the sake of simplicity, we consider only one resource for the server (*e.g.*, CPU).

Reduce server fragmentation. We use the variance of the residual CPU capacity of the physical servers. The rationale behind using variance in server defragmentation is explained in Figure 3.3 with basic physical topology. Initially, two physical servers have residual capacity of 3 CPU units each. After receiving a VM request (1) of 1 CPU unit, the residual CPU of the servers become 2 CPU units and 3 CPU units. A future VM request of 2 CPU units has two possible mappings. In the first one (2), the residual CPU of one server is 3 CPU units, which allows a future VM request requiring 3 CPU units. In the second mapping (3), the residual CPU of the servers are 2 CPU units and 1 CPU unit. In this latter case, a future VM request with 3 CPU cannot be accepted. The variance of the residual CPU capacity in the first allocation is 4.5 and 0.5 in the second allocation. We observe that the higher the variance is, the higher the resource defragmentation is and higher the possibility of accepting higher number of VDC requests.

Let $\overline{(\bar{c}_i)}$ be the average residual capacity of hardware i (here we have only CPU, represented by c_1) over all nodes s^p. Formally, the solution that we select among all possible solutions satisfying the previously defined constraints should take into account $\mathcal{V}_{cpu}$, which



Variance of residual CPU capacity
=((0-1.5)²+(3-1.5)²)/(2-1)
= 4.5
It can accept of a VM request of 3 CPU
Residual Capacity = 0 CPU
Residual Capacity = 3 CPU
(2)
(1)
VM request of 1 CPU
Residual Capacity = 3 CPU
Residual Capacity = 3 CPU
Residual Capacity = 2 CPU
Residual Capacity = 3 CPU
VM request of 2 CPU
VM request of 2 CPU
Variance of residual CPU capacity
=((2-1.5)²+(1-1.5)²)/(2-1)
= 0.50
It can not accept of a VM request of 3 CPU
Residual Capacity = 2 CPU
Residual Capacity = 1 CPU
(3)

Figure 3.3: Using variance for resource defragmentation.

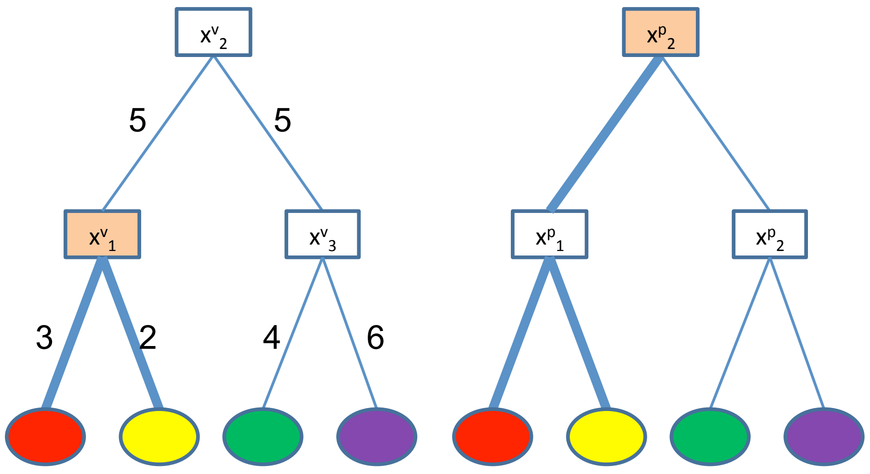
is defined as:

$$\mathcal{V}_{cpu} = \frac{|S^p|}{\sum_{u \in S^p} \left(\bar{c}_1(u) - \bar{\bar{c}}_1\right)^2} \tag{3.6}$$

Minimize communication cost and increase residual bandwidth. We consider the hop-count between the virtual nodes (*i.e.*, VM or virtual switch) multiplied by the corresponding requested bandwidth of the virtual link connecting the two virtual nodes. For example, in Figure 3.4 the cost of mapping virtual switch x_1^v (Fig. 3.4(a)) to physical switch x_2^p (Fig. 3.4(b)) is 2 (hop_count from physical switch x_2^p to the *red* physical server) $\times 3$ (bandwidth required between the virtual switch x_1^v and the *red* virtual server) $+2$ (hop_count from physical switch x_2^p to the *yellow* physical server) $\times 2$ (bandwidth required between the virtual switch x_1^v and the *yellow* virtual server) $= 10$, if we consider only the *red* and *yellow* virtual machines, which are the closest ones to the virtual switch x_1^v.

A solution that matches our needs takes into account $\mathcal{V}_{sw}$, which is defined as:

$$\mathcal{V}_{sw} = \sum_{(vv') \in E^p} \sum_{(uu') \in E^v} y_{(uu')(vv')} \times b((uu')) \times hop\_count(vv') \tag{3.7}$$



(a) Virtual request (b) Physical network

Figure 3.4: Minimizing communication cost

Load balancing link. Mapping virtual switches to some location can possibly require a high number of virtual links than mapping them to other locations, and result in network bottlenecks. In order to avoid bottleneck of physical links, we also use load balancing for virtual links having the same communication cost by minimizing the variance of the residual network bandwidth.

In the same manner as for $\mathcal{V}_{cpu}$, we define $\mathcal{V}link$ as:

$$\mathcal{V}_{link} = \frac{\sum_{(vv')\in E^p}\left(\bar{b}(vv') - \overline{(\bar{b})}\right)^2}{|E^p|} \tag{3.8}$$

Overall Optimization. In order to consider all aforementioned objectives, we define one unique objective, which is to minimize:

$$\alpha \times \mathcal{V}_{sw} + \beta \times \mathcal{V}_{cpu} + \gamma \times \mathcal{V}_{link} \tag{3.9}$$

Since such optimization problem of VDC embedding is NP hard [29], we present a heuristic (shown in Algorithm 1) to solve the problem that has three phases: VM mapping, virtual switch mapping, and virtual link mapping. In order to achieve locality of the VMs, we try to start the mapping process from one physical server. If any of the three phases fails, we increase the number of physical servers adding one adjacent server and try the mapping process again, until we consider all the physical servers in the data center.

Algorithm 1 VDC Embedding Algorithm

```
1: for ($|S^p| = 1; |S^p| \leq TotalNumberofServers; S^p = S^p \cup \{AdjacentServer\}$) do
2:   VM Mapping:
3:   Sort the VMs, $s^v \in S^v$ according the requested
4:   resource capacity
5:   for each $s^v \in S^v$ do
6:     for each $s^p \in S^p$ do
7:       Add an edge from $s^v$ to $s^p$ if $s^p$ satisfies the
8:       capacities of $s^v$
9:     end for
10:    Add a source node and add an edge from source
11:    node to $s^v$.
12:    Add a destination node and add an edge from
13:    each of $s^p$ to destination node.
14:    Solve min-cost flow problem and goto line 1 if fails
15:    update residual capacities of $s^p$.
16:  end for
17:  Switch Mapping:
18:  for each $x^v \in X^v$ do
19:    for each $x^p \in X^p$ do
20:      Add an edge from $x^v$ to $x^p$ if $x^p$ satisfies the
21:      capacities of $x^v$
22:    end for
23:  end for
24:  Add a source node and add an edge from source
25:  node to each of $s^v$.
26:  Add a destination node and add an edge from each
27:  of $s^p$ to destination node.
28:  Solve min-cost flow problem and go to line 1 if fails
29:  update residual capacities of $x^p$.
30:  Link Mapping:
31:  for each $e^v \in E^v$ do
32:    Remove each $e^p \in E^p$ where $b(e^p) < b(e^v)$
33:    Run BFS to compute the shortest path and go
34:    to line 1, if no path found
35:    Update capacity of each link $e^p$ in that path
36:  end for
37:  If all mapping phases are successful, break
38: end for
```

3.4.3 VDC Mapping

In the following, we describe the three phases of the embedding involved in our proposed solution.

VM Mapping In the VM mapping, we map VMs to the physical servers; we do it sequentially so that more than one VM can be mapped to one physical server. In the beginning, requested VMs are sorted according to a requested resource capacity in a descending order. Thus, we can reject a request quickly if there is not enough capacity to satisfy the request. Then we map VMs sequentially so that more than one VM can be mapped to a physical server. First, we take the first VM in the sorted list and find all the physical servers that can satisfy the resource requirements of the VM. Then, we map the VM to the physical server that minimizes the fragmentation cost. The cost function $a(s^p)$ for each server s^p is defined by,

$$a(s^p) \quad = \quad b(s^p) \times \frac{1}{|\bar{c}_1(s^p) - \bar{\bar{c}}_1|} \tag{3.10}$$

where $b(s^p)$ is the used bandwidth of the server s^p and $\overline{(\bar{c}_1)}$ is the mean of the residual CPU of the physical servers. We rely on the used bandwidth of physical servers for load balancing, and on the difference between the residual resource capacity and the average resource capacity of physical servers for server defragmentation, that will satisfy our objectives mentioned in Section 3.4.2.

Switch Mapping After mapping VMs, we proceed to switch mapping. Similar to VM mapping, we build a bipartite graph keeping virtual switches in the left and the physical switches in the right. Then, we add an edge from a virtual switch i to a physical switch j if the residual capacity of the physical switch j can satisfy the requested capacity of the virtual switch i. We add a source node s at the left of the virtual switches and a destination node d at the right of the physical switches. Following that, we add an edge from the source node s to each of the virtual switches and an edge from each of the physical switches to the destination node d, as shown in Figure 3.5. This reduced the switch mapping problem to finding min-cost flow from source s to destination d.

$$\text{Minimize} \sum_{(i,j)\in E} a(i,j).x(i,j) \tag{3.11}$$

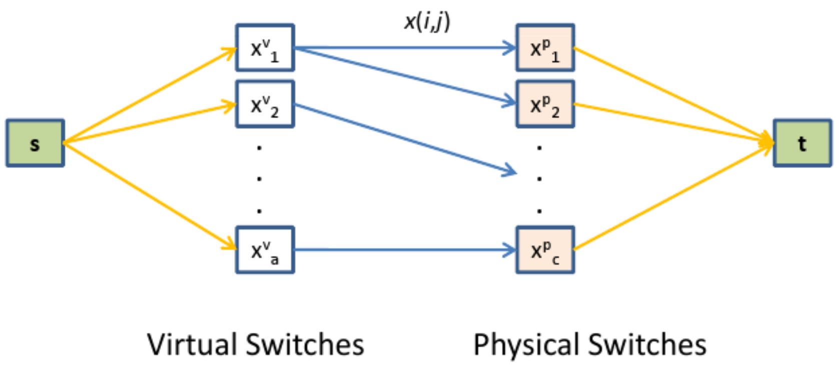


Figure 3.5: Min-cost flow graph used for switch mapping.

