Doctoral Thesis

Università degli Studi di Napoli Federico II

Green Resource Management in Distributed Cloud Infrastructures

Author:
Giovanni Battista Fioccola

Supervisor:
Prof. Giorgio Ventre
Co-advisor:
Eng. Pasquale Donadio
Coordinator:
Prof. Franco Garofalo

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Department of Electrical Engineering and Information Technologies

March 2016

To the loving memory of my grandparents, Antonio and Filomena.

UNIVERSITY OF NAPLES FEDERICO II

Abstract

Computer Science and Control Systems Engineering
Department of Electrical Engineering and Information Technologies

Doctor of Philosophy

Green Resource Management in Distributed Cloud Infrastructures

by Giovanni Battista Fioccola

Computing has evolved over time according to different paradigms, along with an increasing need for computational power. Modern computing paradigms basically share the same underlying concept of *Utility Computing*, that is a service provisioning model through which a shared pool of computing resources is used by a customer when needed. The objective of *Utility Computing* is to maximize the resource utilization and bring down the relative costs. Nearly a decade ago, the concept of *Cloud Computing* emerged as a virtualization technique where services were executed remotely in a ubiquitous way, providing scalable and virtualized resources. The spread of *Cloud Computing* has been also encouraged by the success of the virtualization, which is one of the most promising and efficient techniques to consolidate system's utilization on one side, and to lower power, electricity charges and space costs in data centers on the other. In the last few years, there has been a remarkable growth in the number of data centers, which represent one of the leading sources of increased business data traffic on the Internet. An effect of the growing scale and the wide use of data centers is the dramatic increase of power consumption, with significant consequences both in terms of environmental and operational costs. In addition to power consumption, also carbon footprint of the Cloud infrastructures is becoming a serious concern, since a lot of power is generated from non-renewable sources. Hence, energy awareness has become one of the major design constraints for Cloud infrastructures. In order to face these challenges, a new generation of energy-efficient and eco-sustainable network infrastructures is needed. In this thesis, a novel energy-aware resource orchestration framework for distributed Cloud infrastructures is discussed. The aim is to explain how both network and IT resources can be managed while, at the same time, the overall power consumption and carbon footprint are being minimized. To this end, an energy-aware routing algorithm and an extension of the OSPF-TE protocol to distribute energy-related information have been implemented.

Contents

List of Figures

List of Tables

Abbreviations

AAA	**A**uthentication **A**uthorization **A**ccounting
ADM	**A**dd-**D**rop **M**ultiplexer
ALR	**A**daptive **L**ink **R**ate
ANN	**A**rtificial **N**eural **N**etwork
AON	**A**ll-**O**ptical **N**etwork
API	**A**pplication **P**rogramming **I**nterface
AS	**A**utonomous **S**ystem
ASBR	**A**utonomous **S**ystem **B**order **R**outer
ASON	**A**utomatically **S**witched **O**ptical **N**etwork
ASTB	**A**pplication **S**pecific **T**opology **B**uilder
ASTN	**A**utomatically **S**witched **T**ransport **N**etwork
ATM	**A**synchronous **T**ransfer **M**ode
BER	**B**it **E**rror **R**ate
BFD	**B**est-**F**it **D**ecreasing
BGP	**B**order **G**ateway **P**rotocol
CAPEX	**CAP**ital **EX**penditure
CDN	**C**ontent **D**elivery **N**etworks
CPC	**C**loud **P**ower **C**ontroller
CPM	**C**loud **P**ower **M**eter
CR-LDP	**C**onstraint-based **R**outed **L**abel **D**istribution **P**rotocol
CRMAD	**C**loud **R**esource **M**onitor and **A**nomaly **D**etection
CRUE	**C**loud **R**esource **U**tilization **E**stimator
CSA	**C**lient **S**ystem **A**gent
CSP	**C**onstraint **S**atisfaction **P**roblem
CUE	**C**arbon **U**sage **E**ffectiveness

DCeP	**D**ata **C**enter **e**nergy **P**roductivity
DCiE	**D**ata **C**enter **i**nfrastructure **E**fficiency
DCIM	**D**ata **C**enter **I**nfrastructure **M**anagement
DCPM	**D**ata **C**enter **P**redictive **M**odeling
DHCP	**D**ynamic **H**ost **C**onfiguration **P**rotocol
DL	**D**omain **L**eader
DM	**D**omain **M**anager
DPS	**D**ynamic **P**ower **S**caling
DRAGON	**D**ynamic **R**esource **A**llocation in **G**MPLS **O**ptical **N**etworks
DVFS	**D**ynamic **V**oltage **F**requency **S**caling
DW	**D**igital **W**rapper
DWDM	**D**ense **W**avelength **D**ivision **M**ultiplexing
EC2	**E**lastic **C**ompute **C**loud
EDF	**E**arliest **D**eadline **F**irst
EIA	**E**nergy **I**nformation **A**dministration
ERO	**E**xplicit **R**oute **O**bject
ETSI	**E**uropean **T**elecommunications **S**tandards **I**nstitute
FFD	**F**irst-**F**it **D**ecreasing
FSC	**F**iber **S**witched **C**apable
GbE	**G**igabit **E**thernet
GHG	**G**reenhouse **G**as
GMP	**G**reen **M**igration **P**lan
GMPLS	**G**eneralized **M**ulti-**P**rotocol **L**abel **S**witching
GP	**G**reen **P**ath
GRE	**G**eneral **R**outing **E**ncapsulation
HAC	**H**igh **A**vailability **C**luster
HAL	**H**ardware **A**bstraction **L**ayer
HDM	**H**uman **D**ecision **M**aker
HPC	**H**igh **P**erformance **C**luster
HTC	**H**igh **T**hroughput **C**luster
IaaS	**I**nfrastructure **a**s **a** **S**ervice
ICT	**I**nformation and **C**ommunications **T**echnology
IDM	**ID**entity **M**anagement

IGP	**I**nterior **G**ateway **P**rotocol
ILP	**I**nteger **L**inear **P**rogramming
IoT	**I**nternet **o**f **T**hings
IP	**I**nternet **P**rotocol
IPC	**I**nstructions **P**er **C**ycle
IPMI	**I**ntelligent **P**latform **M**anagement **I**nterface
IQR	**I**nter**q**uartile **R**ange
ISP	**I**nternet **S**ervice **P**rovider
IS-IS	**I**ntermediate **S**ystem to **I**ntermediate **S**ystem
ITM	**IT M**anager
JVM	**J**ava **V**irtual **M**achine
LAN	**L**ocal **A**rea **N**etwork
LC	**L**ocal **C**ontroller
LMP	**L**ink **M**anagement **P**rotocol
LPI	**L**ow **P**ower **I**dle
LR	**L**ocal **R**egression
LRR	**L**ocal **R**egression **R**obust
LSA	**L**ink **S**tate **A**dvertisement
LSC	**L**ambda **S**witched **C**apable
LSD	**L**ink **S**tate **D**atabase
LSP	**L**abel **S**witched **P**ath
LSR	**L**abel **S**witching **R**outer
LVM	**L**ogical **V**olume **M**anager
MAD	**M**edian **A**bsolute **D**eviation
MC	**M**aximum **C**orrelation
MI	**M**illion **I**nstructions
MIPS	**M**illion **I**nstructions **P**er **S**econd
MMT	**M**inimum **M**igration **T**ime
MP	**M**anagement **P**oint
MPC	**M**emory accesses **P**er **C**ycle
MPLS	**M**ulti-**P**rotocol **L**abel **S**witching
MRI	**M**agnetic **R**esonance **I**maging
MU	**M**inimum **U**tilization

NARB	**N**etwork **A**ware **R**esource **B**roker
NAT	**N**etwork **A**ddress **T**ranslation
NAVMP	**N**etwork-**A**ware **V**irtual **M**achine **P**lacement
NFV	**N**etwork **F**unction **V**irtualization
NIST	**N**ational **I**nstitute of **S**ecurity and **T**echnologies
NM	**N**etwork **M**anager
NMS	**N**etwork **M**anagement **S**ystem
NPA	**N**on **P**ower **A**ware
NPE	**N**etwork **P**ower **E**ffectiveness
OAM&P	**O**peration **A**dministration **M**aintenance and **P**rovisioning
OCC	**O**ptical **C**onnection **C**ontroller
OPEX	**OP**erating **EX**penditure
OS	**O**perating **S**ystem
OSPF	**O**pen **S**hortest **P**ath **F**irst
OTF	**O**verload **T**ime **F**raction
OTN	**O**ptical **T**ransport **N**etwork
OXC	**O**ptical cross **C**onnect
PaaS	**P**latform **a**s **a** **S**ervice
PCC	**P**ath **C**omputation **C**lient
PCE	**P**ath **C**omputation **E**lement
PCEP	**P**ath **C**omputation **E**lement Communication **P**rotocol
PE	**P**rocessing **E**lement
PSC	**P**acket **S**witched **C**apable
PSTN	**P**ublic **S**witched **T**elephone **N**etwork
PUE	**P**ower **U**sage **E**ffectiveness
PXE	**P**reboot **E**xecution **E**nvironment
P2P	**P**eer-to-**P**eer
QoS	**Q**uality **o**f **S**ervice
RCE	**R**esource **C**omputation **E**lement
ROI	**R**eturn **O**n **I**nvestment
RPMA	**R**esource **P**ower **MA**nager
RPME	**R**esource **P**ower **ME**ter
RRA	**R**esource **R**elocation **A**lgorithms

RS	**R**andom **S**election
RSVP	**R**esource re**S**er**V**ation **P**rotocol
SaaS	**S**ervice **a**s **a** **S**ervice
SCV	**S**mart **C**onnected **V**ehicle
SDH	**S**ynchronous **D**igital **H**ierarchy
SDK	**S**oftware **D**evelopment **K**it
SG	**S**mart **G**rid
SLA	**S**ervice **L**evel **A**greement
SNR	**S**ignal to **N**oise **R**atio
SONET	**S**ynchronous **O**ptical **NET**working
SP	**S**hortest **P**ath
SPF	**S**hortest **P**ath **F**irst
TCO	**T**otal **C**ost of **O**wnership
TDM	**T**ime **D**ivision **M**ultiplexing
TDMC	**T**ime **D**ivision **M**ultiplexing **C**apable
TE	**T**raffic **E**ngineering
TED	**T**raffic **E**ngineering **D**atabase
THR	Static **THR**eshold
TLV	**T**ype-**L**ength-**V**alue
UC	**U**biquitous **C**omputing
UPS	**U**ninterruptible **P**ower **S**upply
VI	**V**irtual **I**nfrastructure
VIDS	**V**irtual **I**ntrusion **D**etection **S**ystem
VIP	**V**irtual **I**nfrastructure **P**rovider
VLAN	**V**irtual **L**ocal **A**rea **N**etwork
VLSR	**V**irtual **L**abel **S**witch **R**outer
VM	**V**irtual **M**achine
VMM	**V**irtual **M**achine **M**onitor
VNF	**V**irtual **N**etwork **F**unction
VPN	**V**irtual **P**rivate **N**etwork
VRO	**V**irtual **R**esource **O**rchestrator
VSU	**V**irtual **S**ervice **U**ser
VXLAN	**V**irtual e**X**tensible **LAN**

WAN	**W**ide **A**rea **N**etwork
WDM	**W**avelength **D**ivision **M**ultiplexing
WSAN	**W**ireless **S**ensor and **A**ctuator **N**etwork

Chapter 1

An Introduction to Cloud Computing

1.1 Virtualization: Concepts and Taxonomy

Cloud computing is a virtualization technique where services run remotely in a ubiquitous way, providing scalable and virtualized resources. In order to deliver critical services to their customers, many organizations rely on heterogeneous applications and resources to satisfy a growing and multidisciplinary demand. With the objective of providing users with distributed and very intensive applications, virtualization is one of the most promising and efficient techniques to consolidate system's utilization on one side, and to lower power, electricity charges and space costs in data centers on the other [1]. This concept was introduced in the 1960s by IBM Corporation, originally to partition a large mainframe computer into several logical instances: a *Virtual Machine* (VM) was intended as an exact software reproduction of a real machine and all of its subsystems to provide concurrent, interactive access to a mainframe computer. This capability of partitioning allowed multiple processes and applications to run at the same time, thus increasing the efficiency of the environment and decreasing the maintenance overhead. In other words, VM was a fully protected and isolated copy of the underlying system through which users could execute, develop and test applications on the same computer. At the end of 1960s, general-purpose computing was the domain of large, expensive mainframe hardware, therefore virtualization was used to reduce the

hardware acquisition cost by letting more users work on it simultaneously. Thus, for a brief period, this technology flourished both in industry and in academic research. As hardware got cheaper and modern multitasking *Operating Systems* (OSs) emerged, the 1970s and 1980s eroded the value of VMs. By the late 1980s, neither academics nor industry practitioners viewed VMs as much more than a historical curiosity. In 1990s, with the emergence of wide varieties of PC-based hardware and operating systems, in addition to the advent of multi-core computing and ever increasing power of servers, the virtualization ideas were in demand again. Cheaper hardware had led to a number of devices often underused, and the increased functionality of operating systems had made them vulnerable; moreover, there was a need to run different applications that were targeted for a specific hardware or operating system on a single machine. To reduce the effects of system crashes, one application running per machine was preferable; therefore, moving applications into virtual machines and consolidating them onto just a few physical platforms increased efficiency and reduced management costs. The main use for VMs then was to execute heterogeneous applications, originally targeted for different hardware and operating systems, on a given computer. In the virtual world, the physical computer is referred to as the *host* and the virtual machines residing on it are called *guests*. The following definition of virtualization is chosen: *"Virtualization is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing, and many others"* [2]. In 1974, Gerald J. Popek and Robert P. Goldberg were the first ones to identify the hardware requirements to build a virtual machine [3], which is an efficient, isolated duplicate of a real machine; they provided the first technical definition of VM through the introduction of the *Virtual Machine Monitor* (VMM). As a piece of software, a VMM has three essential characteristics: first, the VMM provides an execution environment that is essentially identical to the real machine; second, programs running in this environment show at worst only minor decreases in speed; and last, the VMM is in complete control of the hardware resources: the VM applications and the operating system should not access them in a privileged way. There are two types of VMMs known today:

- A VMM that runs directly on the hardware without the need of a hosting operating system. It is known as *hypervisor*. VMware vSphere, Microsoft Hyper-v and Citrix

Xen server are examples of virtualization software.

- A VMM that runs on top of an operating system. Examples are JavaVM and .NET environments. In this case, VMM monitors virtual machines and redirects their requests to appropriate *Application Programming Interfaces* (APIs) on the host operating system.

Virtualization is also commonly defined as a technology that introduces a software abstraction layer between the hardware and the operating system with applications running on top of it [4]. This abstraction layer is called VMM or hypervisor, which provides a compatible, uniform view of the underlying hardware, and it hides the physical resources of the computing system from the operating system, making machines from different vendors with different I/O subsystems look the same. This means that it is possible to run multiple virtual machines with different operating systems in parallel on the same hardware. During the years it has been proven that, by committing hardware and IT equipment to dedicate applications, the resulting utilization of resources is under a cost-effective threshold. There are different types of virtualization:

- **System level virtualization**: it is the technique that enables the emulation of a real physical system with all its devices, such as CPU, memory, disk, network interfaces and so on; the virtual machine runs a full operating system.
- **Process level virtualization**: it consists in the virtualization of individual application processes; an application has to be written specifically for the VM. It allows the deployment of applications through the execution of a VM for each process without changing the host operating system.
- **Operating system level virtualization**: it refers to the virtualization of the kernel interface of an operating system. It uses the kernel of host OS to instantiate user spaces dedicated to the management of multiple guest OSs, without the need for a hypervisor; multiple user space domains can run in parallel on the same system and share the same kernel. However, each domain is isolated from the others.
- **Resource virtualization**: it consists in virtualizing system specific resources such as storage volumes and network resources. There are various approaches to

perform it: aggregating individual components into a larger resource pool; Grid computing or computer clusters where multiple computers are combined into a large supercomputer; partitioning a single resource such as disk space into smaller resources of the same type.

1.1.1 Hardware Virtualization

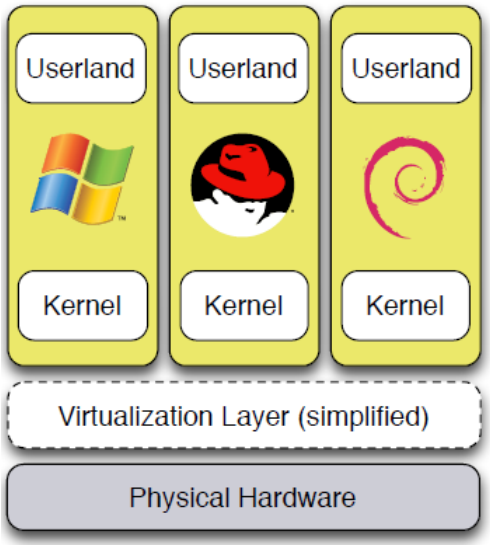


FIGURE 1.1: Hardware virtualization

In hardware virtualization (also known as system level virtualization), the virtual resource represents a full computer and contains the same hardware components of a real computer: CPU, memory, storage devices, network interface controllers (see Figure 1.1). However, these components are all virtual. Operations on virtual components are translated into operations on real components of the physical machine. An instance of such a virtual computer is known as a *hardware virtual machine* or *system virtual machine*. The idea behind this technique is to share the same resources among different users that can create their own VMs. Hardware virtualization offers a number of advantages:

- **Binary compatibility**: existing systems, consisting of applications and their underlying operating systems, can be virtualized without modification of their binary code. This allows legacy systems, which might not work on new hardware, to run in virtual machines with compatible interfaces.
- **Server consolidation**: multiplexing resources by executing multiple virtual machines on the same physical machine can reduce the number of servers used in a

service hosting environment. Often, each service (DNS, DHCP, etc.) is hosted on a dedicated physical machine. Instead, multiple services can be consolidated onto a reduced number of physical nodes, which leads to a reduction of maintenance and energy cost.

- **Isolation**: each virtual machine is isolated from the others, which reduces the amount of damage possible from each operating system. If a virtual machine is compromised by an attacker, his ability to perform attacks on other virtual machines is limited compared to a single multitasking operating system where multiple services share the same environment. However, like in most computer systems, this isolation is not 100% secure: implementation bugs do exist and attackers can exploit them.
- **Migration**: it is possible to capture the state of a virtualized system exploiting the full encapsulation of operating systems in their virtualized execution environment. This state can be transferred on the network to another physical machine to be resumed on the new hardware. Additionally, live migration performs most of the transfer while the virtual machine is executing, minimizing virtual machine downtime.
- **Replication**: it is possible to replicate a virtual machine on another physical node; in case of failure of the primary node, the execution of the virtualized system continues on the secondary node, providing high availability [5].

The major components of the hardware virtualization are:

- **Host OS**: it is the operating system actually running on the hardware and it can access the physical resources through its kernel. It provides running processes with secured and controlled access to physical resources, through a hardware abstraction, also known as *Hardware Abstraction Layer* (HAL), a uniform interface to the underlying physical resources.
- **Guest OS**: it is the operating system running in a virtual environment. It has access to physical resources that are dynamically allocated by the hypervisor or a host OS.
- **Virtual Machine**: it is a software computer that, like a physical computer, runs an operating system and applications.

- **Hypervisor** or **Virtual Machine Monitor**: it provides the software layer that allows the abstraction of physical resources, and it is in charge of mapping virtual operations to physical operations.

Hypervisors can be implemented in two different ways, leading to two different types: type 1 and type 2.

- **Type 1 (native or bare metal)**: native hypervisor is designed to directly control and monitor the hardware resources of the physical node and the guest OS. It runs directly on physical hardware. Several guest OSs can be executed on top of this hypervisor and a privileged one is used to administrate the machine and to control other unprivileged guest OSs (see Figure 1.2). Examples of type 1 hypervisors are Xen [6] and VMware ESXi [7].

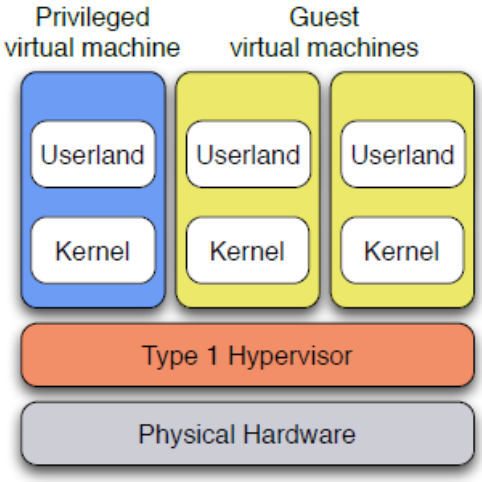


FIGURE 1.2: Type 1 hypervisor

- **Type 2 (hosted)**: this type of hypervisor does not execute directly on bare hardware, but on top of an host OS; hence, it consists in a distinct software layer that is added as a typical application, and it does not replace the OS that is installed on the physical node (see Figure 1.3). A hosted hypervisor interacts with hardware via the system calls of the host OS. The hosted hypervisors show lower performance then the native ones. Examples of type 2 hypervisors are KVM [8], VirtualBox [9], and VMware Workstation [10].

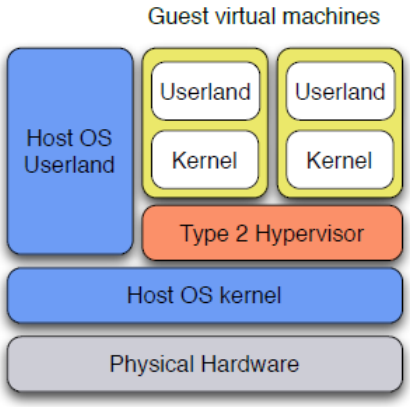


FIGURE 1.3: Type 2 hypervisor

Hardware virtualization can be categorized as:

- **Full virtualization**: the VMs are an accurate reconstruction of a specific physical platform, therefore no modification of the guest OS is required. In 2000s, Intel and AMD introduced extensions to support full virtualization: VT-x and AMD-V respectively.

- **Paravirtualization**: the VMM provides VMs with an optimized version of the hardware physical resources. Therefore the guest OS is modified, making it aware that it is running inside a virtual machine. There is the involvement of the hypervisor, which performs on behalf of the guest OS in the execution of critical system calls. This approach is a way of virtualizing a non-virtualizable architecture and it allows better performances because interactions between guest OSs and hypervisor are easier; moreover, with paravirtualization it is possible to avoid emulating real I/O devices. Xen is an example of hypervisor that uses paravirtualization approach.

- **Hybrid approach - Paravirtualized drivers**: like full virtualization, it allows compatibility with unmodified guest OSs and easy implementation of hypervisor, but guest OSs are extended with paravirtualized drivers, without modifying their core (contrarily to normal paravirtualization). This hybrid approach allows better I/O performance than full virtualization, while maintaining compatibility with proprietary operating systems.

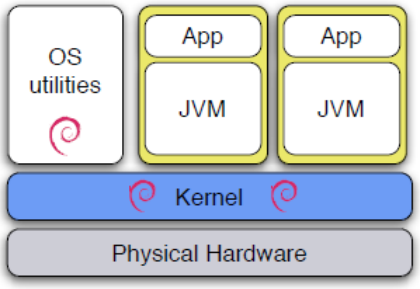


FIGURE 1.4: Process level virtualization

1.1.2 Process Level Virtualization

Process level virtualization consists of platforms that directly execute applications rather than guest OSs. It allows the deployment of applications through the execution of a VM for each process: each application is built specifically to be compatible with a specific process level virtual machine technology. An example of process level virtual machine is the *Java Virtual Machine* (JVM), available on many platforms. Figure 1.4 shows an example of process level virtualization, where two JVMs are executing applications consisting of Java bytecode, on top of Linux kernel.

1.1.3 Operating System Level Virtualization

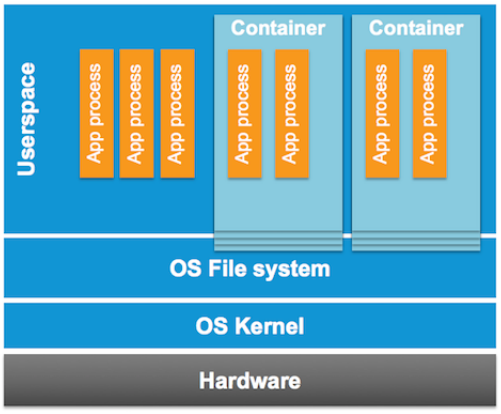


FIGURE 1.5: Operating system level virtualization

Operating system level virtualization concerns the kernel interface virtualization of an OS [11]. Kernel is shared by different user space domains, running in parallel, with each domain independent from the others. The user space domains must be compatible with the underlying kernel. Figure 1.5 shows an example of operating system level virtualization. This approach has some advantages: there is not a slow boot phase,

because when a user space domain starts, the kernel is already running; finally, each virtual machine is not required to have a full kernel loaded, because kernel is shared.

1.1.4 Resource Virtualization

1.1.4.1 Storage Virtualization

It is a process of obtaining a virtual storage from manifold network storage devices, each one acting as a single storage device [12]. The fundamental idea is to group the physical storage systems, which are used by applications to dynamically meet their storage requirements. The objective is to overcome the heterogeneity of storage systems and to provide applications with a single point of access to shared memorization resources, reducing the distance between applications and underlying storage systems by means of an additional abstract administration layer, so that the representation of a datum is decoupled from its physical storage. Storage virtualization allows administrators to perform backup, recovery, archiving very easily and in a short time; storage can be located anywhere and on any type of device, and replicated for dependability purposes. The storage virtualization offers several advantages, such as: easy storage management; optimized and logically centralized view of the storage environment, with easy accessibility of required data; reclamation of storage space; lower number of storage devices and therefore less energy usage; easy migration of data between different storage locations; replication, fail-over and disaster recovery. Amazon, for instance, creates up to three copies in different data centers when storing data. There are also some disadvantages, such as: problems in the interaction between different vendors; network system highly complicated; the attack against a single server might compromise the whole network. There are three types of storage virtualization, implemented according to the infrastructural level [13]:

- **Host-based**: it is usually in the form of a *Logical Volume Manager* (LVM) on a host, which groups more physical volumes into virtual volumes and provides advanced features like snapshot and replication.
- **Network-based**: it is used to virtualize storage through a network connection that will be used by users in order to access it via a specific protocol.

- **Controller-based**: it consists of an intelligent storage array. This type of virtualization can extend all the features of the controller to external memory devices.

1.1.4.2 Network Virtualization

It is a process of combining software and hardware network resources to create one or more virtual networks from a single physical network. Therefore, multiple networks run on the same network device, where each network is independent from the others and from the underlying physical one. Network virtualization involves the splitting of available bandwidth into separate smaller channels, so that network can be shared among different devices. However, even though the bandwidth is shared, the separate channels can be isolated from each other [14]. Technologies that have been used to implement network virtualization are:

- **Virtual Local Area Networks (VLANs)**: they allow a group of computers, which are connected to a LAN switch, to be isolated at the layer 2 of the ISO/OSI stack, by creating different and separated logical groups. VLAN has the following advantages: transparency, because devices can be pooled in a single logical network; security, because vulnerable systems can be hidden in a separate virtual network.

- **Virtual Private Networks (VPNs)**: they are private data networks that connect remote sites and share a common physical network infrastructure. In these networks, data privacy is obtained by using tunnelling protocols and security functions.

- **Asynchronous Transfer Mode (ATM) networks**: they represented for a long time an important aspect of the network virtualization, namely the connection virtualization. The ATM virtual circuits are paths that provide isolation of shared connecting resources, through the reservation of ATM nodes along specific routes.

- **Tunnelling**: a virtual logical topology, which is created above the physical one, is referred to as *Overlay Network*. The nodes of an overlay network are connected through virtual links that correspond to paths in the underlying network. This type of logical networks is usually implemented at the application level: overlay networks have no geographical restrictions, and are flexible and adaptable to

changes. Several organizations are adopting network virtualization based technologies in order to meld the data-link and *Internet Protocol* (IP) approaches. The motivation is to be found in the typical data link technologies scaling issues: the approach of creating virtual L2 networks across IP networks is often referred to as "L2 over L3 tunnelling", and it represents an abstraction layer between the physical and virtual networks. This technique tries to combine the benefits coming from the two layers: the scalability, the fast convergence times and the bandwidth allocation of L3 technologies, and the simplicity in the configuration of L2 networks. L2 over L3 is obtained through encapsulation techniques, such as *General Routing Encapsulation* (GRE), *Virtual eXtensible LAN* (VXLAN), or by using OpenFlow.

Network virtualization consists of three different roles [15]:

- **Infrastructure provider**: it owns physical resources and splits them into several isolated virtual resources, composed of virtualized nodes and links, which are supplied to the virtual network provider.
- **Virtual network provider**: it leases virtualized infrastructures from one or more infrastructure providers, and combines them into complete virtual networks, composed by interconnected virtual nodes and links. Afterwards, virtual networks are sold to a virtual network operator.
- **Virtual network operator**: it uses the virtual network in order to offer and operate services, which are implemented by a set of functions and protocols.

1.1.5 Advantages and Challenges

Virtualization is a growing field, and in high demand. The market for virtualization has increased significantly over the past several years. While it does reduce costs and space, several issues come along with this sudden growth. In this section, we will look at details of some major advantages and challenges facing widespread virtualization.

1.1.5.1 Virtualization Advantages

For a provider of IT services, the use of virtualization techniques has a number of advantages:

- **Flexibility**: manifold virtual instances can run on a single physical machine and it is possible to change their characteristics (RAM, hard disk) while they are running [3].

- **Availability**: it is possible to keep the virtualized instances running even though the physical node has to be shut down, i.e. for hardware upgrade or maintenance. In this case, virtual machines can be temporarily migrated to another physical machine. This ensures better availability of the services and makes it easier to comply with *Service Level Agreements* (SLAs).

- **Scalability**: in case the capacity demand increases, a new physical node can be added to the cluster. It is very easy to add or remove nodes.

- **Hardware utilization**: physical servers usually run with low utilization because their operators employ oversized computing resources to cover peak usage; if virtual machines are used, any load requirement can be satisfied from the resource pool and VMs utilize hardware resources left idle by host OS [16].

- **Security**: by using multiple virtual machines, services are isolated. Indeed, each service could run on a separate VM and if a service is compromised, the other ones are unaffected.

- **Cost**: different virtual machines can be consolidated to run on a smaller number of physical components. With consolidation, the same performance can be obtained through a smaller footprint, and the costly expansion of an existing data center might possibly be avoided.

- **Adaptability to workload variations**: it is possible to automate resource pool management; VMs can be created and configured automatically as required.

- **Energy consumption**: the cost of energy required to make a server work is very high. Consolidation reduces the number of physical components; this, in turn, reduces the expenses for energy supply. Reducing power usage has the advantage of both decreasing utility bills and increasing an organization's compliance with green initiatives. With the continued emphasis on slashing IT budgets generated from the current economy, lowering utility bills is attractive.

1.1.5.2 Virtualization Challenges

Some virtualization challenges are listed below:

- **Limited Overhead**: it depends on the flexibility and involves decreased performance.
- **Single point of failure**: if organizations place all of their virtual resources onto a single host server, it is important to plan virtual resource deployment so that the failure of a single host server will not have catastrophic consequences.
- **Isolation**: full and complete isolation between virtualized resources on the same hardware remains a very difficult problem, particularly in situations that require sharing of information.

1.2 Evolution of Computing

Computing has evolved over time according to three distinct paradigms [17] [18]. The mainframe computing was the first computing paradigm, popular during the 1960s up to the 1970s. Processing power was condensed in a centralized system, a mainframe computer with all functionalities and resources, accessed through simple terminals. The second computing paradigm was the stand-alone computing, which consisted of powerful PCs and workstations that executed a wide range of applications, connected together through a *Local Area Network* (LAN) technology, according to a client-server configuration. This reduced the time to access to a computational resource and it gave the chance to work independently in a parallel way. Finally, network-centric computing was developed in the 1990s and it enabled machines to access both to applications and data on networked servers [19]. Dominant form of a network-centric computing is the Internet computing, which has changed the way applications are developed and distributed, going from a client-server to a web-based browser-server configuration without software installation in client machines. This paradigm has improved the level of transparency of the information between multiple groups within a company, and it has allowed the exchange of data between companies.

1.2.1 The Path to Cloud Computing

In this section, the major architectures that have been used to execute parallel and distributed applications, contributing to the creation of Cloud computing are investigated: clusters and grids.