$$\begin{aligned}
\text{where, } x(i,j) &\in \{0,1\} \\
\sum_{j\in V} x(i,j) &= 0, \text{for all } i \neq s,t \\
\sum_{j\in V} x(s,j) &= \text{number of virtual switches} \\
\sum_{i\in P} x(i,t) &= \text{number of physical switches} \\
E &= \text{Set of edges in the graph} \\
V &= \text{Set of virtual switches in the graph} \\
P &= \text{Set of physical switches in the graph} \\
s &= \text{source node of the graph} \\
t &= \text{destination node of the graph}
\end{aligned} \tag{3.12}$$

The cost function $a(i,j)$ for each edge between a virtual switch i and a physical switch j is defined by $a(i,j) = \sum [hop\_count(j,m) \times bandwidth(i,n)]$, where $m \in S^p$, $n \in S^v$, and n is mapped into m. The intuition behind the cost function is to map the virtual switch to the physical switch that offers the lowest communication cost ($hop\_count \times bandwidth$) for VMs of the same VDC connecting to the virtual switch i. The cost function described above is used as the weight for each edge in the graph of min-cost flow problem. The variable $x(i,j)$ gives the mapping of virtual switches to physical switches. If the value of $x(i, j)$ is 1, it indicates that the virtual switch i is mapped to the physical switch j. By considering the bandwidth of the links when mapping switches, the algorithm increases the residual bandwidth of the network, which ultimately results in higher acceptance ratio of VDC requests.

Link Mapping Finally, we start link mapping after finishing the node mapping and switch mapping. We map each of the virtual links to physical links one by one. Before allocating any link, we first remove all the physical links having residual bandwidth capacity less than the requested bandwidth capacity of the virtual link. Then, we calculate the shortest path between the physical nodes, where the source and destination nodes of the virtual link are mapped, to reduce the communication cost. We use Breadth First Search (BFS) algorithm to find the unweighted shortest path.

3.5 Performance Evaluation

We studied the performance of our VDC allocation algorithm through simulation. In this section, we describe the simulation environments followed by performance metrics and result analysis. We have shown that our VDC algorithm can embed VDC requests that include virtual switches.

As no other existing work consider switch mapping in VDC embedding algorithm, we have shown the advantages of our algorithm for VDC allocation by comparing acceptance ratio, network and CPU utilization with a baseline algorithm of randomized heuristic. We illustrate the presentation of the iterative heuristic by an example: a simple algorithm based on randomized processes. For each VM, a physical server is randomly picked and if it has the capacity to host the VM, the algorithm processes the next VM, otherwise another physical server is picked. This algorithm iterates on the whole set of VMs until all VMs are successfully mapped. The switch mapping is done in the same manner. Finally, the link mapping is done by computing the shortest path for every virtual links.

3.5.1 Simulation Environment

We implemented a discrete event simulator for the evaluation of our proposed algorithm using C++ and used the open source C++ template library LEMON (Library for Efficient Modelling and Optimization in Networks) [8] to solve minimum-cost flow problem.

We used VL2 [27] topology for our physical data center network topology. Our physical data center has 24 servers, 22 switches and 144 links. The links between servers and ToR (top-of-rack) switches have bandwidth 1000 unit whereas the links between ToR switches and aggregate switches have 10 000 units and the links between aggregate switches and intermediate switches have the capacities of 10 000 units. The normalized values of CPU and storage were real numbers, uniformly distributed between 0.5 and 1.

For the VDC requests, we used the VL2 topology, star topology and mixed of them. For all the topologies, the CPU and storage of the VMs were real numbers uniformly distributed between 0 and 0.08. For the VL2 topology, the bandwidth of the links between servers and ToR switches, ToR switches and aggregate switches, and aggregate switches and intermediate switches, were uniformly distributed from 1 to 100, 1 to 1000, and 1 to 1000 respectively. For the start topology, the number of VMs are uniformly distributed from 3 to 10 and the links between the switch and the servers have bandwidth, uniformly distributed from 1 to 100. The VDC requests arrived in a Poisson process. The durations of the VDC requests were following a Poisson distribution having average of 500 time units. Our embedding algorithm serves the VDC requests online and with the passage of time, VDC requests, whose duration have passed, leave the physical data center and new VDC requests arrive. We also consider the case where only switches are mapped randomly while VMs are mapped according to our heuristic.

3.5.2 Performance Metrics

Acceptance Ratio

The acceptance ratio of a certain point of time is the ratio of the number of accepted VDC requests and the total number of VDC requests until that time. This gives a sense of how well an algorithm is performing, but can not completely capture the whole picture of the performance. An algorithm may accept more number of smaller VDC requests which can end up less revenue.

CPU Utilization

The CPU utilization is the total CPU used for embedding VDC requests divided by the total CPU capacity of the physical data center at a certain point of time, which is expressed as percentage.

Network Utilization

The network utilization is the total bandwidth of the links used for embedding VDC request divided by the total bandwidth capacity of the physical data center at a certain point of time, which is also expressed as percentage.

Revenue

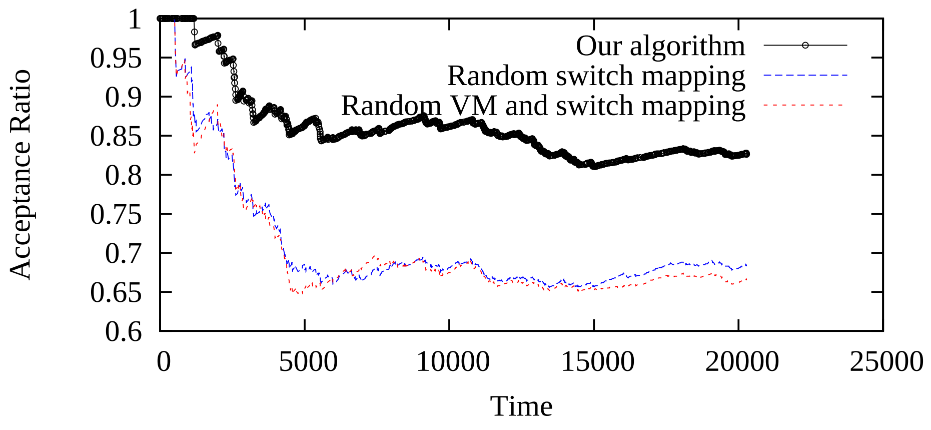
We also computed the revenue generated for an embedding algorithm over time. Revenues are expressed in terms of allocated resources. We have used a revenue model [50] where revenue is considered as sum of bandwidth and CPU. α is used to strike a balance between the two resources.

$$Revenue(G^v) = \sum_{e^v \in E^p} b(e^v) + \alpha \sum_{s^p \in S^p} c_1(s^p)$$

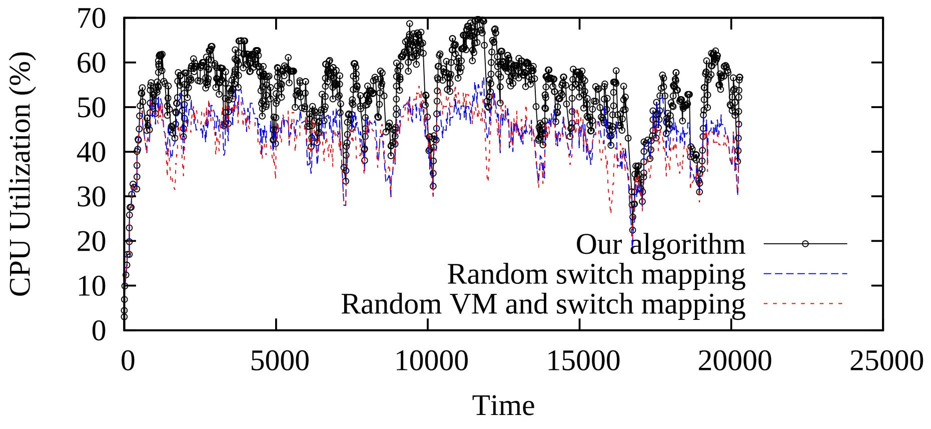
3.5.3 Simulation Results

After running the simulation, we observed that our algorithm was able to embed the VMs as well as the virtual switches to the physical resources. It also embeds both types of topologies which shows the ability of our algorithm to embed VDC of any topology. To show the effectiveness of our algorithm, we compared our algorithm with random switch mapping and random VM mapping. Figure 3.6 and 3.7 show the comparison of performance metrics of our algorithm with random switch mapping and random VM and switch mapping when VL2 topologies were used for input VDC requests. Our algorithm outperforms the random algorithms in terms of each of the metrics. For example, at 20 000 time unit, our algorithm has 0.83 acceptance ratio whereas the random switch mapping and random VM and switch mapping algorithms have acceptance ratio of 0.68 and 0.66 respectively. Our algorithm also has higher CPU and network utilization and revenue. As our algorithm considers the used CPU and bandwidth of the servers while doing VM mapping and bandwidth of the links while doing switch mapping, it can accept higher number of VDC requests as well as higher revenue.

Figure 3.8 and 3.9 show the comparison of performance metrics of the algorithms when both VL2 topologies and star topologies were used for input VDC requests. Here, again we see that our algorithm outperforms the random algorithms in terms of each of the metrics. But here the random algorithms comparatively perform better than for the VL2 only topologies. Because, VDC requests of star topology were added to the input. Star topology has only one virtual switch and all the VMs are connected to the switch. So,



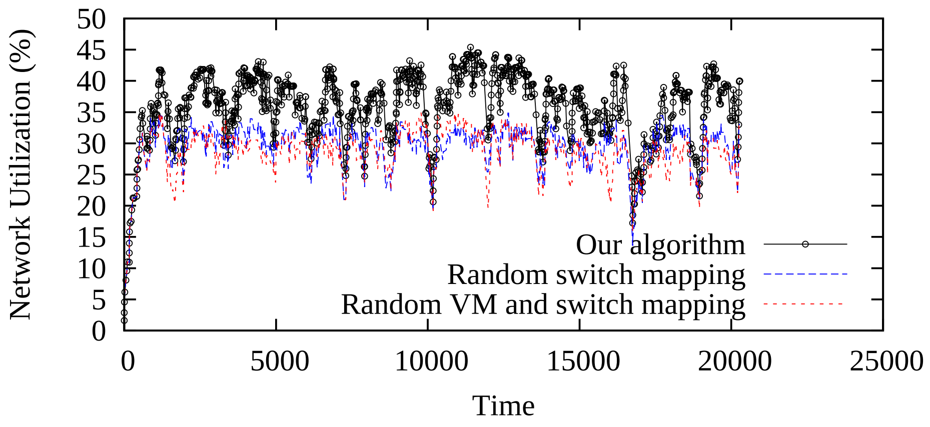
(a) Comparison of acceptance ratio over time.