1.2.1.1 Cluster Computing

Cluster computing can be defined as *"a type of parallel or distributed processing system which consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource"* [20]. Physical nodes are connected through a LAN in the execution of compute-intensive and data-intensive tasks that would be not feasible to execute on a single computer. This network can be a typical Gigabit Ethernet network, or a high speed interconnect such as InfiniBand or Myrinet. Clusters are controlled by resource manager software, which assigns jobs to compute nodes, tracks their status and shares the infrastructure between cluster users. Resource managers contain queues to store jobs waiting for execution, and schedule jobs on available resources according to constraints set by users and/or administrators. In the 1990s, the Cluster-based solutions became very popular and they were massively used mainly for high availability, load-balancing and performance purposes. A possible classification is the following:

- **High Availability Cluster (HAC)**: they are also called fail-over clusters and implement the concept of redundancy [21]. High availability is achieved by having multiple secondary servers that are exact replicas of a primary server, in order to avoid a single point of failure. This solution guarantees fault tolerance that is the ability of a system to operate gracefully even in the presence of any fault: the fault in one component only affects the cluster's power but not its availability. In an active-passive configuration, only a single node is in active state, while the remaining ones are in stand-by mode, ready to take over processing in case active node fails. This solution ensures that the performance for the fail-safe workload is the same before and after fail-over. In an active-active configuration, all nodes are active and share the application processing tasks. If one node fails, the remaining active nodes take over it. The active/active configuration is more cost-effective

than the active/passive configuration, because of a better load distribution between the nodes of the infrastructure, but it provides slower fail-over and possibly reduced performance when a node fails.

- **High Throughput Cluster/High Performance Cluster (HTC/HPC)**: they have been used for a long time for scientific research and commercial activities; they are intended for testing and running large codes, parallel-processing codes, visualization and scientific applications. HPC systems let users execute a single instance of parallel software over many processors, while HTC systems ensure the execution of multiple independent software instances on multiple processors at the same time.

- **Load Balancing Cluster**: they provide better performance by distributing workload among nodes in a cluster. Considering web servers, different queries are handed to different nodes for processing, resulting in a faster overall response time. Load balancing is accomplished by manifold techniques: a round-robin algorithm or more complex algorithms that rely on feedback from the individual machines.

1.2.1.2 Grid Computing

Ian Foster is considered as one of the earlier proponents of Grid technology. He defined a Grid as *"a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high end computational capabilities"* [22]. Later, Foster redefined the Grid as *"a computing environment concerned with the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations"*. He used an analogy to the electricity grid, where users could plug in and use a (metered) utility service: if companies cannot generate their own power, they would purchase that service from a third party capable of providing a steady electricity supply. The same should apply to computing resources: if a node could plug itself into a grid of computers and pay only for the used resources, it would be a more cost-effective solution for companies than buying and managing their own infrastructure. From the technical point of view, Grid computing consists of virtualization and sharing of computing and data resources among different physical domains. Grid computing has a distributed nature, therefore computational nodes could be anywhere in the world: resources are abstracted from the physical location and automatically allocated according

to user's computing needs, without interest in data location. The ensemble of resources is able to work together cohesively because of defined protocols that control connectivity, coordination, resource allocation and security [23]. Grid environment allows service oriented and flexible sharing of heterogeneous resources for compute intensive and data intensive tasks, providing more computational capabilities and increasing efficiency and scalability of the infrastructure. During the years, Grid has rapidly involved multiple applicative domains, ranging from advanced networking to artificial intelligence. The reason behind the constitution of a Grid-based infrastructure is the need for enterprises to solve complex scientific problems, through coordinated resource sharing and collaborative problem-solving [24]. Another benefit of Grid computing is a more robust and resilient infrastructure through decentralization, fail-over and fault tolerance. Ian Foster proposed a simple check-list, according to which a grid system:

- *Coordinates resources that are not subject to centralized control.* A Grid integrates computing resources that belong to different control domains (for example different administrative units of the same company or different companies). Technologically, this requirement addresses the issues of security, policy, payment and membership.
- *Uses standard, open, general-purpose protocols and interfaces.* General-purpose protocols and a common standard for authentication, authorization, resource discovery and resource access are needed, so that the system is able to execute generic applications.
- *Delivers non-trivial quality of service.* It is important to support various *Quality of Service* (QoS) parameters such as response time, throughput, availability, security and co-allocation of multiple resource types to meet complex user demands.

One of the de facto standards for Grid computing implementation is the Globus Toolkit [25], which defines protocols and the middleware layer to guarantee a controlled access to the resources shared in a *Virtual Organization*: this is the combination of different institutions, research centers and/or individuals that often have a unique goal and define a common set of sharing rules. The tasks addressed by Globus involve resources discovery, provisioning, management, security and jobs scheduling. Broadly speaking, there are four types of Grid:

- **Computational Grid**: it is a set of resources combined to aggregate computational capacity. This type uses networks of computers as a single, unified computing resource, and it is suitable for high-throughput computing applications.
- **Scavenging Grid**: the politics of "scavenging" is applied; every time a machine remains idle, it reports its state to the grid node responsible for the management of the resources. This node usually assigns to the idle machine the next pending task that can be executed in that machine.
- **Data Grid**: it is responsible for housing and providing access to data across multiple organizations through a unified interface, and makes them available for sharing and collaboration purposes. In data grids, the focus is on the management of data that are being held in a variety of data storage facilities in geographically dispersed locations.
- **Service Grid**: it is a set of distributed resources, each providing a specific function that needs to be aggregated in order to collectively perform the desired services.

The architecture of a Grid system is often described in terms of layers, each providing a specific function (see Figure 1.6):

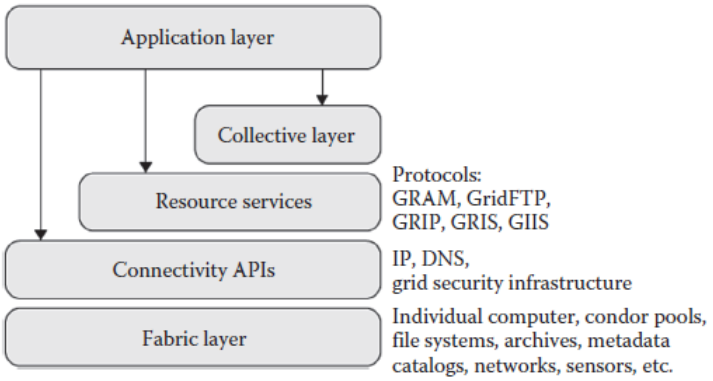


FIGURE 1.6: Layered Grid architecture

- **Fabric layer**: it provides resources that are part of the Grid, such as computational resources, storage systems, electronic data catalogues and networking devices.
- **Connectivity layer**: it consists in the core communication and authentication protocols. For example, the *Grid Security Infrastructure* (GSI) [26] is a public

key–based protocol and it provides every grid transaction with a desired security level.

- **Resource layer**: it includes APIs and *Software Development Kits* (SDKs) for publication, discovery, negotiation, monitoring, control, accounting, and payment concerning operations on shared resources. The *Grid Resource Access and Management* (GRAM) [22] is the protocol that assures the scheduling of computing resources to users and it is also used for monitoring and control tasks.
- **Collective layer**: it implements a variety of sharing behaviours with directory services, brokering services, programming systems, community accounting, authorization services and collaborative services.
- **Application layer**: it is characterized by any user application that is deployed on a Grid.

Grids are characterized by a geographical distribution of devices and a great amount of available data: this enables the realization of complex experiments that would result in very high costs for the single organization. Grid and Cloud computing paradigms basically share the same underlying concept, which is the *Utility Computing*, namely a service provisioning model through which computing resources are used by a customer as needed. This has the objective of maximizing the utilization and bringing down the relative costs. Anyway the two approaches are quite different in terms of purposes and distribution of resources.

1.2.2 Cloud Computing Definition

Cloud computing describes a computing infrastructure depicted as a "cloud", from which users can access applications on demand [27]. The location of this infrastructure is shifted to the network to reduce the costs and this model offers computing, storage and software "as a service" [28]. A plenty of different definitions of this paradigm exists. According to Wang: *"Cloud computing is a set of network enabled services, providing scalable, Quality of Service (QoS) guaranteed, normally personalized, inexpensive computing platforms on demand, which could be accessed in a simple and pervasive way"* [29]. Markus Klems pointed out that *"immediate scalability and resources usage optimization are key elements for the Cloud. These are provided by increased monitoring,*

and automation of resources management in a dynamic environment" [30]. He also defined Cloud computing from an economic point of view: *"Building on compute and storage virtualization technologies, and leveraging the modern Web, Cloud computing provides scalable and affordable compute utilities as on-demand services with variable pricing schemes, enabling a new consumer mass market"* [31]. Carl Hewitt claimed that *"Cloud computing is a paradigm in which information is permanently stored in servers on the internet and cached temporarily on clients that include desktops, entertainment centers, table computers, notebooks, wall computers, handhelds, sensors, monitors, etc."* [32]. According to Armbrust: *"Cloud computing, the long-held dream of computing as a utility has the potential to transform a large part of the IT industry, making software even more attractive as a service"* [33]. Tikotekar emphasized the presence of a pay-per-use economic model: *"Cloud is a pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the infrastructure provider by means of customized SLAs"* [34]. Buyya et al. also added that *"to reach commercial mainstream it is necessary to strengthen the role of SLAs between the service providers and the consumers of that service"* [35]. McFedries introduced the concept of data center as *"the basic unit of the Cloud offering huge amounts of computing power and storage by using spare resources"*. National Institute of Standards and Technology (NIST) defined the Cloud computing as *"a model for enabling convenient, on demand network access to a shared pool of configurable computing resources such as networks, servers storage, applications and services that can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models"* [36]. A comprehensive definition of Cloud is proposed in [37]: *"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the infrastructure provider by means of customized SLAs. The set of features that most closely resemble this minimum definition would be scalability, pay-per-use utility model and virtualization"*.

1.2.3 Cluster, Grid and Cloud Computing: a Comparison

Cluster computing is just cooperation between computers in order to solve a task or a problem. Grid computing is similar to Cluster computing: it combines resources from different organizations to reach a common goal. The major difference is that a cluster is homogeneous while grids are heterogeneous. Cluster computers have the same hardware and operating system, therefore they act as a single machine, while Grid enables the sharing and aggregation of geographically distributed autonomous resources, which could have different operating systems and hardware. Moreover, grids are distributed over a LAN or WAN, while clusters are normally gathered in a single location. Another difference lies in the way resources are handled. In case of Cluster, resources are managed by a centralized resource manager, while in a Grid every node is autonomous and acts like an independent entity. As introduced in Section 1.2.1.2, Cloud and Grid computing are similar for managing sets of distributed computing resources and implementing the more general model of Utility Computing, offering a common environment for distributed resources, but there are some differences. Grid has the objective of using large amounts of resources to solve very complex and computational expensive problems in science and industry, while Cloud provides a platform to develop, test and deploy Web-scale applications and services [31]; by using virtualization, Cloud provides an infrastructure useful to deploy services for users, offering customized, scalable and QoS guaranteed computing environments. Grid has heterogeneous resources geographically distributed, while Cloud provides service providers with resources they require, giving the impression of a single dedicated resource. All resources constituting a Grid computing infrastructure are predefined and predetermined, while Cloud computing releases or augments dedicated resources dynamically, depending on the demand. Therefore, in the Grid paradigm the focus is on the processes to share and schedule the utilization of resources for different users; Cloud is fundamentally distinguished by the establishment of a pay-per-use relation between a customer and a provider, oriented towards business rather than academic resource management [38]. Table 1.1 compares different features of Cluster, Grid and Cloud highlighting the similarities and differences between paradigms.

TABLE 1.1: Comparison between Cluster, Grid and Cloud computing

	Cluster	**Grid**	**Cloud**
SLA	Limited	Yes	Yes
Allocation	Centralized	Decentralized	Both
Resource handling	Centralized	Distributed	Both
Loose coupling between nodes	No	Yes	Yes
Reliability	No	Half	Full
Security	Yes	Security through credential delegations	Security through isolation
User friendliness	No	Hard to manage	Yes
Virtualization	Half	Virtualization of data and computing resources	Virtualization of HW & SW
Interoperability	Yes	Yes	Half
Standardized	Yes	Yes	No
SOA	No	Yes	Yes
Multi-tenancy	No	Yes	Yes
Self-service	No	Yes	Yes
Computation service	Computing	Max Computing	On demand
Heterogeneity of resources	No	Yes	Yes
Scalable	No	Nodes and sites scalability	Nodes, sites, and hardware scalability
Inexpensive	No	No	Yes
High level services	No	Yes	No

1.2.4 The Future of Cloud Computing: Fog Computing

Cisco recently proposed a new computing paradigm called *Fog Computing* [39], which extends the Cloud computing paradigm to the edge and runs generic application logic on resources throughout the network. It is defined as *"a highly virtualized platform that provides compute, storage, and networking services between end devices (typically but not exclusively located at the edge of network) and traditional Cloud Computing data centers"*. In [39], Fog computing consists of Fog servers near to wireless sensors and mobile devices, in order to lighten their computational load and data processing. In the Fog computing concept, users are served by computing nodes close to the network edge (e.g., road side units in vehicular networks) to reduce latency and communication overhead, with periodic updates from the remote Cloud [40]. Cloud becomes a problem for latency-sensitive applications, because they require nodes in their proximity to satisfy delay requirements; indeed, Cloud imposes a high communication latency because of sending events from a user through the core network to the data centers [41]. In contrast to the Cloud, introducing intelligence in the network allows Fog computing resources to perform processing near the edge, reducing end-to-end latency; on the contrary, latency–tolerant and large–scope aggregation can still be efficiently performed in the core of the network [42]. Fog is the appropriate platform for various critical *Internet of Things* (IoT) applications and services, including connected vehicles and smart cities, but also for *Ubiquitous Computing* (UC) approaches, by extending Cloud computing services. The extension is motivated by [43]:

- Edge location, location awareness and low latency: Fog nodes provide services at the edge of network, including latency-sensitive applications.
- Geographical distribution: Fog requires geographical distribution of resources in contrast to the more centralized Cloud.
- Large-scale sensor networks, which communicate various data about the environment usually by wireless access.
- Very large number of nodes, because of geographical distribution.
- Support for mobility.
- Support real-time communications with mobile devices.

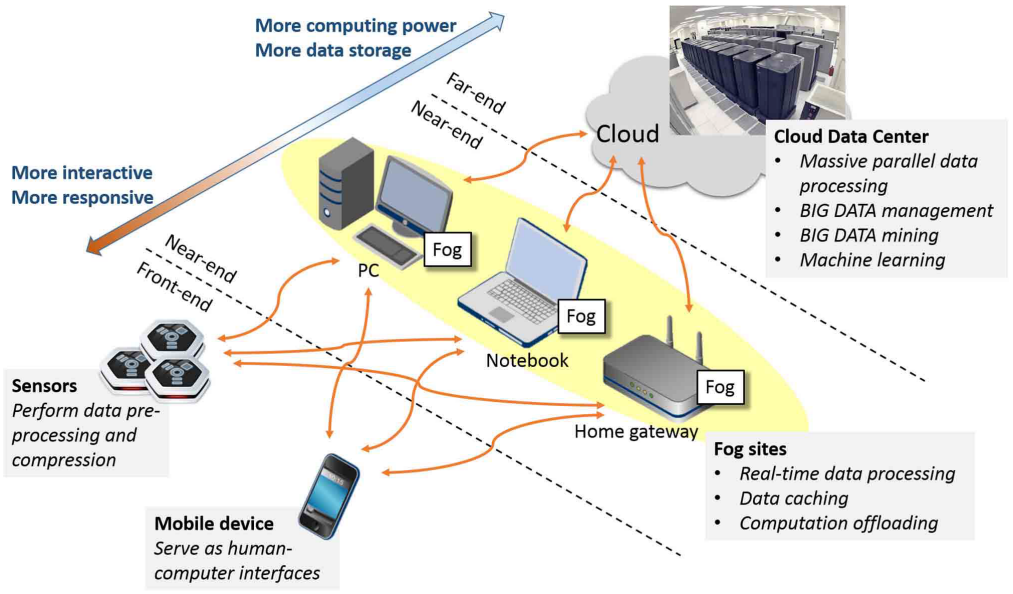


FIGURE 1.7: Conceptual architecture of Fog and Cloud infrastructure

- Support heterogeneous devices and interoperability with different providers.
- Requirement of on-line analytic and interplay with the Cloud.

Fog computing plays an important role in at least three scenarios: *Smart Connected Vehicles* (SCVs), *Smart Grids* (SGs), and *Wireless Sensor and Actuator Networks* (WSANs). In particular, the characteristics of Fog computing make it the best platform to support energy-constrained WSANs. To be efficiently useful in the above fields, the Fog computing platform follows a multi-tier architecture. Figure 1.7 illustrates a three-tier architecture, showing the interplay and complementary roles of Fog and Cloud.

The first tier (front-end) includes wireless sensors and mobile devices; the second tier (near-end) includes IT products (e.g., PC, TV, set-top boxes, etc.) acting as proxies between front-end devices and far-end servers. They are called Fog servers, which offer computing capacity and storage capability to front-end devices, enhancing their battery-life and performance, and reducing latency. The third tier (far-end) includes Cloud servers located in data centers, which are used as repository for data that have a long permanence; moreover, they are useful to make deductions and predictions, beyond the capacity of Fog servers. A federation of both Clouds and Fogs can support highly heterogeneous systems, where network-intensive operators are placed on distributed fog

nodes and computational-intensive operators in the Cloud. However, developing applications using Fog computing resources is not trivial because it involves orchestrating dynamic and heterogeneous resources at different levels of network hierarchy.

1.3 Anatomy of Cloud Computing

The Cloud computing enables the ubiquitous and on demand network access to a shared set of configurable computing resources, which can be quickly acquired and released with minimal effort and interaction with the service provider. Cloud computing reduces the cost related to the delivery of services, while speeding up service deployment. It supports several technologies such as server, storage and virtualization that brings together virtual applications quickly. Cloud storage can be used to store and hold applications; moreover, it is often combined with other Cloud services such as Cloud database, Cloud data and Cloud security. Cloud computing has many advantages: enterprises do not need buying a large number of hardware devices and they achieve economies of scale. There are also critical problems such as security and performance in a public Cloud. On the contrary, a private Cloud has some advantages compared to public Clouds: it is a platform implemented within the corporate firewall and it provides more control over the company's data, ensuring security and quality of service. Private Cloud has weaknesses too: it is difficult to ensure high performances and to provide flexible services. Hybrid Cloud is a model combining both public Cloud and private Cloud models: it is designed to quickly scale company's needs and to handle peak-loads, extending private Cloud with the resources of a public Cloud. Real-time scalable resources are accessible from a Web browser by customers, which pay only for the used computing resources without knowledge of the underlying technological infrastructure. Other advantages are: costs of purchasing hardware and software are reduced; minimal time to provide resources; simplicity of operation; novel and complex computing architectures; mechanisms for disaster recovery. This results in reduced capital expenditure and reduced operational costs. Therefore, Cloud technology is a great innovation opportunity for companies and also for the research field.

1.3.1 Cloud Computing Characteristics

The essential characteristics of Cloud environments are [44]:

- **On demand and Self-service**: users can automatically consume computing capabilities according to their needs at any moment; moreover, they can customize and personalize their computing environments.
- **Multi-tenancy and Resource pooling**: multi-tenancy is a *"characteristic of a software program that enables an instance of the program to serve different consumers (tenants) whereby each is isolated from the other"* [45]. The resources are accessed independently from the location, so that the customer does not know their physical or geographical location. Although resources are shared among multiple users, customer gets to feel that he is the sole proprietor through his Cloud control panel. Multi-tenancy model allows resources being dynamically assigned on the basis of customer's demands and needs, giving rise to another concept called resource pooling that allows for *"Cloud providers to pool large-scale IT resources to serve multiple Cloud consumers. Different physical and virtual IT resources are dynamically assigned and reassigned according to Cloud consumer demand, typically followed by execution through statistical multiplexing"* [45]. Cloud providers manage heterogeneous infrastructure resources, such as hardware, software, processing servers and network bandwidth.
- **Rapid elasticity and Scalability**: resources can be rapidly, elastically and automatically scaled, as demand rises or drops. The user is provided with the illusion of being able to access unlimited resources. Resources can be scaled according to different software configurations, hardware performance and geographical locations.
- **Measured service**: the resources requested by a customer should be measurable according to a set of quantitative parameters in order to automatically control and optimize their use through continuous measurement of performance indicators. The measures should be available to the users via API for transparency purposes and for allowing rapid modification of QoS requests. Besides, these measures are useful to the provider in order to realize accounting and billing if there is an economic deal.

- **Broad network access**: capabilities are available through a broadband network and accessed with standard mechanisms, which promote their use by heterogeneous platforms (mobile phones, tablets, laptops, workstations, etc.).

1.3.2 Cloud Delivery and Deployment Models

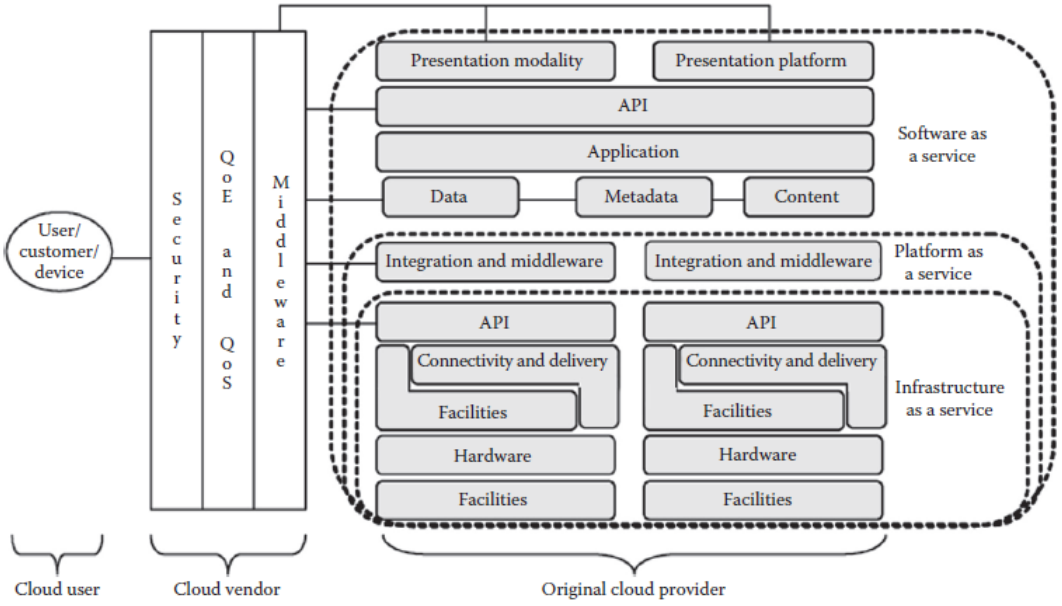


FIGURE 1.8: Cloud delivery model

Cloud infrastructures can be classified concerning the service and the deployment model they implement. From the service model perspective, new XaaS services are gradually taking the place of resources traditionally used [46]. These services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) (see Figure 1.8), made available to consumers as pay-per-use subscription services [38]. This basic classification has been extended and it has assumed very different granularities; hence, if any service X is offered by means of an universal access, in a scalable and elastic way and in a pay-as-you-go basis, it is possible to use the term *XaaS*. In detail:

- **Infrastructure as a Service (IaaS)**: according to NIST, IaaS is defined as *"the capability provided to the consumer to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The*

consumer does not manage or control the underlying Cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls)". This layer is also known as *Resource Cloud*, since it represents the lowest level of abstraction. The aim of the IaaS paradigm is to provide IT resources as services that are delivered through the network, by hiding in such a way the sophistication of the underlying infrastructure. Cloud users can manage virtual resources such as: CPU, memory, storage, operating system. In this scenario, the main advantages of IaaS are the elasticity and the pay as-you-go model: users can scale their infrastructure according to their needs, dynamically resize the infrastructure depending on the load, and pay only for resources that are actually used. This is different from traditional systems where users have to size their infrastructure according to the peak load, leading to wasted resources and increased expenses. IaaS allows users to have a high degree of control over their infrastructure. They manage virtualized operating systems with administrative rights and can customize execution environments as required, which eases the migration of existing applications and systems to IaaS Clouds. Well-known IaaS platforms include Amazon EC2 [47], Rackspace [48], and Science Clouds [49].

- **Platform as a Service (PaaS)**: NIST defines PaaS as *"the capability provided to the consumer to deploy onto the Cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, or storage but has control over the deployed applications and possibly application hosting environment configurations"*. PaaS solutions provide a development platform and APIs to build and run applications, by using programming languages, libraries, services and tools supported by the provider. The platform is in charge of allocating resources to applications and scaling them according to user demand. The consumer does not handle the underlying Cloud infrastructure but he has control over installed applications and configuration settings of the host environment. Windows Azure [50], Google App Engine [51], and Hadoop are some well-known PaaS platforms.

- **Software as a Service (SaaS)**: SaaS has been defined by NIST as *"the capability provided to the consumer to use the provider's applications running on a*

Cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a Web browser (e.g., web-based e-mail). The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings". Software or application is provided to customers as a service, without the need to install applications on customer's PC; this reduces costs and maintenance. SaaS solutions integrate hardware, development platforms and applications, providing a self-contained operating environment that is used to deliver the entire user experience, including the content, its presentation and the business logic. Cloud-based services are accessible regardless of the location and the type of used devices. Prominent SaaS applications include Salesforce.com for CRM, Google Docs for document sharing, and Web e-mail systems like Gmail, Hotmail, and Yahoo! Mail.

Cloud services also differ in how their deployments are made available to users. According to this criteria, Clouds are classified in four types: private, public, hybrid and community.

- **Private Cloud**: it is a Cloud infrastructure managed and used exclusively by a single organization (e.g. a company or a laboratory). Before the emergence of Cloud computing, many organizations already used virtualization in order to manage their internal computing infrastructures. There is generally no billing involved in these systems. Private Cloud is ideal for companies and organizations that must comply with a series of regulations and desire to efficiently manage their resources.

- **Public Cloud**: it is available to the general public, whether they are individuals or organizations. The main requirement to access a public Cloud is to provide a payment method (e.g. a credit card number) to be billed for resource usage. This is the model followed by commercial Cloud providers like Amazon EC2 and Rackspace.

- **Hybrid Cloud**: Cloud infrastructure is a composition of two or more deployment models. For instance, a company can utilize a public Cloud to maintain service

levels in the face of rapid workload fluctuations, and a private Cloud to manage sensitive data.

- **Community Cloud**: it is restricted to a limited set of users belonging to specific organizations, or sharing a common goal. It may be owned or managed directly by the organization, a third party or their combination. Scientific Clouds are an example. This model shows similarities with grids, in which multiple organizations share resources to reach a common goal.

1.3.3 Actors in Cloud Computing

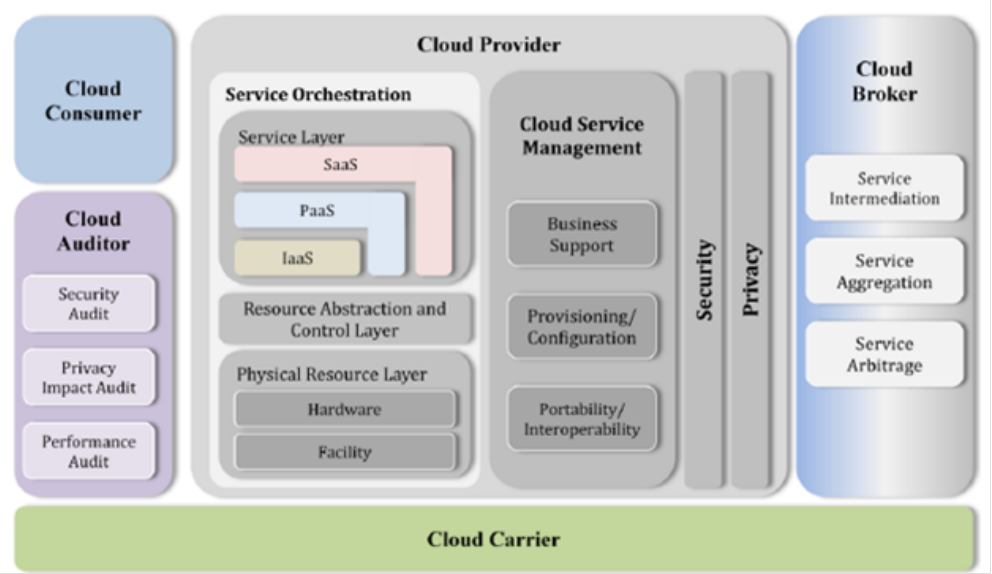


FIGURE 1.9: Cloud Reference Architecture

The NIST Cloud Computing Reference Architecture [52] includes five participants (actors) for the Cloud domain. Figure 1.9 illustrates the actors and their responsibilities. The reference architecture defines the actors as follows:

- **Cloud Consumer**: it is the entity that purchases and uses Cloud services from a Cloud Provider or a Cloud Broker. It can be an individual or organization, which establishes a business relationship with a provider.

- **Cloud Provider**: a person, organization or entity that provides Cloud based services, manages resource allocation and control, and it is responsible for managing physical infrastructure, resource provisioning, monitoring, business related services and migration of services between Clouds. Cloud providers usually have

a service catalogue from which the customer can select one or more services. The relationship between provider and consumer is often sanctioned by SLA contracts. Among the activities under its supervision there are:

  - service deployment;
  - service orchestration;
  - Cloud services management;
  - security and privacy.

- **Cloud Auditor**: a third party that can conduct independent assessment (evaluation and auditing) of Cloud services, in terms of security, privacy, availability and performance of the Cloud implementation, to ensure that vendor operates as expected and that security requirements are met. The focus is on how to monitor the services, and to identify quantitative metrics for different kinds of measured properties.

- **Cloud Broker**: it is an entity that manages selection, usage, performance and delivery of Cloud services; moreover, it negotiates the relationship between consumer and provider because of the management complexity of Cloud services. The broker offers three capabilities:

  - *Service Intermediation*: the broker interfaces with the provider and it is responsible of integrating and combining Cloud services, to enrich the service offered by the provider and add some capabilities, such as strategic sourcing, vendor management, service value management (QoS monitoring, reporting).
  - *Service Aggregation*: the broker may aggregate multiple services offered by the provider and ensures the interoperability and security of data between systems. It provides them to the customer, acting as a reseller.
  - *Service Arbitrage*: it is similar to aggregation, with the difference that services are not fixed. Broker has the flexibility to choose services from multiple agencies.

- **Cloud Carrier**: it provides access to consumers through network and other access devices. Moreover, it provides connectivity and transport of Cloud services from providers to consumers (e.g. an *Internet Service Provider* (ISP)). This role is usually covered by network operators or telecommunication agencies. A provider

usually sets up SLAs with a carrier to provide services consistent with the level of SLAs offered to Cloud consumers.

A possible scenario involves the interaction between a consumer and a provider that may be mediated by a broker. The consumer can request a particular service the broker can actually deliver as a composition of multiple services offered by different providers or by an individual provider. When the consumer decides to rely on a Cloud provider, it establishes a SLA contract. In turn, the provider decides to sign a contract with a carrier to ensure a specific quality of service. The auditor is the entity that has to make sure everything that has been agreed between a consumer and a provider is actually respected. The analysis of Cloud Computing solutions should be conducted by taking into consideration several factors, benefits and risks: an economic analysis can also be useful to compare the cost reduction with the adoption of a Cloud solution against the on premise approach. The trade-off to be investigated lies in the economic benefit and the achievable service level, in terms of reliability and performance, which is not easily measurable and transparent to the consumer.

1.3.4 Cloud Computing Benefits

The common benefits associated with Cloud computing are explained in this section [53].

Cost Efficiency

Saving money is the leading motivation for Cloud computing. It allows organizations to reduce the costs of computer hardware and software, decreases the amount of personnel necessary to manage data, and permits to save space and energy used to run computer equipment. Moreover, Cloud computing diminishes the capital investment for project start-ups. Sharing the processing resources among multiple users cuts down infrastructure costs. IT departments, thanks to Cloud computing, save money for application implementations, maintenance and security, taking advantage at the same time from economies of scale. They can increase hardware utilization percentage and reach massive capacities immediately, avoiding the investments in new infrastructures, the license of new software or the training of additional personnel. Indeed, the Cloud provider can

give to the companies support and expertise. In addition, Cloud also allows companies to realize new opportunities to create enhanced services, saving time and money. Organizations only pay for the actual utilization (i.e. pay-as-you-go). The pay-as-you-go model provides several benefits:

- Organizations reduce IT *Capital Expenditure* (CapEx) investments because these costs are transferred to providers that spread them among their client base. This is particularly useful because IT projects have a ramp up period that might last one to six months, in which usage is low. Indeed, when companies spend capital on equipment, they will consider the same CapEx for each month of the project, regardless of the actual use. If companies invested less CapEx on equipment and software, and moved that investment to the Cloud *Operating Expenditure* (OpEx), they could have a better alignment of investments and costs in relation to the real usage.
- Enhancement of the *Return-On-Investment* (ROI).
- IT maintenance, upgrade and support costs are transferred from Cloud consumers to Cloud providers.