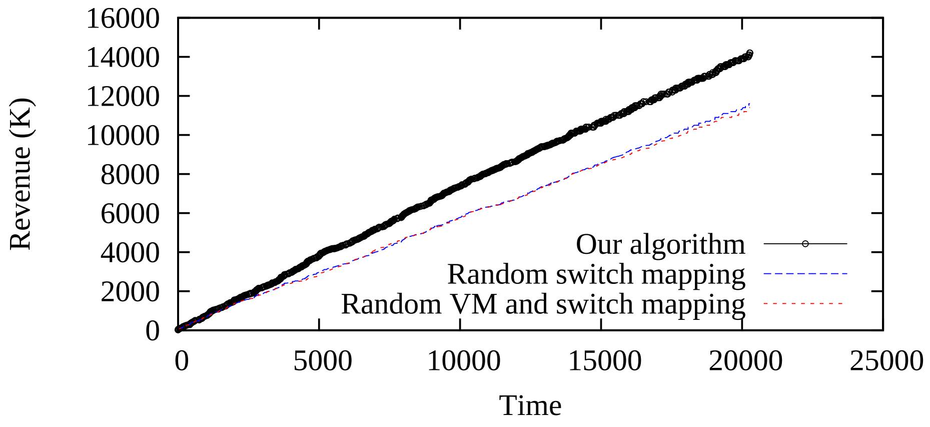
(b) Comparison of CPU Utilization over time.

Figure 3.6: Acceptance ratio and CPU utilization (VL2 topology).

there is less diversity for the embedding algorithm, specially for the switch mapping. Our algorithm has higher acceptance ratio, CPU and network utilization and revenue than the other two algorithms. We also observe that random switch mapping performs better than random VM and switch mapping because random switch mapping utilizes our VM mapping phase for its VM mapping. In both of the above cases, our algorithm has higher acceptance ratio, as well as, higher revenue, which demonstrates the advantages of our VDC embedding algorithm.



(a) Comparison of network utilization over time.

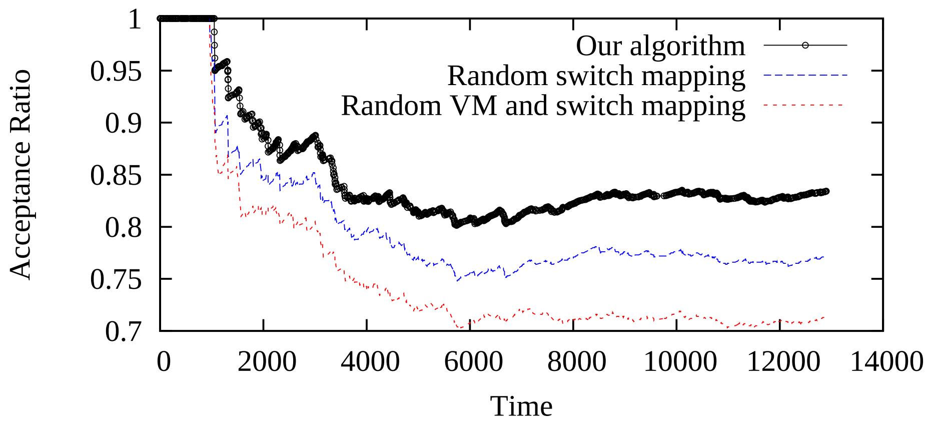


(b) Comparison of revenue over time.

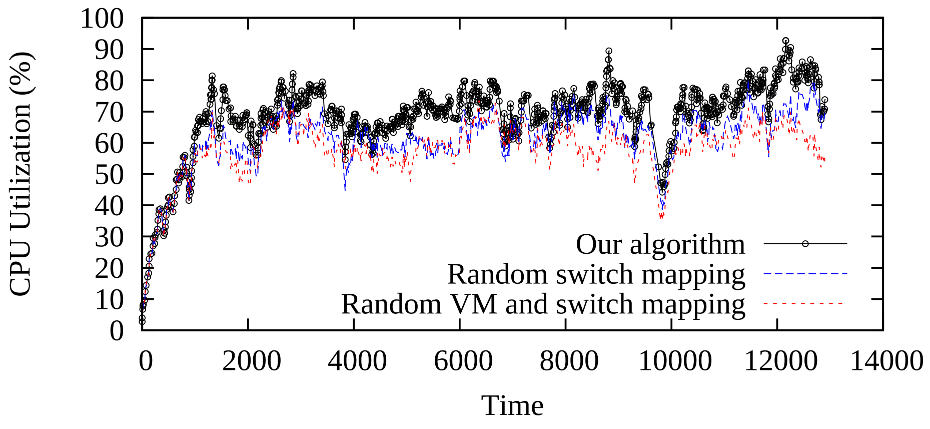
Figure 3.7: Network utilization and revenue (VL2 topology).

3.6 Summary

In this chapter, we first described the existing literature on the VDC embedding problem. Then we provided a new formulation of VDC embedding problem where virtual switches are considered in addition to VMs and virtual links in the VDC requests. After that we discussed a general approach to tackle the embedding problem and then we proposed a three-phase minimum-cost based heuristic to solve the embedding problem. We evaluated



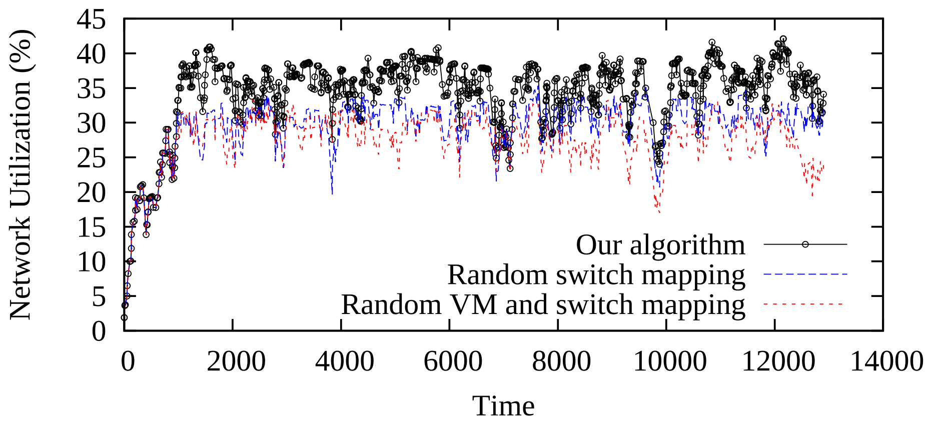
(a) Comparison of acceptance ratio over time.



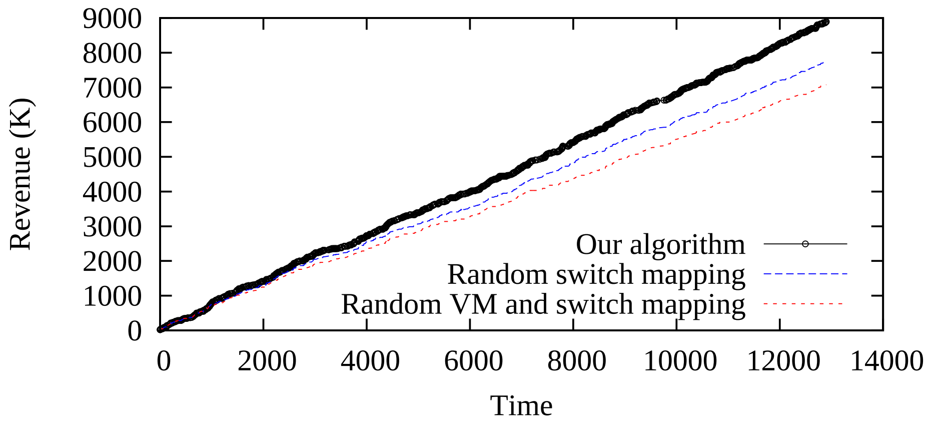
(b) Comparison of CPU utilization over time.

Figure 3.8: Acceptance ratio and CPU utilization (mix of VL2 and star topology).

the performance of our embedding algorithm using a discrete event simulator. We showed that our algorithm has higher acceptance ratio, CPU and network utilization, as well as higher revenue. We did not consider the availability of the VDC in this chapter. However, as service providers are increasingly hosting different business services, providing guaranteed availability of VDCs becomes extremely important. In the next chapter, we propose an allocation scheme devised for providing guaranteed VDC availability.



(a) Comparison of network utilization over time.



(b) Comparison of revenue over time.

Figure 3.9: Network utilization and revenue (mix of VL2 and star topology).

Chapter 4

Survivable Virtual Data Center Embedding

4.1 Introduction

In the previous chapter, we have proposed a resource allocation scheme for VDCs aiming to achieve maximizing revenue. However, we did not consider the availability of VDC in our embedding algorithm. Service disruption even for seconds, may cause high losses in revenue and reputation for an SP. On the other hand, InP may end up paying huge penalties for the violation of service level agreement. Hence, providing guaranteed availability of VDCs is of great importance both to SP and InP.

Recently, few proposals have been made to improve the availability of VDCs either through availability-aware resource allocation schemes or redundancy provisioning techniques [45, 25, 46, 18, 47]. However, these works did not consider the heterogeneity of the underlying physical components. In this chapter, we address this compelling challenge and propose **Hi**gh-availability **V**irtual **I**nfrastructure Management (Hi-VI) framework [35] that takes into account the heterogeneity of data center computing and networking equipments to dynamically provision backup resources in order to ensure the required VDC availability.

The remainder of the chapter is organized as follows: Section 4.2 surveys previous work on failure characterization in data centers highlighting the heterogeneity in terms of failure rate and availability. We also summarize representative work on availability-aware VDC allocation schemes and discuss their limitations. We present in Section 4.3 and 4.4 our technique to compute the availability of a VDC and we provide the mathematical formulation of the availability-aware VDC allocation problem. The proposed solution is then described in Section 4.4.2. Simulation results are discussed in Section 4.5. Finally, we draw our conclusions in Section 4.6.