Scalability

The Cloud has an elastic nature that enables organizations to deploy solutions quickly; resources are added on-demand and released if not necessary, responding to unexpected growth in demand. Cloud computing solutions are so efficient in scaling up and down that they permit to reach a competitive advantage by paying only for the IT resources actually used.

Reduced Effort for IT Resource Management

Data and applications can be shared in the Cloud by each organization and its partners. Companies can better handle the application life cycle when resources are greater and at lower costs. Cloud computing also allows companies to gain market shares, to improve services for customers and to focus on their core business.

Flexibility

Cloud computing ensures much more flexibility than traditional IT methods. Cloud services are based on flexible infrastructures that could change in relation to demand fluctuations; therefore, resources can be quickly provisioned, de-provisioned, and/or relocated to be employed in other parts of the system. This fast and elastic reply to business requests gives organizations a competitive advantage in the marketplace. Moreover, Cloud flexibility allows companies to fully-outsource all aspects of their infrastructure, or outsource hardware while maintaining control of their IT infrastructure.

Efficiency

Another fundamental feature of Cloud computing is efficiency, achieved through scalability, rapid and easy deployments, and only paying for the actual use of resources. In this way, companies can save time and money in deploying their own IT infrastructures. Therefore, organizations can focus their energies on research and development, and transfer the management of operational activities to the Cloud.

Agility

The ability to get applications to market very quickly is an important feature of Cloud computing. It is achieved by using the most appropriate resources and maximizing cost efficiency. Cloud computing improves time to service, removes the gap between business and IT, and increases organizational agility.

Rapid Developments and Deployments

Applications and services can be developed more efficiently thanks to Cloud computing, and the speed of deployment and testing cycles is higher; organizations could complete within a short while activities that otherwise would take days, weeks or even months to be realized.

High Reliability

Organizations could reach a high level of business continuity and resiliency if they deployed applications on high-performance Cloud computing platforms. When companies use their own systems, they cannot obtain the same results without a considerable expense in disaster recovery sites and replication of software. Cloud providers keep redundant copies of client data at multiple Cloud data centers by means of advanced computing platforms. Then, Cloud providers could safeguard consumers in case of local physical disasters, accidental data loss or corruption, and malicious data destruction.

Improved Security

In most situations, Cloud providers are more experienced in security than in-house IT departments thanks to security experts, compliance to standards and certifications. This is a fundamental topic in Cloud environments and it will be detailed in the following section.

Device and Location Independence

Users could access systems wherever they are and no matter what device they are using thanks to the "location and device independence" of the Cloud, therefore increasing end-user performances.

Energy Efficiency

Lots of companies do their best to make environmentally-responsible choices, and they could reduce the use of energy, cooling and floor space thanks to Cloud computing. Moreover, Cloud providers own technologically advanced data centers that are more energy efficient, therefore allowing an effective reduction in power consumption and a lower impact on environment.

1.3.5 Cloud Computing Risks

This section presents a list of Cloud computing risks for Cloud users, as well as regulatory and certification issues for Cloud providers [53].

SLA Management

The SLA defines the services acquired from Cloud providers in measurable terms. A SLA can include service availability, measurements of performance, mean time to respond, penalties if the Cloud provider does not follow the terms of the agreement, etc. When a consumer and a provider set up an economic agreement, they cannot renegotiate SLA parameters. The definition of the deal is not something immediate because sometimes a service is offered by several providers. A technical problem consists in the monitoring methods of SLA parameters like, for example, those concerning security. A major issue in Cloud computing is that SLAs have some weaknesses: responsibility in case of incident, penalties when the agreement is not satisfied, protection from legal risks are not clearly defined.

Vendor Lock-in

Cloud consumers should take into account what could happen if the provider must be changed, as a result of a worsening in the service quality or because of an unacceptable growth of the costs when the contract is renewed. Many organizations are worried that once they use Cloud services, the provider "locks in" their data and systems. The providers that simplify data portability should be preferred, so that the change from a provider to another one is easier. International standards should preserve the customer, but the technological evolution of the Cloud has been so rapid that most of the providers have developed their own solutions relying on ad-hoc protocols. Therefore, customers are often forced to accept a non-standard solution and they have huge difficulties to migrate to other providers.

Migration Issues

In case of problems, it is fundamental to guarantee that Cloud data and services can be moved to another provider. Therefore, a standardization of transfer processes is fundamental.

Performance

Cloud infrastructures could potentially produce unpredictable performance problems caused by the sharing of resources. The fluctuation of individual workloads can determine effects on available CPU, network and disk I/O resources.

Portability and Interoperability

Thanks to the portability, Cloud consumers can easily move data from a Cloud provider to another one. In addition, the portability promotes the competition among different providers. Finally, interoperability enables different systems to uninterruptedly communicate with each other.

Disaster Recovery

Cloud providers might place heavy restrictions on disaster recovery testing procedures. Moreover, information could not immediately be identified properly in the event of a disaster, because of the dynamic nature of Cloud computing.

Lack of Standards

Cloud consumers do not have standardized metrics to evaluate the security status of their Cloud resources. This might cause troubles concerning security evaluation, audit and accountability.

Uncontrolled Viable Costs

A recurring problem in Cloud computing is represented by unpredictable costs, because of its pay-as-you-go nature.

1.4 Cloud Security and Privacy Factors

In order to manage a company properly, IT systems and data are both essential and they can be considered as fundamental resources. Hence, ensuring the confidentiality, integrity and availability of systems and data is critical for companies [53]. The presence of flaws in the IT security could have significant consequences on the organizations, including economic damages, bad reputation and operational risks. Cloud computing, compared to traditional IT environments, is responsible for new threats and weaknesses. Security risks could involve different levels in the IT stack, and a wide set of technologies and policies to safeguard applications, data and infrastructures. Cloud providers and consumers share the risks related to security depending on the Cloud deployment and service model. Sometimes, data require particular security measures (e.g. medical, genetic, financial data). In such circumstances, users should carefully decide whether it would be better to make use of Cloud services or to maintain in-house processing of data, because of threats like accidental loss or unauthorized access.

1.4.1 Identity Management

A user digital identity is needed to manage the access of users to Cloud services: every time they use a new Cloud service, they have to fill out an on-line form and provide sensitive personal information. Entities may have multiple accounts associated with different service providers, or use multiple services offered by the same provider: a Cloud user has to provide his personally identifiable information through multiple platforms, which can be accessed by unauthorized parties if not properly protected. Therefore, an *IDentity Management* (IDM) mechanism to protect private and sensitive information related to users and processes is crucial in Cloud computing. The authentication among heterogeneous Clouds that constitute a federation requires a high level of interoperability

between different security technologies: interoperability issues range from the use of different identity tokens to different identity negotiation protocols. Also multi-jurisdiction issues can complicate privacy mechanisms, because users often interact with services that have to ensure user's identity is protected from other services which, in turn, the first services interact with. In other words, new IDM mechanisms for authentication and authorization are required because data owners and providers are not in the same trusted domain [54].

1.4.2 Security Policies

Even if resources are situated in the Cloud, however the management of an organization is responsible for the introduction of adequate security policies, which must satisfy business objectives. Security policies should include applicable rules and regulations, and an appropriate control over data in the Cloud.

1.4.3 Data Control

When a company stores data at an external location, many problems might arise, such as the loss of control over data, and the possibility that a Cloud provider gains access, modifies or spreads data. An event of data loss or leakage could cause dramatic effects on a company, including financial implications, compliance violations and legal ramifications. The loss of data could occur in many different ways such as: loss of an encoding key, unauthorized parties gaining access to sensitive data, deletion or alteration of data without a backup of the original content.

1.4.4 Privacy

The information stored in public, hybrid or community Cloud is usually shared amongst several users. This causes problems about the possibility for government authorities or competitors to gain access to data in the Cloud without prior consent of the owner. The existence of a flaw in the confidentiality might lead to the disclosure of information to unauthorized individuals or systems. The risks concerning privacy and confidentiality depend on the terms of service, privacy rights and obligations, and on privacy policies defined by the Cloud provider. Privacy issues in the Cloud are now discussed. First of all,

information in the Cloud might have different legal locations simultaneously, therefore determining different legal consequences. Furthermore, there is the risk that external users might access data stored in the Cloud, for example: malicious Cloud providers such as administrators in a SaaS environment, malicious Cloud consumers such as application developers and testers in a PaaS environment, and malicious third-party users in an IaaS environment. External attacks could be represented by remote software and hardware attacks of Cloud applications and infrastructures. The most common security threats involve phishing and software vulnerability exploitation. The purpose of an attacker is to steal credentials, to manipulate data and to eavesdrop sensitive transactions. Providers can take advantage of stronger authentication methods and of proactive monitoring to identify unauthorized activities. Finally, laws could oblidge a Cloud provider to analyze user data to detect any criminal activity.

1.4.5 Integrity of Services and Data

Integrity protects users from any accidental modification of data. Cloud providers should therefore ensure data integrity (consistent and correct) in any situation (i.e. transfer, processing, storage or retrieval). Some threats against data integrity are: cross-site scripting, if the input contains JavaScript code that is wrongly executed by a victim's browser; command injection, if the input includes commands that are wrongly executed via the operating system; SQL injection, if the input contains SQL code that is wrongly executed in the database back-end. Moreover, improper defined security perimeters or improper configuration of virtual machines and hypervisors might cause data integrity problems if system resources are not successfully isolated. Finally, improper access management procedures might cause unauthorized accesses or intentional damage to data. APIs are the only available entry point for users to interact with Cloud platforms: different security risks may occur if the exposed APIs have a flaw.

1.4.6 Availability of Services and Data

"Being available when necessary" is a significant performance indicator for any IT environment. This allows to avoid service interruptions caused by hardware failures, system upgrades, power outages and attacks. There are different concerns about availability in the Cloud. First of all, turn to Internet for transferring data could determine availability

and security problems related to Internet connection and bandwidth speed reduction. Secondly, in a completely decentralized Cloud computing environment, the ability to maintain a stable connectivity between untrusted components while being fault-tolerant is particularly difficult. Moreover, there are some modifications that could cause negative effects, such as software or hardware changes to the existing Cloud services, tests affecting other Cloud consumers or infrastructure changes made by the Cloud provider. The denial-of-service attack might affect all Cloud service models. Its goal is to block access to data and applications. Indeed, it makes resources unavailable, by stressing system hardware. In this case, the consumer has to deal with a service outage and a higher power consumption, which often means huge costs. Other risks include the inhibition of backup procedure testing because of data confidentiality; the consequences upon recovery time may be remarkable when these processes are not tested. Lastly, the availability of the systems could be influenced by single points of failure in the access path to the Cloud.

1.4.7 Encryption

Inadequate encryption of data represents a significant risk in Cloud computing environments. The transfer of data over a network increases the risks of hijacking, leakage or interception. Data leakage can also occur in case of vulnerabilities in the Cloud APIs. A stronger access control, the encryption and protection of the data integrity, and an effective key generation could prevent this type of threat. Furthermore, it is fundamental to define who manages the encryption or decryption keys (i.e. Cloud consumer or provider), and whether or not the encryption technique being used is effective. Key stores also represent a risk and they must be safeguarded in storage, in transit and in backup. When key storage is improperly managed, all encrypted data could be compromised. Therefore, it is important to limit the access to key stores to authorized entities.

1.4.8 Network Security

The introduction of physical data center allows attackers to access the VMs, data and applications in the network from anywhere. The most common network security threats in the Cloud are hacking and intrusion. The creation of a single virtual server hosting

several VMs could determine several network related risks. For example, a new emerging risk is represented by mobile device attacks. Also, intrusion detection, firewalls and other types of protections are weaker in the virtualized environment, making it possible for viruses and malicious software to attack VMs on the same VLAN. Unsafe or compromised VMs can further serve as backdoors to the whole virtual environment.

1.4.9 Laws and regulations

To guarantee both internal and external data security, organizations must comply with requirements, which are set by an industry or by a government body. In the Cloud computing often an off-site location, outside the legal coverage of the organization, hosts data and applications. In the case of public Clouds, if the provider moves sensitive data on servers situated in countries that do not have the same privacy standards or copyright laws, the customer should be properly informed. Indeed, these regulations could conflict with the Cloud consumer's legal or regulatory obligations in his/her home country. It needs to be said that the employment of federated Clouds, as well as the exchange of data between data centers in different countries, is considered to be the cause of the worsening of this kind of problem. Therefore, wherever data are located, compliance needs to be ensured. Lastly, Cloud computing introduces taxation issues both in the Cloud provider's and in the consumer's country.

1.5 Green Cloud Computing

Green Computing is defined as *"the study and practice of designing, manufacturing, using and disposing of computers, servers, and associated subsystems, such as monitors, printers, storage devices, and networking and communications systems efficiently and effectively with minimal or no impact on the environment"* [55]. Energy efficiency is becoming increasingly important for Cloud data centers too; indeed, because of their growing scale and wide use, energy consumption and carbon footprint have become a great issue. The SMART 2020 report [56] provides an estimate of global emissions, which will increase from 40 to 53 $GtCO_2$ (Gigatons of CO_2) by the year 2020. *Information and Communications Technology* (ICT) sector accounts for 2 percent of the worldwide global energy emissions, which could be halved through the extensive use of Cloud computing.

In addition, ICT could contribute to decrease by 15 percent global emissions caused by non-ICT processes. Cloud service providers need solutions to reduce carbon footprint and to guarantee QoS constraints. The European Union has pointed out that emission reductions of 15% - 30% are needed before year 2020 to limit the worldwide temperature increase to less than 2 ℃. Cloud computing reduces costs, improves efficiency and creates a more sustainable world, while maintaining performances at an appropriate level. The following factors, related to Cloud computing, contribute to decrease carbon footprint and reduce energy consumption costs:

- **Dynamic provisioning**: Cloud providers allocate resources on demand, reducing the number of unused servers that are needed for a short period and are mostly idle during the year; in traditional systems, computing resources are oversized to cope with the worst case, in which energy demand reaches highest peaks. Cloud computing allows to reduce waste, balancing supply and demand of resources.
- **Multi-tenancy**: Cloud providers assign the same resource to multiple companies at a time, reducing the number of running servers and carbon emissions.
- **Server utilization**: an on-premise infrastructure runs with low utilization, with no efficiency. On the contrary, virtualized hardware allows executing different applications on the same server in an isolated environment. Therefore, there are utilization levels up to 70 percent, impossible to achieve with on-premise resources.
- **Data center efficiency**: data centers use different methods such as water or air-cooling, and advanced power management techniques to maximize energy efficiency. Statistics show that Cloud data centers save electricity around 40 percent more than traditional data centers.

To enhance energy efficiency in Cloud computing, it is important to study the distribution of power in data centers. One major contributor to today's data centers power consumption is their physical infrastructure (e.g. power and cooling equipment), which is used to support the IT equipment (e.g. compute, storage, network). Studies have shown that physical infrastructure alone can amount to more then 50 percent of the total data center power usage [57] (see Figure 1.10); servers consume 80 percent of the total IT load and 40 percent of total data center power consumption. Companies like Google, Facebook and eBay adopt the *free cooling* technique to lower the air temperature

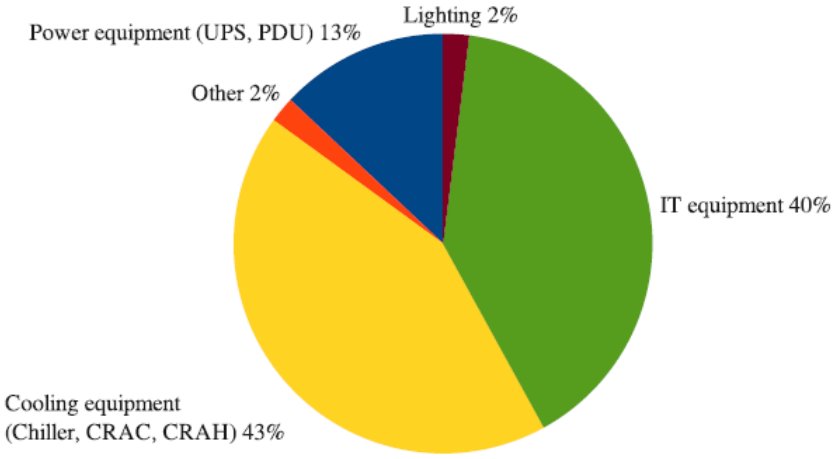


FIGURE 1.10: Data center power consumption

in data centers, by using naturally cool air or water instead of mechanical refrigeration. Consequently, the power needed for cooling decreases, reaching 100 percent of energy saving in areas where climate allows zero refrigeration. The awareness in the data centers energy consumption has become the main purpose of companies. The focus is on how to lower power consumption costs and, on the other side, to maximize the system's efficiency. The objectives must be pursued during the initial phase of the system design, in order to use techniques and models that increase the green benefits. This process imposes new challenges to the system engineers: the design of the architecture must involve green aspects regarding the interactions among the system components, the network devices and the software stack. Among the objectives that are often pursued by data center owners we can mention, on one side, the reduction of the operational costs, manual maintenance and the carbon emissions, and on the other one, the increase of the devices lifetime. Some of these purposes are addressed by the *Data Center Infrastructure Management* (DCIM) layer, which tends to include all the assets and physical resources in the traditional management procedures, such as IT asset life-cycle management and facilities monitoring.