4.2 Literature Survey

In this section, we present related work on data center failure characterization and representative proposals addressing survivable resource allocation in virtualized data centers.

4.2.1 Failure Characterization in Data Centers

Gill *et al.* [25] presented a large-scale study of failures in several Microsoft data centers over one year. The authors characterized failures of networking devices and assessed the impact of their failures and the effectiveness of network redundancy in data centers. Furthermore, they observed that the failure rates of different equipments can vary significantly depending on their type (servers, top-of-rack switches, aggregation switches, routers, load balancers) and model. For instance, Load Balancers (LBs) exhibit high probability of failure during one-year period (over 20%), whereas switches have lower failure probability (less than 5%). Furthermore, failure rates of different devices are unevenly distributed. For example, the number of failures across load balancers are highly variable with a few outlier LB devices experiencing more than 400 failures over the one year period. Finally, the analysis of failure traces revealed that correlated device and link failures are extremely rare.

Wu *et al.* [45] presented an automated failure mitigation system called *NetPilot*, which alleviates failures in large scale data center network before operators diagnose and repair the root cause. The authors built their system based on an analysis of failures in several production data center networks over a six-month period. They identified three main causes of failures: software failures which constitute 21% of the total number of failures, hardware failures accounting for 18% and finally misconfigurations, the most dominant source of the failures (38%). The authors found that usually simple steps of mitigation are very effective in reducing repair times. However, certain failures incur much more repair time and hence cause significant network downtime. This concurs with the finding of [27] that reported that more than 95% of network failures can be fixed within 10 minutes whereas at least 0.09% of them can take more than 10 days to resolve. This again shows the heterogeneity of the failures in terms of repair times and potential impact.

Vishwanath *et al.* [43] analyzed failures of more than $100,000$ servers deployed in multiple Microsoft data centers over a duration of 14 months. They found that hard disk, memory and raid controller failures were the main reason for server failures. For instance, they reported that failures of hard disks represent 78% of the total failures causing service disruption. They also noticed a high correlation between the number of disk drives in the server and the number of server failures. In addition, they found that servers that have experienced failures are likely to fail again in the near future. This results in a skewed distribution of server failure rate.

Based on these observations, we can summarize the main characteristics of failures in data centers as follows: (1) failure rates and availability are heterogenous across the physical components, (2) correlated failures are extremely rare. This suggests that heterogeneity should be considered when mapping virtual components onto the physical infrastructure. Furthermore, since correlated failures are rare, it is reasonable to assume failures to be independent.

4.2.2 Survivable embedding in Virtualized Data Center

Bodik *et al.* [18] proposed an allocation scheme that aims at minimizing the impact of failures (i.e., maximizing fault-tolerance) on the virtual data center (termed "service" in the paper) while reducing bandwidth usage in the core of the data center network. The VDC fault-tolerance is measured by the *worst-case survival* metric defined as the fraction of VMs belonging to the same VDC that remain operational during a single worst-case failure. However, this work does not consider the availability of the underlying physical components. Besides, considering only worst-case failure (which happens in aggregation/core switches) results in ignoring other failures (e.g., in top-of-rack switches). Furthermore, the authors assume a physical server can only host one VM from the same VDC. As a result, the approach tends to extensively spread VMs, leading to higher bandwidth usage.

Xu *et al.* [46] proposed a VDC allocation scheme that considers embedding backup VMs and virtual links with the goal of minimizing consumed resources. However, they do not consider the availability of the physical machines and they also assume that the number of backups is known beforehand. Yeow *et al.* [48] addressed these limitations and they proposed a reliable VDC embedding scheme that achieves the desired availability for VDCs by estimating the required number of backups for the virtual nodes based on the availability of physical machines. They also introduced a technique to allow VDCs to share backup nodes and links. However, this work considers only homogeneous clusters, which means all servers have same probability of failure and availability, which is an unrealistic assumption. They also assumed that a physical node cannot host more than one virtual node from the same VDC.Yu *et al.* [49] proposed a backup provisioning scheme for improving virtual infrastructure survivability while minizing resources used to provision backups. Assuming that only a single failure could occur at a time, they formulate a Mixed Integer Linear Program (MILP) that estimates the required number of redundant nodes and their placement in order to minimize networking resources provisioned for the backup nodes.

Rahman *et al.* [37] presented two policies for solving survivable virtual network embedding problem. The first policy addresses failures proactively by provisioning backup paths for potential failures in the future, however, this approach may lead to the wastage of up to 50% of physical resources. The second policy heuristic is a reactive approach that precomputes a set of possible backup detours for each substrate link.When a substrate link fails, the affected virtual links are rerouted along one of the backup detours. However, this approach does not consider multiple link failures.

Table 4.1 compares the features of survivable embedding proposals. The limitations of the state of the art research can be summarized as follows:

- Previous proposals have either ignored the availability of the underlying physical components (e.g., [18, 46]) or considered that the cluster is homogenous, i.e., nodes have similar failure rates and availability (e.g., [48]). Hence, it is more realistic and challenging to consider the heterogeneity existing in production data center environments in order to take more informed resource allocation decisions and improve availability of the embedded VDCs.

- Existing proposals (e.g., [48, 49]) assume a single physical server can host at most one virtual node from the same VDC. This assumption is not realistic in production environments. For instance, if a VDC comprises hundreds of VMs, these schemes map them onto hundreds of physical servers. This results in higher bandwidth consumption and requires more physical nodes to be active. Ideally, it should be possible to allow multiple VMs from the same VDC to be hosted on a single physical node if the required availability is satisfied. This will result in reduced bandwidth usage and less active physical nodes, and lead to reduced enegy costs, increased VDC acceptance and InP revenue.

- Previous work (e.g., [48]) does not consider the failure rate of networking components (e.g., physical switches) when computing the VDC availability. However, virtual links are mapped onto physical paths that may cross multiple physical switches. It is therefore mandatory to factor in switches' availability when computing the availability of VDCs.

Table 4.1: Comparison of survivable embedding schemes

Proposals	Backup provisioning		Estimation of the number of backups	Heterogeneity	Computing Availability	VM Colocation
	Virtual Nodes	Virtual Links				
Xu *et al.* [46]	Yes	No	No	No	No	Yes
Yu *et al.* [49]	Yes	No	No	No	No	No
Yeow *et al.* [48]	Yes	No	Yes	No	Yes	No
Rahman *et al.* [37]	No	Yes	N/A	N/A	No	N/A
Bodik *et al.* [18]	No	No	No	No	No	No
Hi-VI	Yes	Yes	Yes	Yes	Yes	Yes

In this paper, we aim to address these limitations by developping a technique to estimate VDC availability in a heterogeneous environment and then leverage it to devise a more efficient resource allocation scheme that achieves availability requirements and at the same time minimizes energy costs.

4.3 VDC Availability in Heterogeneous Cloud Environments

In this section, we provide a technique to compute the VDC availability in heterogeneous cloud environments.

Once the SP provides the requirements of his VDC in terms of resources and availability, the InP is responsible for mapping the VDC onto the physical data center such that the required availability is satisfied. Hence, the InP should be able to (1) evaluate the availability of the embedded virtual components based on the availability of the underlying physical infrastrucure, and to (2) estimate the number of backups needed to meet the required availability.

In the following, we first describe how we model the physical data center and the VDC requests. We then present a technique to compute the availability of a VDC taking into consideration the heterogenous characteristics of the physical equipments. Finally, we formulate the survivable VDC embedding problem as an optimization problem that minimizes the number of active physical machines, the bandwidth usage in the data center network as well as the number of backups while satisfying the required VDC availability.

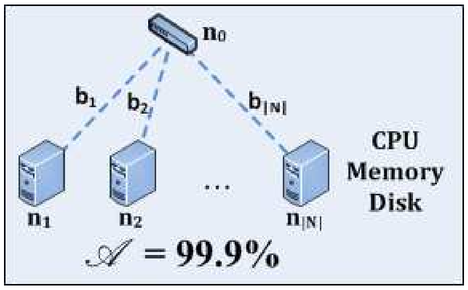
4.3.1 Physical Data Center

We model the data center network as a graph $\bar{G} = (\bar{N}, \bar{L})$ where $\bar{N}$ is the set of physical nodes and $\bar{L}$ is the set of physical links. $\bar{N}$ includes the set of physical machines $\bar{M}$ and the set of physical switches and routers $\bar{S}$ (i.e., $\bar{N} = \bar{M} \cup \bar{S}$). Each physical node $\bar{n} \in \bar{N}$ has a residual capacity $c^r_{\bar{n}}$ for each resource type $r \in R$ where $R = \{1..|R|\}$ is the set of resource types. Each link $\bar{l} \in \bar{L}$ has a residual bandwidth capacity $b_{\bar{l}}$. The availability $\mathscr{A}_{\bar{n}} \in [0, 1]$ of a physical component $\bar{n}$ is given by:

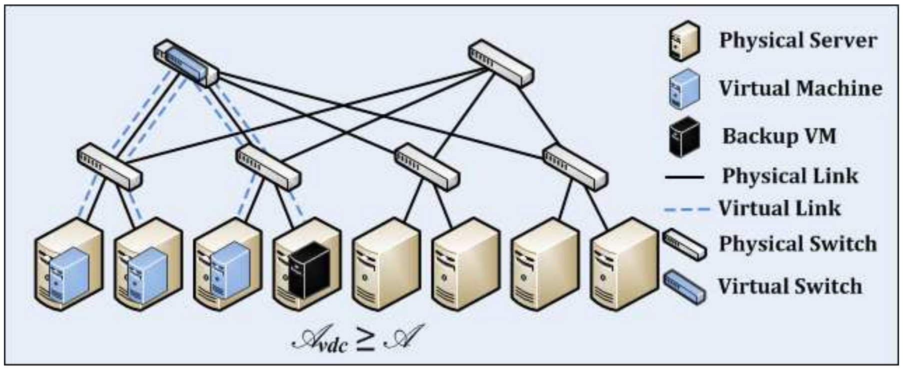
$$\mathscr{A}_{\bar{n}} = \frac{MTBF_{\bar{n}}}{MTBF_{\bar{n}} + MTTR_{\bar{n}}} \tag{4.1}$$

where $MTBF_{\bar{n}}$ and $MTTR_{\bar{n}}$ represent respectively the Mean Time Between Failures and the Mean Time To Repair for the node $\bar{n}$. Both $MTBF_{\bar{n}}$ and $MTTR_{\bar{n}}$ can be obtained from historical records of failure events. Furthermore, we define $\bar{u}_{\bar{n}\bar{l}}$ and $\bar{v}_{\bar{n}\bar{l}}$ as boolean variables that indicate whether a physical node $\bar{n} \in \bar{N}$ is the source and the destination of physical link $\bar{l} \in \bar{L}$, respectively.

4.3.2 VDC Requests



(a) Format of the VDC request.



(b) VDC Mapping (with 1 backup node).

Figure 4.1: A sample VDC request and its embedding in physical data center.

In this work, we limit our study to VDC requests having a star topology as shown in Figure 4.1(b). Such a virtual topology is suitable for hosting many types of applications like web applications, MapReduce and BLAST [15]. Hence, a SP has to specify the number of virtual nodes constituting the VDC, the amount of resources for each of the VMs (i.e., CPU, memory and disk) and links (i.e., bandwidth) as well as the required VDC availability (see Figure 4.1(a)). Similar to a physical data center, a VDC request can be modelled as a graph $G = (N, L)$, where N is the set of virtual nodes (including the virtual switch) and L is the set of virtual links. The required availability of the VDC is denoted by $\mathscr{A}$. Each virtual node $n \in N$ has a capacity c_n^r for each resource type $r \in R$, and each virtual link $l \in L$ has a bandwidth capacity b_l. Since we have only a single virtual switch, we reserve for it the index 0. We also define two boolean variables u_{nl} and v_{nl} to indicate whether a virtual node $n \in N$ is the source or the destination of virtual link $l \in L$, respectively.

Variable Definitions

We hereafter define variables that capture the mapping of virtual nodes and links onto the physical infrastructure. Let $x_{n\bar{n}} \in \{0, 1\}$ be a boolean variable that indicates whether virtual node n is mapped onto the substrate machine $\bar{n}$. Let $f_{l\bar{l}} \in \{0, 1\}$ be a boolean variable that indicates whether physical link $\bar{l}$ is used to embed virtual link l.

We also define $w_{n\bar{s}} \in \{0, 1\}$ that indicates whether physical switch $\bar{s}$ is used to embed the virtual link connecting the virtual switch to the virtual node n. In other words, if $w_{n\bar{s}} = 1$ the failure of physical switch $\bar{s}$ causes the virtual node n to be unavailable. Hence, $w_{n\bar{s}}$ can be expressed as:

$$w_{n\bar{s}} = \begin{cases} 1 & \text{if } \sum_{l \in L} \sum_{\bar{l} \in \bar{L}} (u_{nl} f_{l\bar{l}} u_{s\bar{l}} + u_{nl} f_{l\bar{l}} v_{s\bar{l}} \\ & +v_{nl} f_{l\bar{l}} u_{s\bar{l}} + v_{nl} f_{l\bar{l}} v_{s\bar{l}}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where $u_{nl} f_{l\bar{l}} u_{\bar{s}\bar{l}}$ indicates whether physical link $\bar{l}$ is used to embed virtual link l, and virtual node n is source of l, and physical node $\bar{s}$ is source of $\bar{l}$. We also define $y_{\bar{n}}$ as a boolean variable that indicates whether a physical node $\bar{n} \in \bar{N}$ is used either to embed a VM or switch or to embed a virtual link (as an intermediate node in the physical path).

4.3.3 Computing VDC Availability

In the following, we provide a technique to compute the availability of a VDC request $G = (N, L)$. Let N_B and L_B denote the set of backup nodes and links that are provisioned by the InP in order to improve the availability of the VDC. The resulting graph including the backup links and nodes is denoted by $G' = (N', L')$ where $N' = N \cup N_B$ and $L' = L \cup L_B$

where N_B and L_B are the set of backup nodes and links, respectively. Note that for the considered star topology we have $|N_B| = |L_B|$. A VDC is available if the number of failed virtual nodes is at most the number of provisioned backups. Let $Pr(k)$ be the probability that k virtual nodes fail. Hence, the availability of the whole VDC $\mathscr{A}_{vdc}$ can be written as:

$$\mathscr{A}_{vdc} = \sum_{k=0}^{K} Pr(k) = Pr(0) + \sum_{k=1}^{K} Pr(k) \tag{4.3}$$

Let us first compute the probability that no virtual node fails $Pr(0)$. It can be written as the product of the availability of all physical nodes hosting the VDC components:

$$Pr(0) \quad = \prod_{\bar{n}:y_{\bar{n}}=1} y_{\bar{n}} \mathscr{A}_{\bar{n}} \tag{4.4}$$

Next, let us compute the probability $Pr(k)$ where $k \geq 1$. The failure of k virtual nodes occurs only when physical failures result in k VM failures. Let $g_{\bar{m}}$ be the number of VMs mapped to physical machine $\bar{m}$. It can be written as:

$$g_{\bar{m}} = \sum_{n \in N'} x_{n\bar{m}} \qquad \forall \bar{m} \in \bar{M} \tag{4.5}$$

Let $g_{\bar{s}}$ be the number of VMs that are disconnected if the physical swtich $\bar{s}$ fails. In other words, this switch is used either to embed the virtual switch or as an intermediate node between the physical server hosting the VM and the physical node hosting the virtual switch. We have:

$$g_{\bar{s}} = \sum_{n \in N'} w_{n\bar{s}} \qquad \forall \bar{s} \in \bar{S} \tag{4.6}$$

To evaluate the probability of k virtual nodes failure, we need to consider every possible physical node failure that will lead to k virtual nodes failure. The probability of having a single physical node failure that causes k virtual nodes failure can be written as:

$$\sum_{\bar{n}:g_{\bar{n}}=k} \Big((1 - \mathscr{A}_{\bar{n}}) \prod_{\bar{t} \in \bar{N} \setminus \{\bar{n}\}:y_{\bar{t}}=1} y_{\bar{t}} \mathscr{A}_{\bar{t}} \Big)$$

where $(1 - \mathscr{A}_{\bar{n}})$ is the probability of failure of physical node $\bar{n}$ and $\prod_{\bar{t} \in \bar{N} \setminus \{\bar{n}\}:y_{\bar{t}}=1} y_{\bar{t}} \mathscr{A}_{\bar{t}}$ is the probability that all other nodes used to embed the VDC are available. Note that we consider the failure of physical nodes that can impact k virtual machines (i.e., $g_{\bar{n}} = k$).

However, in practice, multiple physical nodes can fail simultaneously and lead to k virtual node failure. Therefore, we have:

$$Pr(k) \;\geq\; \sum_{\bar{n}:g_{\bar{n}}=k} \Big((1-\mathscr{A}_{\bar{n}}) \prod_{\bar{t}\in\bar{N}\setminus\{\bar{n}\}:y_{\bar{t}}=1} y_{\bar{t}}\mathscr{A}_{\bar{t}}\Big) \tag{4.7}$$

Using Equations 4.3, 4.4, and 4.7, we have:

$$\begin{aligned}\mathscr{A}_{vdc} \;&\geq\; \Bigg(\prod_{\bar{n}:y_{\bar{n}}=1} y_{\bar{n}}\mathscr{A}_{\bar{n}}\Bigg)\\ &+\; \sum_{k=1}^{K}\Bigg(\sum_{\bar{n}:g_{\bar{n}}=k}\Big((1-\mathscr{A}_{\bar{n}}) \prod_{\bar{t}\in\bar{N}\setminus\{\bar{n}\}:y_{\bar{t}}=1} y_{\bar{t}}\mathscr{A}_{\bar{t}}\Big)\Bigg)\end{aligned}$$

This provides a lower bound on the availability of the VDC. Let $\mathscr{A}_{vdc}^{lb}$ denote this lower bound, it can be written as:

$$\begin{aligned}\mathscr{A}_{vdc}^{lb} \;&=\; \Bigg(\prod_{\bar{n}:y_{\bar{n}}=1} y_{\bar{n}}\mathscr{A}_{\bar{n}}\Bigg)\\ &+\; \sum_{k=1}^{K}\Bigg(\sum_{\bar{n}:g_{\bar{n}}=k}\Big((1-\mathscr{A}_{\bar{n}}) \prod_{\bar{t}\in\bar{N}\setminus\{\bar{n}\}:y_{\bar{t}}=1} y_{\bar{t}}\mathscr{A}_{\bar{t}}\Big)\Bigg)\end{aligned} \tag{4.8}$$

That is, the availability of the VDC $\mathscr{A}_{vdc}$ is at least $\mathscr{A}_{vdc}^{lb}$.

4.4 Availability-aware Embedding

4.4.1 Problem Formulation

In the following we formulate the availability-aware embedding problem. We start by describing the embedding constraints then provide the optimization objective function.

- **Embedding constraints:**

When embedding the VDC, there are many constraints that should be satisfied. For instance, in order to ensure that the capacities of physical resources are not violated, the following constraints must be satisfied:

$$\sum_{n\in N'} x_{n\bar{n}} c_n^r \leq c_{\bar{n}}^r \qquad \forall \bar{n} \in \bar{N}, r \in R \tag{4.9}$$

$$\sum_{l\in L'} f_{l\bar{l}} b_l \leq b_{\bar{l}} \qquad \forall \bar{l} \in \bar{L} \tag{4.10}$$

We also require the link embedding to satisfy the flow constraint between every source and destination node pairs in each VDC topology, namely:

$$-\sum_{\bar{l}\in\bar{L}} \bar{v}_{\bar{n}\bar{l}} f_{l\bar{l}} + \sum_{\bar{l}\in\bar{L}} \bar{u}_{\bar{n}\bar{l}} f_{l\bar{l}} = \sum_{n\in N} x_{n\bar{n}} u_{nl} - \sum_{n\in N} x_{n\bar{n}} v_{nl} \qquad \forall l \in L', \bar{n} \in \bar{N} \tag{4.11}$$

Here $\sum_{n\in N} x_{n\bar{n}} u_{nl}$ is equal to 1 if n is the source of the link l of VDC and n is embedded in the physical node $\bar{n}$. Equation 4.11 essentially states that the total outgoing flows of a physical node $\bar{n}$ for a virtual link l is equal to the total incoming flows unless $\bar{n}$ hosts either a source or a destination virtual node.