1.5.1 Motivations for Greening Cloud

To show transparency and corporate responsibility, the number of IT companies that voluntarily disclose their carbon emissions along with efforts to be environment-friendly, is in constant increase every year. These companies pursue green certifications for their buildings and businesses, and provide integral reporting of their environmental impact

every year. In general, carbon emissions are mainly disclosed for marketing, financial, and governmental motivations: indeed, firms' value decreases significantly if carbon footprint is high or information about carbon emission rates are kept secret. From the financial standpoint, a recent study [58] shows that the firm's value decreases, on average, by US$ 212,000 for every additional thousand metric tons of carbon emissions produced. This result translates to a firm value penalty of US$ 1.4 billion. On the other hand, governments are issuing taxes on carbon emissions, whose cost per ton of CO_2 is between 25 and 30 US$, with a financial impact on the companies leading to over US$ 92.8 billion in annual costs to be paid by them.

1.5.2 Major Causes of Energy Waste

Among data center typical assets, there are critical equipments (also known as IT equipments), which are necessary to the service delivery, and non critical equipments that are essential to the correct operation of the first category of devices (e.g. power units, cooling equipment and generators). Both the equipment types are very energy demanding and they can be characterized by inefficiency: regarding the IT devices, reasons of a poor energy efficiency can be found in the utilization of old servers and power supply units that consume too much power, or in the unbalanced ratio between the physical resources and the actual needs. Servers are one of the main power consumers in data centers, as seen in Section 1.5. The most important causes are the following:

- **Low server utilization**: the number of servers is quickly increasing but most of them are underutilized. Indeed, according to Natural Resources Defence Council (NRDC) report [59], average server utilization was between 12% and 18% from 2006 and 2012, while consumption was between 60% and 90% of peak power. By consolidating virtual servers on a smaller number of hosts, server utilization increases, therefore the number of required servers and overall energy use will be greatly reduced.
- **Idle power waste**: servers are idle about 90 percent of time [59], during which they consume about 70 percent of peak power. Indeed, even when the server is not loaded, energy is consumed to run OS and to maintain essential hardware devices.

- **Lack of a standardized metric**: there is not a standardized metric of server energy efficiency to enable scheduling algorithms to make decisions, maximizing energy saving.
- **Energy efficient solutions not widely adopted**: small and medium sized data centers are very inefficient compared to big Cloud farms.

1.5.3 Terminology

Two fundamental terms, namely power and energy, are defined. Electrical power is defined as the rate at which electrical energy is transferred by a circuit. It is measured in Watt or Joules per second. Electrical power is computed by multiplying the current (I) with voltage (V) (see Equation 1.1):

$$P = I \times V \tag{1.1}$$

Current represents the amount of electrical charge (i.e. number of coulombs) flowing over the wire per second, referred to as Amperes (Amps). Voltage represents the change in electrical potential energy per unit of charge on the wire. It is measured in joules per coulomb. There is a direct relationship between I and V: the greater the voltage the more current will flow. On the other hand, energy is a quantity typically measured in Watt-seconds (Ws). It is defined as power consumed over a period of time (see Equation 1.2):

$$E(T) = \int_0^T P(t)\,\mathrm{d}t. \tag{1.2}$$

Power is an instant value while energy is the integral of power over a period of time. While the definition of power management and energy management appears clear in theory, in practise the distinction between the mechanisms is often blurry as both terms are used interchangeably. To quantify power consumption in Cloud computing, it is possible to use power measurement techniques that measure actual power consumption through monitoring tools.

1.5.4 Power Measurement Techniques and Power Efficiency Metrics

Data center power measurement techniques allow to account for the data center power consumption usage and efficiency. Power usage can be valued by means of direct, indirect, or hybrid power measurements. Direct power measurements are typically performed using hardware that is either embedded into the equipment (e.g. server) or externally attached. Indirect power measurements are performed by estimating the power usage using power models. They are useful because most servers in modern data centers are not equipped with power measurement devices and VM power cannot be measured by sensors. Finally, hybrid power measurements combine direct and indirect power measurements.

In addition to power models, the first step towards being able to optimize the physical infrastructures power usage and compare it with other data centers is the ability to measure their power efficiency. Therefore, in 2007 the Green Grid defined two data centers efficiency metrics [60], namely *Power Usage Effectiveness* (PUE) and its reciprocal *Data Center infrastructure Efficiency* (DCiE):

- **Power Usage Effectiveness** (PUE) [61]: it is defined as the ratio of the total power entering the data center to the power used by the IT equipment (see Equation 1.3). It is a measure of how much energy is used by the computing equipment in contrast to cooling and other overheads. For example, a PUE value of 2.0 means that for every watt used to power IT equipment, an additional watt is required to deliver the power and keep the equipment cool. In the ideal world, a PUE equals to 1 is desirable. This would imply that all power going into the data center is consumed by its IT equipment. Obviously, the reality looks different as some power is required to support data centers physical infrastructure. Moreover, the actual PUE heavily depends on the current IT infrastructure load and physical infrastructure conditions [62]. For example, when the IT infrastructure is fully utilized ($\approx 99\%$), it will typically imply a higher IT equipment power usage thus decreasing the data center PUE.

Modern data centers have a PUE close to 1.12 (e.g. Google[1] data centers in the second quarter of 2015).

$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power} \tag{1.3}$$

- **Data Centre infrastructure Effectiveness** (DCiE): it is defined as the ratio of the IT equipment power to the total data center power usage, and the result is multiplied by 100 to capture what percentage of the power entering the data center was consumed by the IT equipment (see Equation 1.4). It is the reciprocal of the PUE. For example, a PUE value of 3.0 corresponds to a DCiE of 33 percent, meaning that IT equipment was consuming 33 percent of the facility's power and this is very inefficient. Power entering the data center can be captured by the utility meter. IT infrastructure power can be measured at the output of an *Uninterruptible Power Supply* (UPS). UPS provides backup power to the IT infrastructure during periods of power grid outages. It is plugged in between the power grid and the IT infrastructure.

$$DCiE = \frac{1}{PUE} = \frac{IT\ Equipment\ Power}{Total\ Facility\ Power} \tag{1.4}$$

Other useful metrics are:

- **Data Center energy Productivity** (DCeP) [63]: it is defined as the ratio of useful work to the total energy consumed by the facility to produce this work (see Equation 1.5). It is a measure of the productivity of the data center, in contrast to the PUE and DCiE that do not take into account the useful work. The period of time over which energy is measured is called *assessment window*. It should be no shorter than about twenty times the mean run time of any of the tasks initiated in the assessment window.

$$DCeP = \frac{Useful\ Work\ Produced}{Total\ Data\ Center\ Energy\ Consumed\ Producing\ this\ work} \tag{1.5}$$

Total data center energy may be estimated based on a measured value of the total energy consumption of the IT equipment multiplied by the current data center

[1] http://www.google.com/about/datacenters/efficiency/internal/

PUE value. Decreasing PUE or equivalently increasing DCiE has the effect of improving DCeP.

The Green Grid proposes the following equation for useful work (see Equation 1.6):

$$Useful\ Work = \sum_{i=1}^{M} V_i \cdot U_i(t,T) \cdot T_i \tag{1.6}$$

where:

- M is the number of tasks initiated during the assessment window;
- V_i is a normalization factor that allows the tasks to be summed numerically;
- T_i is 1 if task i completes during the assessment window and 0 otherwise;
- $U_i(t,T)$ is a time-based utility function for each task, where the parameter t is the elapsed time from initiation to completion of the task, and T is the absolute time of completion of the task.

- **Carbon Usage Effectiveness** (CUE) [64]: it is a rating of the quantity of CO_2 emissions produced per unit of energy consumed in a data centre (see Equation 1.7).

$$CUE = \frac{Total\ CO_2\ emissions\ caused\ by\ Total\ Data\ Center\ Energy}{IT\ Equipment\ Energy} \tag{1.7}$$

The denominator is the same value as the denominator of the PUE metric and at the numerator there are the total carbon emissions caused by total data center energy. The units of CUE metric are kilograms of CO_2 per kilowatt-hour (kWh);

- **Data Center Predictive Modeling** (DCPM): it is the ability to forecast the performance of a data center in the future, e.g. its energy use, energy efficiency or cost.

1.5.5 Power Saving Policies in Cloud Computing

A plethora of works addressed the problem of reducing the energy consumption inside a single data center. A solution is to exploit hardware counters to predict application behaviour, reducing energy consumption up to 24 percent [65]. Authors in [66] propose a framework for energy measurement and automatic decision making for resource allocation, characterized by different steps: present real time energy consumption data

to users; involve users in decisions; develop automatic resource allocation techniques to have a trade-off between performance and energy consumption. Another solution consists in consolidating the workload onto a reduced number of servers; in more detail, in [67] authors formulate an *Integer Linear Programming* (ILP) problem where the decision variable is the number of servers used to run the tasks, and the objective is to minimize power consumption, the costs of running tasks and costs due to task migration. Also the problem of task assignment to clusters of servers inside a data center is formulated as an ILP problem, with the objective to minimize the number of used servers [68]. An additional energy reduction mechanism such as *Dynamic Voltage Frequency Scaling* (DVFS) is considered in [69]. The key idea of DVFS is to reduce the CPU frequency and voltage, and consequently power consumption, during periods of low utilization. The dynamic power consumption of a CPU can be approximated as (see Equation 1.8):

$$P = C \times f \times V^2 \tag{1.8}$$

where C is the switching capacitance, f the switching frequency, and V the supply voltage. A linear reduction in voltage implies quadratic power saving. As this method decreases the number of instructions the processor executes in running a program, the program takes a longer time and the performance reduce; moreover, additional energy is required to rise the frequency and voltage level back when required. In Section 1.5.2, we said that servers are idle about 90 percent of time, during which they consume about 70 percent of peak power; DVFS acts at server level, therefore its power saving is low compared to other methods.

Servers could be powered down or put into sleep mode when they are unused; this technique is complex but efficient, and it was proposed to dynamically turn on and off servers, achieving significant energy saving [70] [71] [72]. Important reduction in energy consumption ($\approx 25\%$) can be achieved by introducing virtualization in data centers and consolidating VMs [73], to face the IT equipment under-utilization: it allows to drastically reduce the number of used servers and to avoid the commitment of a single server to a dedicated activity. The different VMs of the virtual data center require guaranteed bandwidth between them for communication. Authors in [74] presented a resource management framework that adopts dynamic VM migration and a consolidation algorithm to minimize the number of active servers during low-demand periods. Dynamic optimization and further workload consolidation into an even fewer number of server can be

performed thanks to VM live migration. Live migration is a facility implemented by all the hypervisors that enables the transfer of a running virtual machine from a source host to a destination one, assuming a shared storage for the VM images, with almost negligible downtime for the applications. Thermal-aware algorithms to allocate resources inside a single data center have also been proposed in the literature [75] [76], with the aim of consolidating VMs onto servers in close proximity to each other, lowering cooling costs. Energy-efficiency has also interested networking architecture and protocols design. This task is also referred to as *Green Networking*: the usual approaches deal with techniques for dynamically adapting network devices processing capabilities and available bandwidth to meet traffic load and services requirements, or standby/wake-up modes [77].

1.6 Research Challenges and Contributions

Computing has evolved over time according to different paradigms, along with an increasing need for computational power. The first paradigm was the *Mainframe Computing*, in which processing power was concentrated in a centralized system accessed through simple terminals. The second paradigm was the *Stand-Alone Computing*, which consisted of workstations that reduced the time to access to computational resources and gave the chance to work in a parallel way; instead, *Network Computing* enabled users to exploit distributed computing resources on networked servers, and changed the way applications were distributed, going from a client-server to a web-based browser-server paradigm. Afterwards, the exponential growth of science and engineering problems required large amounts of computational resources, and scientists relied heavily on parallel and distributed computing to solve them. These computing paradigms worked by separating problems in small tasks and solving them in parallel on clusters. In this context, the *Cluster Computing* solutions became very popular: physical nodes were interconnected through a LAN in the execution of compute-intensive and data-intensive tasks that could not be executed on a single computer. An additional computing evolution was triggered by the massive geographic distribution of very powerful computing resources: *Grid Computing* emerged in the mid 90s. Grid computing enabled controlled computing and data sharing among different physical domains. The reason behind the constitution of a Grid-based infrastructure was the need for enterprises to solve complex

scientific problems, through coordinated resource sharing and collaborative problem-solving. Modern computing paradigms basically share the same underlying concept of *Utility Computing*, that is a service provisioning model through which a shared pool of computing resources is used by a customer when needed. The objective of *Utility Computing* is to maximize the utilization and bring down the relative costs.

Nearly a decade ago, the concept of *Cloud Computing* emerged as a virtualization technique where services were executed remotely in a ubiquitous way, providing scalable and virtualized resources. The spread of *Cloud Computing* has been also encouraged by the success of the virtualization, which is one of the most promising and efficient techniques to consolidate system's utilization on one side, and to lower power, electricity charges and space costs in data centers on the other.

Over the last few years, the *Information and Communications Technology* (ICT) sector has become a significant source of energy consumption and pollution. Indeed, ICT accounts for 2% of the worldwide global energy emissions (equivalent to the global aviation industry [78]), which will increase from 40 to 53 $GtCO_2$ between 2002 and 2020. The European Union has pointed out that emission reductions of 15 - 30% are needed before the year 2020 in order to keep the worldwide temperature increase within the limit of 2 ℃. Several organizations are adopting virtualization and Cloud technologies with the aim of reducing power consumption, creating a more sustainable world, and increasing the overall resource efficiency, while maintaining performances at an appropriate level. But this is not enough. In fact, the growth of Cloud computing and virtualization techniques, fueled by an increasing demand for large-scale processing, has led to the establishment of large scale data centers. However, these same data centers that provide Cloud services, at the same time bring about a tremendous rise of electricity consumption: data center energy consumption grew to an estimated 40 GW in 2013, and it is expected to increase considerably by 2020. In the coming years, networks and data centers will contribute to 25% of the ICT power consumption. According to a report published by Greenpeace [79], the Cloud is comparable to a country with the sixth highest power consumption in the world. This phenomenon occurs because current data centers host lots of applications with dedicated servers, storage and network infrastructures, and therefore resources are vastly over-provisioned to meet the application service goals. All these factors cause high power consumption.

In addition to energy consumption, also carbon footprint of the Cloud infrastructures is becoming a serious concern, since a lot of power is generated from non-renewable sources. As a consequence, the environment is deeply impacted by more *Greenhouse Gas* (GHG) emissions, since most of the data centers are still powered by either coal or nuclear power plants. In this sense, a fundamental measure to be taken would be to reduce the data center energy consumption when its utilization is low, so as to cut down the *Total Cost of Ownership* (TCO) (the sum of initial CapEx added to ongoing and long-term OpEx) and carbon footprint. Besides ensuring eco-friendly solutions, Cloud service providers must take into account *Quality of Service* (QoS) constraints, in order to satisfy SLAs in the presence of dynamic resource sharing and unpredictable interactions across many applications. Such a discourse explains the recent growing interest in green Cloud computing among the research community and ICT industry. The aforementioned problems can be solved by designing energy-efficient resource allocation solutions, according to which resources are assigned with the aim of satisfying user requirements and optimizing energy consumptions. The whole problem of resource allocation in Cloud while minimizing energy consumption and carbon footprint remains a very challenging issue.

After having analyzed the main issues concerning the interaction of ICT and energy consumption, another important dimension to be considered is the run-time orchestration of virtual resources in Cloud infrastructures. The service demands coming from customers are rapidly growing, and Cloud providers deploy an even increasing number of geographically distributed data centers, with consequent management challenges. Requested services are becoming increasingly sophisticated since users need to deploy their own applications with the topology they choose, while having the control on both the infrastructure and the applications.

In this context, a Cloud service model that has gained a lot of attention over the past years is commonly referred to as Infrastructure-as-as-Service (IaaS). In IaaS, the resources (i.e. compute and storage) are provisioned on-demand by the Cloud providers in the form of *Virtual Machines* (VMs). IaaS systems enable the joint deployment of infrastructures and applications, and their realization requires a control platform for orchestrating the provisioning, configuration, and management of the virtual resources over physical hardware. Yet, since the orchestration process is complex and potentially

error-prone if performed manually, the need for a tool that minimizes human intervention in order to automatize resource configurations and to remain scalable is necessary. Indeed, an easy configurable system allows to reduce costs and scale data center energy consumption. From the Cloud provider's perspective, however, building a system to orchestrate resources is challenging, due to the growth of data centers, QoS constraints and SLAs.

One last important dimension at which we are interested is to preserve priority services in Cloud Computing. Cyber-attacks on Cloud systems pose high risks to Cloud service provider infrastructures and customer data. Even if the administrators are unable to isolate the attacker, solutions are needed in order to guarantee bandwidth and QoS under a *Denial of Service* (DoS) attack.

In this thesis, we address the aforementioned problems and their multiple facets. We investigate new energy-efficient models and algorithms, and introduce a Cloud-based solution to preserve priority services in case of attacks. In Chapter 2, a novel energy-aware resource orchestration framework for distributed Cloud infrastructures is introduced, in order to manage both network and IT resources in a typical optical backbone. We discuss the enabling technologies for the next-generation optical transport network deployment, detailing the *Generalized Multi-Protocol Label Switching* (GMPLS) fundamentals in order to facilitate the computation of traffic engineered paths. Then, the *Path Computation Element* (PCE) is presented, which is the key component of the proposed architecture. The PCE provides functions of path computation in support of traffic engineering in networks controlled by GMPLS. We provide a high-level system architecture overview by focusing on the definition of the different layers of the whole infrastructure, and introducing the main components of the resource orchestrator. Finally, a green migration plan that is obtained by applying resource relocation algorithms is discussed.

In Chapter 3, the aforementioned resource management framework is used to offer resources to service providers in the form of virtual infrastructures, while ensuring that energy consumption and CO_2 emissions are minimized. Service providers can easily deploy services directly into the infrastructure of the network providers and invoke them upon the customers' requests, by meeting the Quality of Service (QoS) objectives of end-users. This is what we call "resource orchestration". Resource orchestration is supported by a centralized software entity, named *Virtual Resource Orchestrator* (VRO), the aim

of which is to optimally allocate the resources needed to instantiate a requested virtual infrastructure. The resource orchestrator is designed to adaptively adjust to varying workloads so that high resource utilization and Quality of Service can be achieved. The VRO is a resource allocation system that manages network and IT resources to achieve energy-aware objectives and SLAs on shared virtualized infrastructures, and that also performs distributed VM management. Particularly, the VRO integrates a power management mechanism that automatically detects overloaded and underloaded physical machines, and consolidates VMs in order to release lightly utilized servers. In more detail, we deal with the problem of reducing energy consumption and carbon footprint in distributed Clouds while, at the same time, preserving SLAs and providing performance guarantees in terms of minimum guaranteed bandwidth and maximum activation time for switch interfaces, through dynamic reconfiguration of the network. Furthermore, we propose a novel traffic-engineering algorithm that minimizes energy consumption and CO_2 emissions of networks connecting multiple distant data centers, based on energy cost and carbon footprint metrics. To evaluate the effectiveness of our proposal, extensive simulations are conducted in a test scenario composed of eight data centers connected through an optical backbone. Then, we have developed a framework for emulating energy-efficient network environments and for automatically deploying these in the Cloud, so that network providers can customize and automatically generate running testbed instances based on the customization. Finally, a prototype of the VRO architecture has been implemented on a local testbed, and configured to experimentally evaluate the proposed platform. The experimental results have proven our system to be energy-efficient.

In Chapter 4, a *Virtual Intrusion Detection System* (VIDS) is developed. We discuss related work about Cloud security. Then, a framework for a secure Cloud infrastructure is proposed. It contributes to react to malicious attacks by implementing a real-time traffic analysis, providing re-actions and using ubiquitous control systems, in order to protect high-priority services. The VRO-based architecture has been proven to be flexible enough to adapt to varying requirements and objectives in the management of network infrastructures.

Chapter 2

A PCE-based Architecture for Combined IT and Network Orchestration

2.1 Context and Motivations

Nowadays, the *Green Cloud* is one of the most popular topics in ICT. The *Green Cloud* research is focused on the development of new methodologies in order to minimize energy consumption without affecting negotiated service levels. New Cloud infrastructures are needed, which combine flexibility, *Quality of Service* (QoS), and energy savings.

In this chapter, we present a *Green Cloud Enabler*, which is an energy-aware resource orchestration framework for IaaS Cloud infrastructures; the *Green Cloud Enabler* combines innovative methods and mechanisms useful to dynamically improve energy efficiency of both network and IT components of multi-site Cloud infrastructures [80]. The objective is to reduce energy consumption of Cloud (IT) and networking devices (ICT) by taking into account operational metrics collected from physical devices. The *Green Cloud Enabler* is based on the standard *Path Computation Element* (PCE) architecture [81], which is a scalable inter-domain/intra-domain centralized controller. Even if the research community is actively working in the network power management area [82] [83], little has been done to increase energy efficiency of all the resources involved in a multi-site Cloud infrastructure; in fact, recent efforts aim at developing energy saving techniques

addressed to each single device. Network power management usually concerns performance and sleep states, which minimize power consumption when switches are on or idle respectively. The performance states dynamically change the rate of links and their associated interfaces. The sleep states quickly turn off network interfaces when they are idle. Such solutions are useful in traditional ICT systems, while in Cloud infrastructures, *Virtual Machine* (VM) placement and dynamic VM scheduling algorithms are also required to consolidate the computational load into a subset of physical resources, in order to trigger power saving mechanisms. The problem of finding the right allocation of VMs into a set of physical machines in Cloud data centers is equivalent to the multidimensional, multiple-choice and multi-constrained *Bin Packing Problem* (BPP) [84]: *"Given a set of virtual machines and a pool of physical hosts, each one described by a multidimensional resource vector, the objective of the problem is to find the best mapping of the virtual machines to the physical hosts, maximizing resource efficiency usage"*. The problem is multi-dimensional because the mapping depends on a multidimensional resource vector capturing the resources associated with both hosts and VMs, such as CPU utilization, network bandwidth and RAM. Moreover, the opportunity of having multiple choices is due to the fact that only one physical host for each available group must be selected in order to allocate a VM. Finally, the bin packing is often subject to multivariate constraints. An additional VM allocation problem is however enabled by the live migration mechanism implemented by common hypervisors. The live migration allows the transfer of a running VM from one physical machine to another, by assuming a shared storage for the VM images, with a slight downtime for the applications. The live migration makes possible to react to the fluctuating resource requirements of the VMs. Today, live migration and consolidation are the most important techniques related to the efficient resource allocation in data centers [85], because the live migration allows the reallocation of VMs on several physical hosts, thus maximizing the resource utilization. For instance, when the demand is low and there is an inefficient utilization of resources, several virtual machines can be consolidated into a lower number of physical hosts, in order to turn off the idle devices, thus saving power consumption. However, an aggressive consolidation might overload the hosts because the usage of resources begins to saturate, leading to a slow-down of the application execution and to a higher power consumption; these effects might have negative consequences on the QoS, thus violating the *Service Level Agreements* (SLAs). Instead, when VMs request an increasing demand

of resources, they might be migrated to physical hosts that have a lower load, thus preventing SLA violations. Hence, the VM consolidation must find a trade-off between power consumption and QoS [86]. A new variant of the VM allocation problem must be addressed, which consists of choosing the set of migrations to be performed in order to consolidate the VMs into fewer physical hosts, starting from the current allocation of the virtual machines and taking into account QoS and load fluctuations. The aim is to minimize energy consumption. In literature, several VM consolidation algorithms have been proposed, but they are rarely integrated in realistic multi-site Cloud environments. In this chapter, a novel energy-aware resource orchestration framework for distributed Cloud infrastructures is discussed. The aim is to show how both network and IT resources can be managed in a typical optical backbone while, at the same time, the overall power consumption and carbon footprint are being minimized. Then, a *Green Migration Plan* is proposed in order to orchestrate the resources by applying dynamic consolidation algorithms.

2.2 Overview of Optical Transport Networks

This section reviews the enabling technologies for the next-generation optical transport network deployment. The focus is then put on dynamic wavelength-routed optical transport networks, specifically on those deployed according to the ITU-T ASON architecture and provided with a GMPLS control plane, as defined by the IETF.

2.2.1 Architectures of Backbone Telecommunication Networks

In the past, backbone networks had a multi-layered architecture that was composed of several layers with different technologies, in order to manage the increasing IP data traffic demands while keeping traditional voice-centric traffic support. Because of the Internet's prominence and widespread of standard Ethernet for LANs, IP was utilized for introducing and providing different services, becoming a service integration layer in the next generation architecture of the network. IP packets transported application data, but IP did not guarantee QoS, *Traffic Engineering* (TE) functionalities and reliability mechanisms, which were capabilities provided by ATM; therefore, running IP over ATM

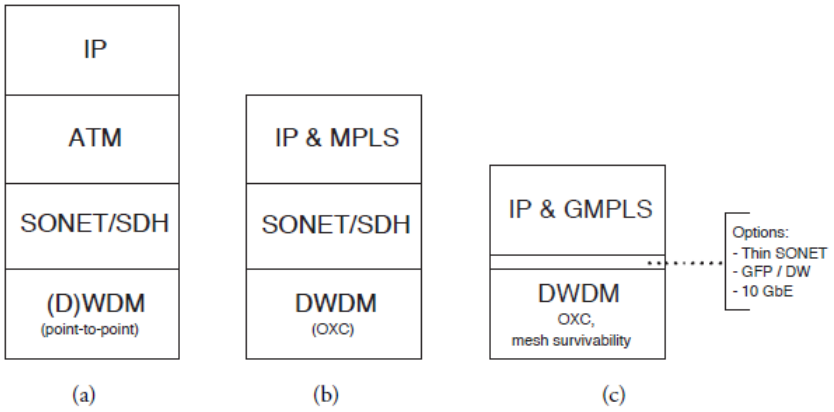


FIGURE 2.1: Evolution of the backbone network architecture

was a solution that added to IP the missing features. Finally, *Synchronous Optical NETworking* (SONET) and *Synchronous Digital Hierarchy* (SDH), responsible for network resilience, were used as a transport layer to carry traffic over fiber, while *Dense Wavelength Division Multiplexing* (DWDM) increased optical fiber capacity. Multi-layered architectures have the following drawbacks:

- Resource utilization is not optimal because of the overhead caused by multiplexing traffic coming from upper layers into flows of a lower layer, resulting in a 80% utilization of lower layer links [87]. Moreover, only 50% of SONET/SDH capacity is used to transport payload because of protection switching techniques, further reducing the utilization factor.
- Every layer acts autonomously, increasing network management complexity and network costs.
- The introduction of new services is very complex because of a different management system for each layer.
- Overlapped functionalities.

The evolution of backbone telecommunication networks is shown in Figure 2.1. ATM is gradually replaced by *Multi-Protocol Label Switching* (MPLS) and *Generalized MPLS* (GMPLS); moreover, many tasks of SONET will be delegated to DWDM, resulting in IP/GMPLS over DWDM with a thin layer (Thin SONET, *Generalized Framing Procedure* (GFP), *Digital Wrapper* (DW) or 10 *Gigabit Ethernet* (10 GbE)) between them. Transport network will also evolve from SONET rings, with *Add-Drop Multiplexers*

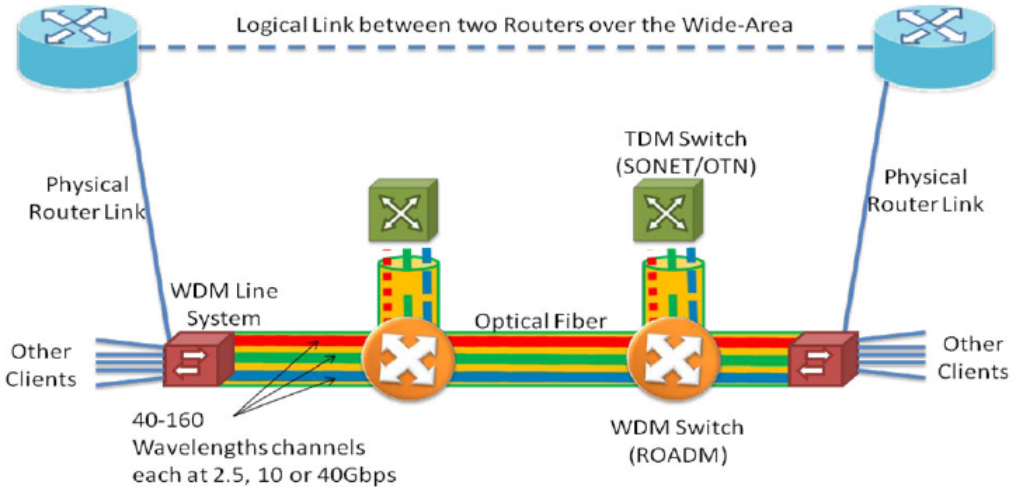


FIGURE 2.2: IP connectivity in the transport network

(ADMs) interconnected through point-to-point links, to *Optical Cross Connects* (OXCs) in a mesh configuration.

2.2.2 The Transport Network and the Internet

The current Internet backbone is composed of packet-switched IP networks that have autonomous ownership, administration and management. To achieve global connectivity, these networks use *Border Gateway Protocol* (BGP) to advertise IP address reachability and choose routes across routing domains known as *Autonomous Systems* (ASs), which are interconnected through network equipments (gateways or routers). IP networks have automated control mechanisms that include routing and signalling protocols implemented in each router, which is responsible for traffic routing and forwarding. Moreover, management functions (configuration, monitoring and maintenance) are carried out by highly qualified personnel. IP packets are physically transferred between routers in an underlying network known as transport network. Transport networks currently support different clients: IP networks, cellular networks, private networks and *Public Switched Telephone Networks* (PSTNs). Over recent years, traditional voice services have moved to IP as well as cellular data and voice; private networks are moving to packet-based solutions too (e.g. Virtual Private Network). Therefore, in the near future the only client for transport networks will probably be the Internet. Transport network includes high-capacity wavelength division multiplexed fiber links terminated at the WDM line-systems. A fiber link carries multiple optical signals by employing *Wavelength Division*

Multiplexing (WDM). The WDM technology splits the optical transmission spectrum into a number of non-overlapping wavelength (or frequency) bands, each one supporting a single communication channel, which are transported within the same fiber. Different WDM (e.g., Dense or Coarse) technologies can be found depending on the spacing between the channels to be transmitted [88]. Each wavelength channel is capable of carrying for example 10 or 40 Gbps, with 40, 80 or more wavelengths multiplexed per fiber (see Figure 2.2). As the wavelength channels operate at such high line-rates, traffic grooming is adopted. Traffic grooming is the process of grouping smaller traffic flows into larger units, such that they can be processed as a single entity at a reduced cost [89]. This can be done for example via *Time Division Multiplexing* (TDM) switches. Transport networks are not automated and they are exclusively intra-AS, therefore there is not the equivalent of Internet's inter-AS communication.

DWDM was not able to provide many characteristics of SONET/SDH, such as: framing, protection, restoration, *Operation, Administration, Maintenance, and Provisioning* (OAM&P). As a result, ITU-T solved the problem by defining the *Optical Transport Network* (OTN) architecture, combining the advantages of SONET/SDH and DWDM, and introducing *Forward Error Correction* (FEC) [90]. In this kind of high bit-rate core networks, although data transmission through the links was optical, data had to be electrically processed at each node so that upper layers' tasks (monitoring, QoS providing, etc.) could take place. To do this, *Optical-Electrical-Optical* (O-E-O) conversions were needed. Therefore, despite the improvements introduced by opaque networks, the limitations of the electrical processing of the signal were an important bottleneck for achieving a low-power/high bit-rate core network; as a result, the next step was migrating the backbone towards *All-Optical Networks* (AONs), where the routing and the processing took place in the optical domain. OTN and AON lacked the ability to provide lightpaths automatically according to customer needs, hence ITU-T defined an *Automatically Switched Optical Network* (ASON) model [91]. The ASON model has been generalized for different transport technologies (not only for OTN), and defined as *Automatically Switched Transport Network* (ASTN). In parallel to ITU-T, within the GMPLS framework, IEEE has developed a model of the multi-layer network and of its control plane protocols; the protocols are useful for implementing the ASON/ASTN functionalities.