Furthermore, we need to consider the node placement constraints. For instance, VMs should only be placed in physical servers whereas virtual switches may be placed either in switches (e.g., flowvisor instance [40]) or servers (e.g., open vSwitch instance [9]). Hence, we define $\tilde{x}_{n\bar{n}}$ to indicate whether physical node $\bar{n}$ is able to host virtual node n of the VDC. Thus, if a virtual node n is a virtual machine (not a switch), we have $\tilde{x}_{n\bar{n}} = 0 \, \forall \bar{n} \in \bar{S}$ and $\tilde{x}_{n\bar{n}} = 1 \, \forall \bar{n} \in \bar{M}$. Hence, the the placement constraint can be written as:

$$x_{n\bar{n}} \leq \tilde{x}_{n\bar{n}} \qquad \forall n \in N', \bar{n} \in \bar{N} \tag{4.12}$$

Additionally, we need to ensure that the minimum number of provisioned virtual nodes is at least the number of nodes required by the SP. Hence, we have:

$$\sum_{n\in N'} \sum_{\bar{n}\in\bar{N}} x_{n\bar{n}} \geq |N| \tag{4.13}$$

Furthermore, to ensure that the virtual switch is mapped, the following equation must hold:

$$\sum_{\bar{n}\in\bar{N}} x_{0\bar{n}} = 1 \tag{4.14}$$

Furthermore, $y_{\bar{n}}$ must be equal to 1 if the physical node $\bar{n}$ is used to host any virtual node or used as an intermediate node to embed a virtual link. This implies the following constraints must hold:

$$y_{\bar{n}} \geq x_{n\bar{n}} \qquad \forall n \in N', \bar{n} \in \bar{N} \tag{4.15}$$

$$y_{\bar{n}} \geq w_{n\bar{n}} \qquad \forall n \in N', \bar{n} \in \bar{S} \tag{4.16}$$

$$y_{\bar{n}} \geq f_{l\bar{l}} \bar{u}_{\bar{n}\bar{l}} \qquad \forall \bar{n} \in \bar{N}, l \in L', \bar{l} \in \bar{L} \tag{4.17}$$

$$y_{\bar{n}} \geq f_{l\bar{l}} \bar{v}_{\bar{n}\bar{l}} \qquad \forall \bar{n} \in \bar{N}, l \in L' \tag{4.18}$$

We have also to ensure that the VDC availability $\mathscr{A}_{vdc}^{lb}$ is higher than the required availability. That is:

$$\mathscr{A}_{vdc}^{lb} \;\; \geq \;\; \mathscr{A} \tag{4.19}$$

- **Objective function:**

The goal of the embedding is to minimize the number of the physical nodes used for embedding the VDC as well as the amount of consumed bandwidth while maintaining the constraints of Equations 4.9-4.19. Hence our objective function can be written as follows:

$$\min(\alpha \sum_{\bar{n} \in \bar{N}} y_{\bar{n}} p_{\bar{n}} + \beta \sum_{\bar{l} \in \bar{L}} \sum_{l \in L'} f_{l\bar{l}} b_l + \gamma \sum_{\bar{n} \in \bar{N}} \sum_{n \in N'} (x_{n\bar{n}} \sum_{r \in R} w^r c_n^r)) \tag{4.20}$$

where $p_{\bar{n}}$ is the energy cost defined as:

$$p_{\bar{n}} = \begin{cases} 0 & \text{if the machine } \bar{n} \text{ is already active} \\ t_{c\bar{n}} & \text{if the machine is off} \end{cases} \tag{4.21}$$

$t_{c\bar{n}}$ is cost of turning on the machine $\bar{n}$ and w^r is the weight factor for resource type $r \in R$, which depends on the scarcity of the resource. The weight factor α, β and γ are used to strike the balance between energy cost, communication cost, and computation cost. This optimization problem is $\mathcal{NP}$-hard as it generalizes the multi-dimensional bin packing problem[52]. Therefore, we provide a heuristic to solve the problem in the subsequent section.

4.4.2 Proposed Heuristic

This sub-section describes a heuristic for solving the availability-aware VDC embedding problem that ensures that each embedded VDC satisfies its requirements in terms of availability and resources. The goal is to minimize the number of active machines and the consumed bandwidth in the data center network with the goal of increasing InP's income.

Algorithm 2 Availability-aware VDC embedding

```
1: **VM Mapping Phase:**
2: Sort servers $\bar{M}$ by status (active or not) and availability
3: Sort $N$ by their $size_n^i$ defined in Eq. 4.22
4: for each virtual machine $n \in N$ do
5:     if $EmbedNode(n, 1) = -1$ then
6:         Reject request
7:     end if
8: end for
9: $B \leftarrow 0$
10: while $\mathscr{A}_{vdc}^{lb} < \mathscr{A}$ and $B \leq B_{\max}$ do
11:     if $B = 0$ then
12:         $B = (\min_{\bar{n} \in \bar{N}} \sum_{n \in N} x_{n\bar{n}})$
13:         $EmbedNode(n_{max}, B)$ {$n_{max}$: the node with the largest size.}
14:     else
15:         $B \leftarrow B + 1$
16:         $EmbedNode(n_{max}, 1)$
17:     end if
18: end while
19: **Switch and Link Mapping Phase:**
20: $C_{c,min} \leftarrow \infty$ {$C_{c,min}$ is the minimum communication cost}
21: for each $\bar{n} \in \bar{N}$ do
22:     $cost(\bar{n}) \leftarrow 0$
23:     Compute total communication cost $C_c$
24:     if $C_c < C_{c,min}$ then
25:         $C_{c,min} \leftarrow C_c$ ; $p \leftarrow \bar{n}$ {$p$: candidate physical node for hosting the switch}
26:     end if
27: end for
28: if $B \leq B_{\max}$ then
29:     Accept the VDC request
30: else
31:     Reject the VDC request
32: end if
```

Our algorithm is described in Algorithm 2. The VDC embedding is carried out in two phases: (1) VM mapping, (2) virtual switch and link mapping. All physical servers are sorted based on their status (active or inactive) and availability. Since our aim is to reduce

Algorithm 3 EmbedNode(n,b)

```
1: Input : virtual node $n$, number of replicas $b$
2: Output: Physical node $\bar{n}$ or -1
3: Sort servers $\bar{M}$ by status (active or not) and availability
4: for $i \leftarrow 1, b$ do
5:     Find $\bar{n} \in \bar{M}$ able to host $n$
6:     if Not found then
7:         Return -1
8:     end if
9:     Embed $n$ in $\bar{n}$
10: end for
11: Return $\bar{n}$
```

the number of active servers, the algorithm tries first to embed VMs in the active servers. When a VDC is received, the VMs are sorted in descending order according to size of their requested resources. The size of VM n is captured by $size_n$ defined as:

$$size_n = \sum_{r \in R} w^r c_n^r \qquad \forall n \in N \tag{4.22}$$

where w^r is the weight factor for resource type $r \in R$, which depends on the scarcity of the resource. The intuition is that it is usally harder to map large VMs. The algorithm parses the sorted list of servers and finds the one that can satisfy the resource requirements of the VM n. This aims also at embedding VMs in servers with the highest availability in order to avoid the need for backups. After embedding all VMs $n \in N$, we start mapping the virtual switch and the links. If $\mathscr{A}_{vdc} < \mathscr{A}$, we provision B backups where B is the minimum number of VMs that are embedded in a single physical server. That is:

$$B = \min\big(\sum_{n \in N} x_{n\bar{n}}\big) \qquad \forall \bar{n} \in \bar{N} \tag{4.23}$$

The idea is to provision enough backups to take over the failure of the physical machines hosting the lowest number of VMs. After embedding B backup VMs, we check again $\mathscr{A}_{vdc}$. If it is still lower than the requested availability, another VM backup is provisioned in a new physical node that is not hosting any of the previously embedded VMs. This process is repeated until the required availability is reached. In this case, the VDC is accepted. In order to avoid overly backup provisioning, the InP can set a maximum number of backups $B_{\max}$. If the number of backups exceeds $B_{\max}$, the request is rejected.

Once the VM mapping is done, the virtual switch and link mapping are carried out jointly. To embed a virtual link, we consider the shortest path between the physical node hosting the switch and the one hosting a VM. We define the virtual link communication cost as the total number of hops in the corresponding physical path multiplied by the link bandwidth (i.e., *hop_count* $\times$ *bandwidth*). The total VDC communication cost (C_c) is then defined as the sum of communication costs of all virtual links. In order to find the optimal embedding for the virtual switch and the links, the algorithms computes the VDC communication cost for all possible placements of the virtual switch. The final embedding is the one that minimizes this cost.

In our heuristic, we first try to embed VMs into physical servers that are already active. This leads to less energy consumption, and hence to reduced costs. Furthermore, we try to embed VMs of the same VDC as close as possible to each other. This reduces communication costs between VMs and the consumed bandwidth in the network. Finally, the algorithm adds backup incrementally until the required availability is met to avoid the over-estimation of backup requirement. Therefore, our heuristic takes into consideration the goals stated in the objective function (Eq. 4.20).

4.5 Performance Evaluation

In this section, we evaluate the effectiveness of our availability-aware VDC embedding algorithm (Hi-VI) through simulations. To this end, we simulate a physical data center of 120 physical machines organized into four racks. The data center network consists of 4 top-of-rack switches, 4 aggregation switches, and 4 core switches connected according to the VL2 topology. We assume each physical machine contains 4 CPU cores, 8 GB of memory, 1 TB hard disk space, and 1 Gbps NIC cards. In order to consider the heterogeneity of the data center equipments, availabilities of servers and switches are selected randomly from $\{0.99, 0.999, 0.9999, 0.99999\}$. VDC requests arrive following a Poisson distribution with an average rate of 0.02 requests per second during day time and 0.01 requests per second during night time. This reflects demand fluctuations in data centers. We assume all VDCs have a star topology consisting of a single virtual switch connected to multiple VMs. The number of VMs per VDC is taken randomly between $1-20$. The size of each VM in terms of CPU, memory and disk is chosen randomly between $1-4$ cores, $1-2$ GB of RAM and $1-10$ GB of disk space, respectively. The capacity of virtual links are also generated randomly between $1-100$ Mbps. Furthermore, the required availability for each VDC is generated randomly from $\{0.99, 0.999, 0.9999\}$ chosen purposely to be higher than the availability guaranteed by Google Apps SLA [6]. The lifetimes of VDCs are exponentially distributed with an average of 3 hours. Finally, if a VDC is not accepted immediately because it is not possible to meet the requirements in terms of availability or resources, it is kept in a waiting queue for a maximum of one hour after which it is automatically withdrawn.

Since, previous proposals in the literature ignore equipment heterogeneity in production data centers, it is not possible to directly compare them to Hi-VI. Therefore, we developed a baseline resource allocation algorithm that combines [18] and [48]. Specifically, the baseline operates in two steps: similar to [18], it starts by spreading VMs across active physical nodes in order to maximize the availability. Then, the algorithm provisions a backup VM in a randomly selected physical node and evaluates the VDC availability. This backup provisioning process is repeated until the required availability is satisfied. It is worth noting that, similar to [48], the baseline is oblivious to the existent heterogeneity in terms of failure rates and availability of the underlying physical components (since the placement of backups does not consider the availability of physical nodes).

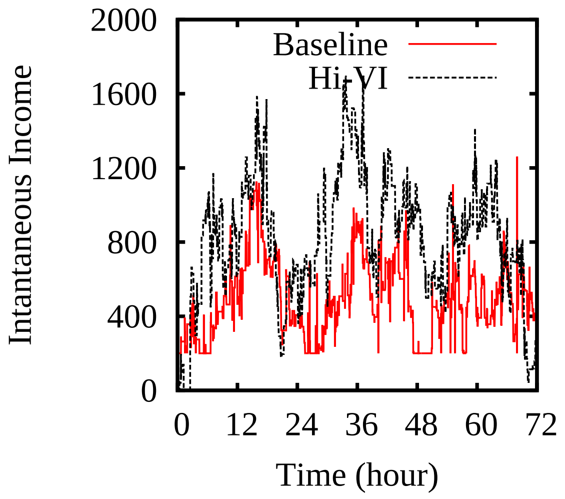


Figure 4.2: Instantaneous Income

We first evaluate the instantaneous income of Hi-VI and the number of accepted VDC requests compared to the baseline algorithm. The instantaneous income is provided by the following equation[1]:

$$\mathcal{R}_{inst} = \sum_{v \in V} \left(\mu^b \sum_{l \in L} b_l^v + \sum_{n \in N} \sum_{r \in R} \mu^r c_n^{vr} \right) - \mu^e \sum_{\bar{n} \in \bar{N}} y_{\bar{n}} E_{\bar{n}} \tag{4.24}$$

[1]In order to compute the instantaneous income, resource demands (in terms of CPU, memory and bandwidth) are normalized between 0 and 1.

where $\mu^b$and μ^r are the unit selling prices for bandwidth and resource type r for a single timeslot, respectively. V is the set of embedded VDCs at the current timeslot and the superscript v refers to VDC number v. The energy cost paid by the InP and the energy consumed by machine $\bar{n}$ during a timeslot are denoted by μ^e and $E_{\bar{n}}$, respectively.

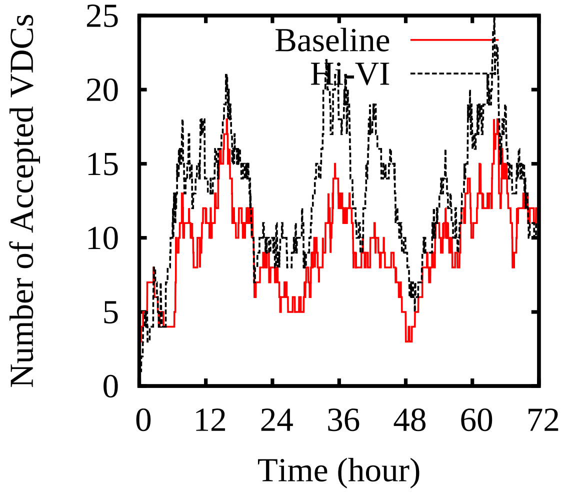


Figure 4.3: Number of accepted VDCs.

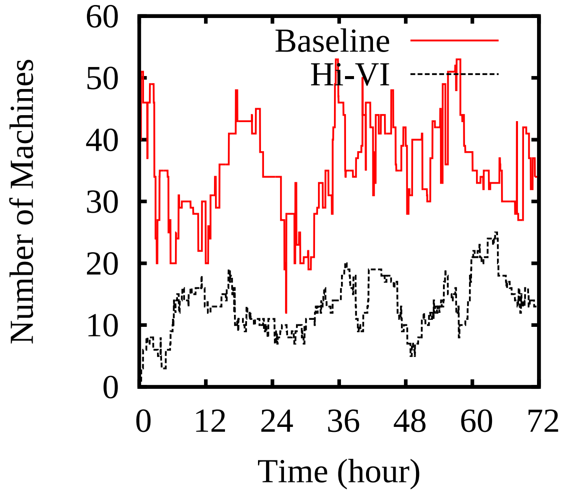
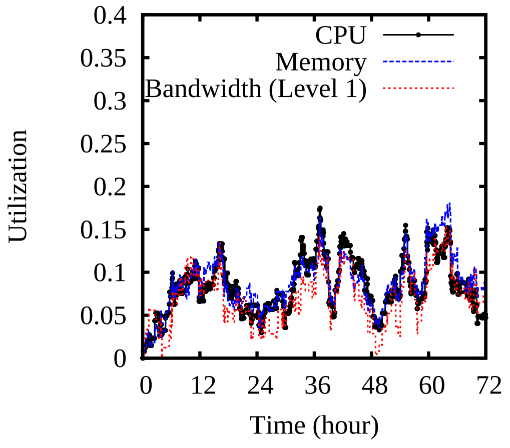
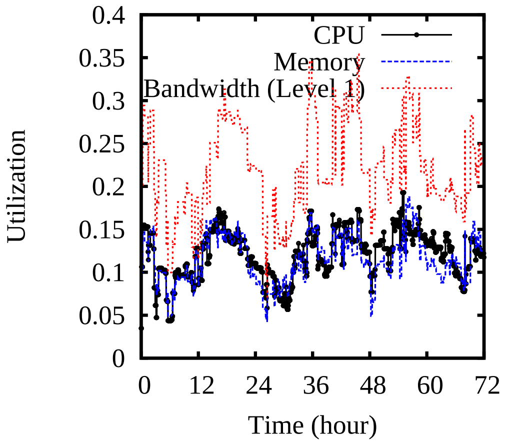


Figure 4.4: Number of active physical machines over time

Figure 4.2 shows that Hi-VI leads to much higher instantaneous income than the baseline. Figure 4.3 confirms that our algorithm accepts more VDC requests than the baseline. One reason for this higher income is the higher acceptance of VDC requests. Another reason is that the number of used physical machines is higher with the baseline algorithm than with Hi-VI (Figure 4.4). This is because the baseline algorithm spreads the VMs across the physical machines, and hence turns on more servers. Thus, it leads to a higher energy costs than Hi-VI.



(a) Hi-VI

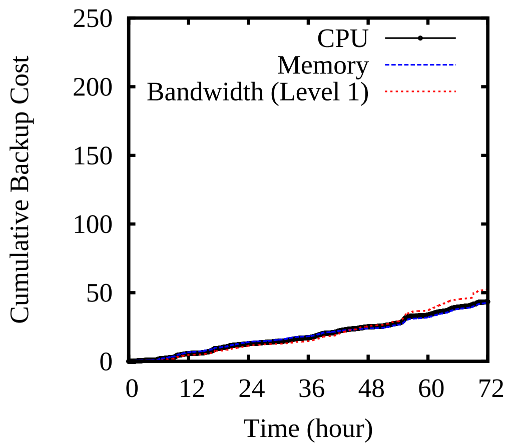


(b) Baseline

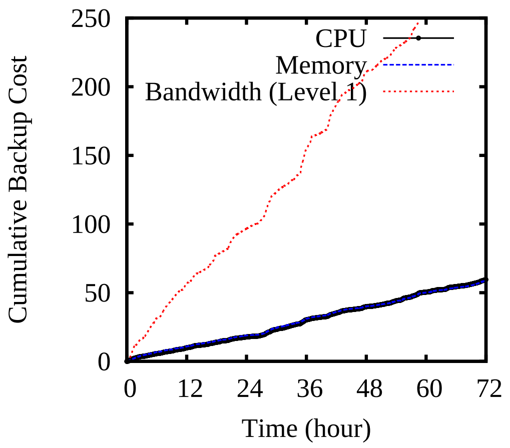
Figure 4.5: Utilization of CPU, memory and bandwidth.

The utilization of the different resources (CPU, memory, and bandwidth) for Hi-VI and the baseline is depicted in Figure 4.5. We can notice that the utilization of CPU and memory are comparable for both algorithms. However, the baseline has accepted much less VDCs, which means that the baseline has allocated a lot of resources for provisioning the backups. We also notice a significant difference in the bandwidth utilization. The baseline requires more bandwidth than Hi-VI, although it accepts less VDC requests. The baseline spreads the VMs across the physical servers and thus uses more bandwidth to embed the virtual links.

Finally, Figure 4.6 shows the cumulative backup costs of CPU, memory and bandwidth for both algorithms. The backup cost of a VDC is computed as the amount of resources used by the provisioned backup multiplied by its lifetime. We can see that the baseline has allocated much more backup resources than Hi-VI in terms of CPU, memory and



(a) Hi-VI



(b) Baseline

Figure 4.6: Backup cost.

bandwidth. These results clearly show that Hi-VI outperforms the baseline in terms of income, accepted VDC requests and significantly reduces backup costs.

4.6 Summary

In this chapter, we proposed a resource allocation framework for VDCs (Hi-VI) that guarantees VDC availability considering the heterogeneity of the production data center components in terms of failure rates and availability. Simulation results show the effectiveness of Hi-VI. Compared to heterogeneity-oblivious solutions, Hi-VI increases by up to 20% the InP net income while minimizing by up to 40% the operational costs.

Chapter 5

Conclusion

Cloud computing has evolved as the key computing model now-a-days for hosting different kinds of applications and services. In this model, Infrastructure Providers (InPs) own data centers and lease resources to Service Providers (SPs) in an on-demand basis.

The first challenge addressed in this thesis is the Virtual Data Center (VDC) embedding problem. The goal of the VDC embedding problem is to allocate resources to VDCs while minimizing the bandwidth usage in the data center and maximizing the cloud provider's revenue. We proposed a new VDC embedding solution that explicitly considers the embedding of virtual switches and virtual links in addition to virtual machines. Simulations show that our solution results in high acceptance rate of VDC requests, less bandwidth consumption in the data center network, and increased revenue for the cloud provider

The second challenge that we addressed in this thesis is about providing VDCs with defined availability. As service availability is a prime concern for service providers, InPs are prompted to roll out computing and networking resources with hard availability guarantees. Despite recent research on the problem, none has considered the heterogeneity of the data center components in terms of failure rates and availability to estimate the required amount of backup resources to reserve in order to ensure the targeted availability. We proposed a High-availability Virtual Infrastructure (Hi-VI) management framework that jointly allocates resources for VDCs and their backups while minimizing total energy costs. Hi-VI uses a novel technique to compute the availability of a VDC that considers both (1) the heterogeneity of the data center networking and computing equipment, and (2) the number of redundant virtual nodes and links provisioned as backups. Simulations demonstrate the effectiveness of our framework compared to heterogeneity-oblivious solutions in terms of revenue and the number of physical servers used to embed the VDCs.

We summarize the contributions of this thesis in the next section. Future research directions are then provided at the end of this chapter.

5.1 Summary of Contribution

The contributions of this thesis are summarized below:

Background of Virtualized Data Center offers the following:

- Introduction to the concepts of virtualized/virtual data centers.
- Identification of the different stakeholders in data center virtualization environments.

Embedding of VDC provides the following:

- A critical analysis of the existing proposals addressing the VDC embedding problem.
- A mathematical formulation of VDC embedding problem where along with VMs and virtual links, we also considered virtual switches in the VDC requests.
- A novel embedding algorithm: a three-phase minimum-cost-flow-based heuristic to solve the VDC embedding problem, considering residual bandwidth, server defragmentation, communication costs and load balancing.
- Through simulations, we have shown that our embedding algorithm was able to embed VMs and virtual links, as well as virtual switches and achieves high acceptance ratio, CPU and network utilization, as well as higher revenue.

Hi<u>gh</u>-availability <u>V</u>irtual <u>I</u>nfrastructure Management (Hi-VI) framework delivers the following:

- A survey of existing work on data center failure characterization and representative proposals addressing survivable resource allocation in virtualized data centers.
- A VDC management framework that takes into account the heterogeneity of cloud data center equipments to dynamically provision backup resources in order to ensure the required VDC availability.

- Through simulations, we demonstrated that Hi-VI is able to satisfy VDC's availability and resource requirements while minimizing operational costs (notably energy costs). Compared to heterogeneity-oblivious solutions, Hi-VI increases by up to 20% the cloud provider's net income while minimizing by up to 40% the operational costs.

5.2 Future Research

Our work on resource allocation in virtualized data centers can be potentially extended in a number of directions. In the following, we will discuss some of the most important ones:

VM Migration VM migration allows the InP to improve resource utilization and communication locality, mitigate performance hotspots, achieve fault tolerance, reduce energy consumption, and facilitate system maintenance activities. However, these benefits come with migration cost that includes communication cost, service disruption, and management overhead. Hence, it would be interesting to explore how to improve our embedding scheme with VM migration capability.

Embedding across multiple data centers In this thesis, we have considered the VDC embedding problem in a single data center. However, the InP may have a distributed cloud infrastructure including multiple data centers at different physical locations. The InP may want to embed VDCs across multiple data centers for several reasons, for instance, to provide geographical redundancy through multiple data centers to mitigate natural disasters. Also, the InP can achieve reduction in energy costs by considering the fluctuation of electricity prices over time and data center locations. Therefore, embedding VDCs across a distributed infrastructure is another potential research challenge.

Reducing Carbon Foot Print According to the Environmental Protection Agency (EPA) data centers consumed about 3% of the US total electricity use in 2001 [4]. Furthermore, estimated energy consumption of data center servers was about 2% of the world's electricity. It has been reported recently that, in 2012, the data centers around the world are responsible for 0.25 percent of the worldwide carbon emission, which constitutes 10% of information and communication technologies (ICT) emissions [13]. Other reports [5, 26] also confirm the high power consumption of data centers. As a consequence, InPs are facing tremendous pressure to operate on renewable sources of energy (*e.g.*, solar and wind power) to make their infrastructures more green and environment-friendly.

Hence, to address this, the InPs need to design efficient VDC embedding algorithms that minimize the carbon footprint of their data centers by considering usage of renewables and carbon emissions in different locations.

VDC Topology in Hi-VI The proposed Hi-VI framework considers only VDC requests having a star topology in order to simplify the computation of the VDC availability. In the future, Hi-VI can be extended to consider more general VDC requests having other topologies.

References

[1] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/. Accessed: 2014-01-15.

[2] Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3/. Accessed: 2014-01-15.

[3] Downtime Outages and failures - understanding their true costs. http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html. Accessed: 2014-01-15.

[4] Energy Efficiency and Sustainability of Data Centers. http://www.sigmetrics.org/sigmetrics2011/greenmetrics/Carey_GreenMetricsKeynote060711.pdf.

[5] Energy Logic: Reducing Data Center Energy Consumption by Creating Savings that Cascade Across Systems. http://www.cisco.com/web/partners/downloads/765/other/Energy_Logic_Reducing_Data_Center_Energy_Consumption.pdf.

[6] Google Apps Service Level Agreement. http://www.google.com/apps/intl/en/terms/sla.html. Accessed: 2014-01-15.

[7] Google Compute Engine. https://cloud.google.com/. Accessed: 2014-01-15.

[8] LEMON Graph Library. http://lemon.cs.elte.hu/trac/lemon. Accessed: 2014-01-10.

[9] Open vSwitch. http://openvswitch.org/. Accessed: 2014-01-10.

[10] VMware. http://www.vmware.com. Accessed: 2014-01-15.

[11] Xen. http://xen.org. Accessed: 2014-01-15.

[12] Data Center: Load Balancing Data Center Services SRND, 2004.

[13] ITU, Toolkit on Environmental Sustainability for the ICT Sector (ESS). http://www.itu.int/ITU-T/climatechange/ess/index.html, 2012.

[14] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In *Proceedings ACM SIGCOMM*, August 2008.

[15] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards Predictable Datacenter Networks. In *Proceedings ACM SIGCOMM*, August 2011.

[16] MF. Bari, R. Boutaba, R. Esteves, L. Granvilley, M. Podlesny, M. Rabbani, Q. Zhang, and F. Zhani. Data Center Network Virtualization: A Survey. *to appear in IEEE Communications Surveys and Tutorials*, 15(2):909–928, 2012.

[17] T. Benson, A. Akella, A. Shaikh, and S. Sahu. CloudNaaS: A Cloud Networking Platform for Enterprise Applications. In *Proceedings ACM SOCC*, June 2011.

[18] Peter Bodík, Ishai Menache, Mosharaf Chowdhury, Pradeepkumar Mani, David A. Maltz, and Ion Stoica. Surviving failures in bandwidth-constrained datacenters. In *Proceedings of ACM SIGCOMM*, 2012.

[19] N. F. Butt, N. M. M. K. Chowdhury, and R. Boutaba. Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding. In *Proceedings IFIP Networking*, May 2010.

[20] D. Carr. How Google Works. July 2006.

[21] M. Chowdhury and R. Boutaba. A Survey of Network Virtualization. *Computer Networks*, 54(5):862–876, 2010.

[22] M. Chowdhury, M.R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, 20(1):206 –219, feb. 2012.

[23] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2004.

[24] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings USENIX OSDI*, December 2004.

[25] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of ACM SIGCOMM*, 2011.

[26] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *ACM Computer Communication Review*, 39(1):68–73, 2009.

[27] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: a scalable and flexible data center network. In *Proceedings of ACM SIGCOMM*, 2009.

[28] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In *Proceedings ACM SIGCOMM*, August 2009.

[29] C. Guo, G. Lu, H.J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In *Proceedings ACM CoNEXT*, December 2010.

[30] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese. NetShare: Virtualizing Data Center Networks across Services, May 2010.

[31] C. Leiserson. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, 1985.

[32] Xiaoqiao Meng, V. Pappas, and Li Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[33] L. Popa, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. FairCloud: Sharing The Network In Cloud Computing. In *Proceedings ACM Hotnets*, November 2011.

[34] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica. A Cost Comparison of Datacenter Network Architectures. In *Proceedings ACM CoNext*, November 2010.

[35] M. G. Rabbani, Mohamed Faten Zhani, and Raouf Boutaba. On achieving high survivability in virtualized data centers. *To Appear in IEICE Transactions on Communications*, E97-B(1), January 2014.

[36] Md Rabbani, Rafael Esteves, Maxim Podlesny, Gwendal Simon, Lisandro Zambenedetti Granville, and Raouf Boutaba. On tackling virtual data center embedding problem. In *IM*, 2013.

[37] M. Rahman and R. Boutaba. SVNE: Survivable virtual network embedding algorithms for network virtualization. volume 10, pages 105–118, 2013.

[38] M. R. Rahman, I. Aib, and R. Boutaba. Survivable Virtual Network Embedding. In *Proceedings IFIP Networking*, May 2010.

[39] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes. Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks. In *Proceedings WIOV*, June 2011.

[40] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Can the production network be the testbed? In *Proceedings of USENIX conference on Operating systems design and implementation*, OSDI, 2010.

[41] A. Shieh, S. Kandulaz, A. Greenberg, C. Kim, and B. Saha. Sharing the Data Center Network. In *Proceedings USENIX NSDI*, March 2011.

[42] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. ISP Topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.

[43] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of ACM Symposium On Cloud Computing*, SOCC, 2010.

[44] G. Wang and E. Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *Proceedings IEEE INFOCOM*, March 2010.

[45] Xin Wu, Daniel Turner, Chao-Chih Chen, David A. Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. Netpilot: automating datacenter network failure mitigation. *SIGCOMM Comput. Commun. Rev.*, 42(4), August 2012.

[46] Jielong Xu, Jian Tang, K. Kwiat, Weiyi Zhang, and Guoliang Xue. Survivable virtual infrastructure mapping in virtualized data centers. In *IEEE International Conference on Cloud Computing (CLOUD)*, 2012.

[47] Wai-Leong Yeow, Cedric Westphal, and Ulas C. Kozat. Designing and embedding reliable virtual infrastructures, March 2010.

[48] Wai-Leong Yeow, Cedric Westphal, and Ulas C. Kozat. Designing and embedding reliable virtual infrastructures. *SIGCOMM Comput. Commun. Rev.*, 41(2), April 2011.

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

[50] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. *ACM Computer Communication Review*, 38(2):17–29, April 2008.

[51] Q. Zhang, M. F. Zhani, Q. Zhu, S. Zhang, R. Boutaba, and J. Hellerstein. Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments. In *Proceedings IEEE/ACM International Conference on Autonomic Computing (ICAC)*, September 2012.

[52] Mohamed Faten Zhani, Qi Zhang, Gwendal Simon, and Raouf Boutaba. VDC Planner:dynamic migration-aware virtual data center embedding for clouds. In *IM*, 2013.