2.2.3 Dynamic Optical Networks: the ASON Architecture

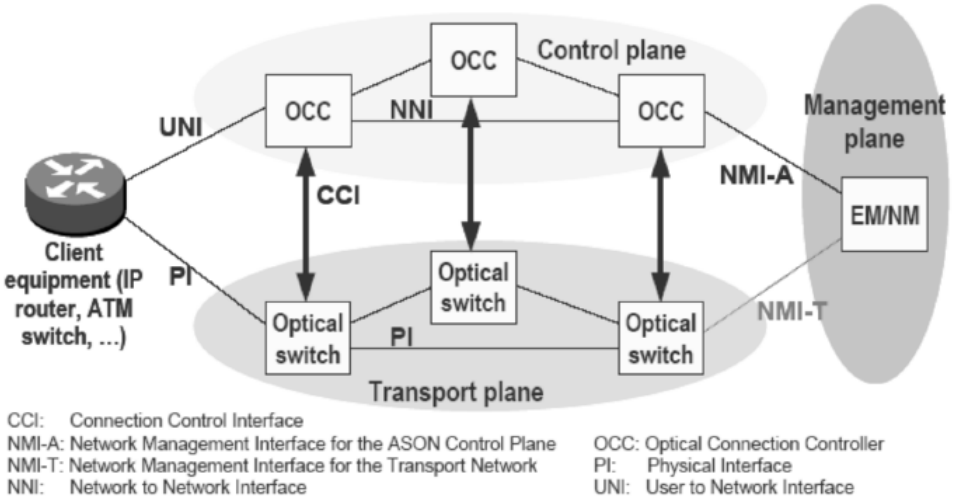


FIGURE 2.3: The ASON architecture

The ASON architecture is defined at a high level of abstraction, introducing three planes (data, control and management) and showing their interactions (see Figure 2.3).

The ASTN/ASON extends the OTN with optical channel connections provided in a fast and automatic way via control plane, in order to achieve dynamic connectivity. Data plane includes optical switches and fibers, hence the transport network where the transfer of user data takes place. Control plane provides intelligence to the network and defines network topology through the propagation of control packets; it can be further divided into:

- **Signalling Plane**: it consists of messages that explicitly reserve a path in the network.
- **Routing Plane**: it consists of intra-AS routing protocols needed to build the routing table.

In other words, the control plane regulates the way routers process packets. To this end, the *Optical Connection Controllers* (OCCs) manage the optical switches located in the data plane, allowing automated connection provisioning, maintenance and release. Finally, the management plane deals with monitoring of devices (measuring throughput and collecting information), and administration of the other planes; furthermore, the management plane can be used to perform connection setup, supervision and tear-down

of optical channels, in two different ways: manually configuring each network device or contacting the control plane, which automatically manages path computation and reservation. Signalling and routing requirements of the ASON/ASTN are satisfied by GMPLS protocol suite.

2.2.4 The GMPLS Protocol Suite

The Generalized Multi-Protocol Label Switching suite [92] allows dynamic service provisioning in transport networks, by creating an intelligent and automated "unified control plane" to manage different switching regions. It is chosen to implement control plane capabilities in the ASON. GMPLS is obtained by extending MPLS, a mechanism standardized by IETF [93], which uses a label-swapping scheme, rather than address matching, to determine the next hop for a received packet: the packet is forwarded according to the incoming label, which is then swapped by an outgoing label. In MPLS, routers have a look-up table with a mapping between "incoming interface/incoming label to outgoing interface/outgoing label" [93]. But in MPLS, only *Packet Switched Capable* (PSC) interfaces could be managed; the interface identifiers are typically IP addresses, since IP flows between two nodes are normally transported through a single link. However, in an optical network a huge number of fibers, each carrying tens of wavelengths, can be deployed between two network elements. It is then clear that using IP addresses to identify such a huge number of elements is not feasible. Therefore, in GMPLS the notion of label can be generalized, and *Time Division Multiplexing Capable* (TDMC), *Lambda Switched Capable* (LSC), or *Fiber Switched Capable* (FSC) interfaces can be managed in addition to the PSC ones, with timeslots, lambda or wavelengths respectively represented as a label. In other words, GMPLS allows the configuration of nodes with different switching capabilities along a LSP.

Besides, by introducing TE capabilities, GMPLS enables the establishment of *Label Switched Paths* (LSPs) that satisfy network constraints and user preferences, while avoiding congested network elements and increasing the network resource efficiency [94]. A *Traffic Engineering Database* (TED) repository at each node is provided to collect TE information, so as to optimize path computation according to network resources.

Unlike MPLS that manages both data and control plane, GMPLS operates only within the control plane. GMPLS-based control plane deals with routing, signalling and resource management. For routing purposes, *Open Shortest Path First* (OSPF) and *Intermediate System to Intermediate System* (IS-IS) protocols have been proposed, adding extensions to support Traffic Engineering [95] [96]. In the context of ASON/GMPLS networks, the routing involves two main tasks: topological information dissemination and path computation. The former is performed by the routing protocol (e.g., OSPF-TE) and enables the latter, which is carried out by the route computation engine of the OCC. To deal with signalling, the TE version of *Resource reSerVation Protocol* (RSVP) and the *Constraint-based Routed Label Distribution Protocol* (CR-LDP) are proposed [97] [98]. RSVP-TE is responsible for setup, maintenance and tear-down of a LSP; routers exchange signalling packets to reserve resources. Finally, the *Link Management Protocol* (LMP) is responsible for neighbour nodes discovery and link management tasks [99].

Ultimately, path computation is a key issue for TE in GMPLS networks, because it is a very complex task in multi-layer networks. Hence, a cooperation between network entities is required. Therefore, starting from a model where each OCC was provided with an entity to compute routes, we are heading towards an architecture with a routing entity within the control plane, named Path Computation Element, which monopolizes route computation.

2.3 Path Computation Element

A Path Computation Element is defined in [81] as *"an entity (component, application or network node) and a control plane concept, which is capable of computing a network path or route based on a network graph and applying computational constraints during the computation"*. The PCE provides functions of path computation in support of traffic engineering in networks controlled by MPLS and GMPLS. In such networks, when a LSP has to be signalled over a pre-computed path, a PCE performs complex route computations on behalf of the head-end MPLS *Label Switching Router* (LSR), which is termed *Path Computation Client* (PCC) (see Figure 2.4). In this way, the path computation process is decoupled from path establishment and can be performed by taking into account TE information and physical constraints [100]. Instead, a head-end router has partial visibility of the network topology to destination (in complex

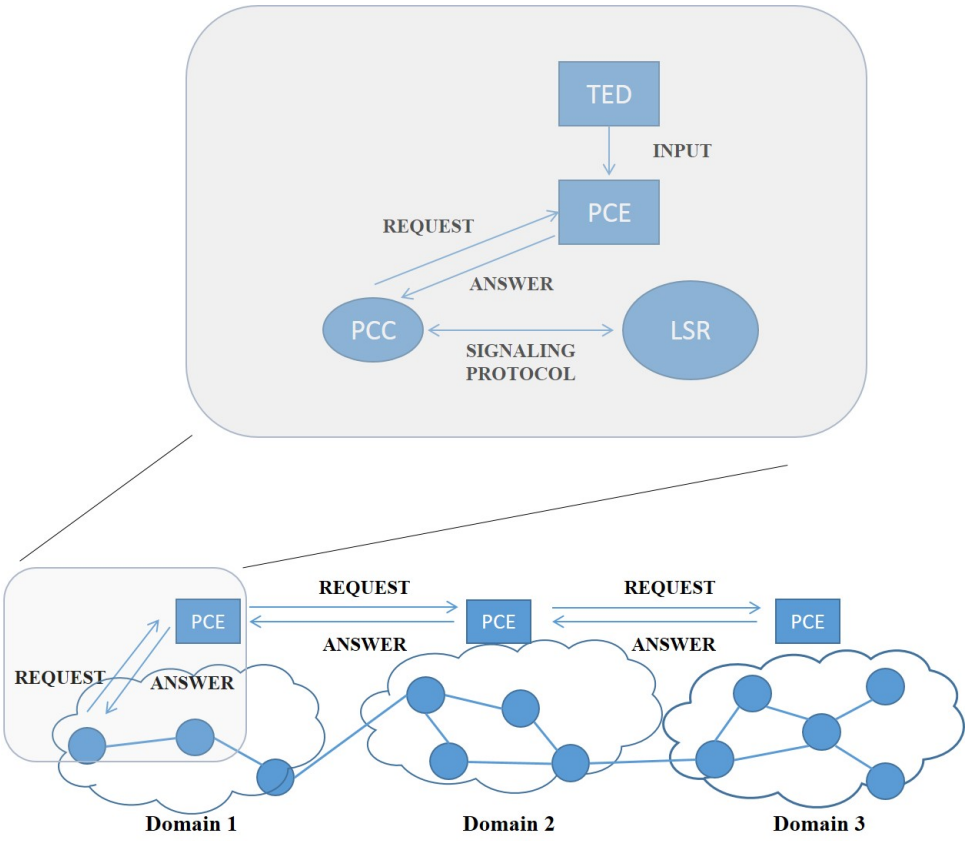


FIGURE 2.4: The PCE architecture

multi-layer, multi-domain or multi-administrative network architecture) [101], so it can calculate an end-to-end intra-domain path but not an inter-domain path.

The PCE-based inter-AS path computation can be performed after the AS chain to the destination is known, by using efficient distributed algorithms and topology information resulting from dissemination mechanisms [102]. In addition, PCE might not have full topology visibility and, in this case, it is able to compute only a loose route. The PCE supplies optimal routes and interacts with the control plane for the set-up of the proposed paths upon receiving requests sent by a PCC, which could be another process or a node, to determine the path from a source to a destination.

In order to execute this task, network state information is gathered into a TED: it contains candidate paths and it is populated with intra-domain routing protocols (OSPF-TE, IS-IS-TE) and BGP information (BGP routes available before winnowing the best route) [100]. By employing information included in the local TED, the PCE identifies primary and backup paths within a domain or an area, and it commonly considers bandwidth requirements, as well as QoS and survivability characteristics.

The PCE computes an inter-domain route, which is provided as *Explicit Route Object* (ERO); the ERO is created and signalled [103]. Through EROs, it is possible to signal a mix of strict and loose hops to be used in the path; in order to establish the LSP, the ingress LSR uses RSVP-TE and encodes the path. A hop might also be a whole AS and it is termed "abstract" node; given a specific domain, abstract and loose hops correspond to a set of strict hops between the ingress *AS Border Router* (ASBR) and the next hop ASBR [100].

The state of the network components is required for routing: the *Interior Gateway Protocol* (IGP) distributes this information through *Link State Advertisement* (LSA). The aim of a PCE-based model is to coordinate the establishment of LSPs among distinct areas of a single domain or within a small group of domains, by means of heuristics conceived to address path computation problems. According to the size, each domain at the inter-domain level might include one or more PCEs to facilitate load sharing and avoid single point of failures; for instance, large domains can be divided into manifold areas, in each of which one PCE manages path computations. At least one PCE per domain is required and it might be located in the same node of the PCC.

A path may be computed by a single PCE if it maintains enough topology and TE information; when an individual PCE does not have sufficient TE visibility, PCEs can cooperate to compute constrained end-to-end inter-domain paths, without sharing any TE information with each other. In this way, the topology visibility issues are solved. In particular, in a multi-domain scenario a PCE interrogates the PCEs of other domains, acting in turn as a PCC. PCE-based model can be of two types [102]: peer-to-peer or hierarchical. In the first case, PCEs of adjacent domains interact with the control plane, collaborate to interchange routing information and they are sequentially queried to determine the availability of the path. In the hierarchical approach [104], there is a local PCE for each domain and a centralized global PCE, which computes paths after receiving information from each domain. In the latter case, the drawback is the limited scalability and the presence of a single point of failure.

A *PCE Communication Protocol* (PCEP) [105] was defined to specify both PCC-PCE and PCE-PCE communications aimed at the computation of LSPs. When a new request arrives, the PCC uses a discovery method to locate PCEs; then, the PCC locally stores PCE capabilities to select one of them according to the specific computation. Finally,

the PCC submits an inquiry to the selected PCE by using PCC-to-PCE communication. The use of a PCE eliminates the need for every node within the network to compute the path, and all link state information is sent only to the PCE. PCEs are particularly useful when end-to-end constraints for protection or path diversity must be taken into account. The deployment of a dedicated PCE will relax the processing power needed by a network node to run constraint-based routing algorithms, and to implement highly CPU-intensive optimization techniques. Moreover, the PCE eliminates the need for the network nodes to maintain the memory demanding TED. By using the PCE, the LSRs are released from intensive computations such as finding disjoint QoS paths.

This approach has two practical advantages. First, the PCE-to-PCE communications provide a scalable path computation scheme, since the responsibility and "visibility" of each PCE ends up in the corresponding AS. Second, the PCE supplies an appealing approach to ISPs, since PCE hides network topology of downstream domains. Moreover, the approach is simple because each PCE computes a segment of the LSP based on its knowledge of the state of resources within its AS, and on the reachability information obtained from BGP. Unfortunately, the major drawback of computing paths by segments is that the resulting paths are likely to be far from optimal. The issue that remains wide open is how to exploit the PCE-based model to compute high-quality primary and backup LSPs across a small group of domains in a viable way, that is, without adversely affecting scalability and confidentiality.

2.4 Green Cloud Enabler Architecture

The *Green Cloud Enabler* is a management infrastructure that orchestrates both network and IT resources. This framework combines innovative methods and mechanisms useful to dynamically improve energy efficiency of both network and IT components of multi-site Cloud infrastructures; moreover, the *Green Cloud Enabler* guarantees flexibility and QoS. A high-level overview of the system architecture is shown in Figure 2.5 [106] and is made up of three layers: the *Physical Layer* (PHL), the *Virtual Layer* (VL), and the *Management Layer* (ML). At the PHL, network nodes and physical servers are grouped together to form a cluster; the physical machines host VMs. Each device is managed by a *Local Controller* (LC), which is composed of *Network Managers* (NMs) and *IT Managers* (IMs). The *Network Managers* are in charge of interact with network nodes,

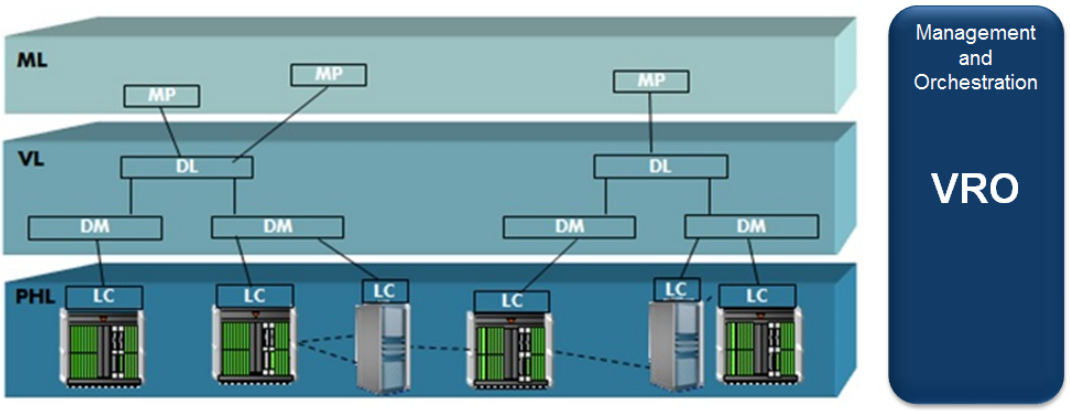


FIGURE 2.5: Green Cloud Enabler architecture

while the *IT Managers* communicate with servers and VMs. The VL provides scalability to the system, and it is composed of a *Domain Leader* (DL) that manages a subset of *Domain Managers* (DMs). Each *Domain Manager* oversees a group of *Local Controllers* and virtual machines. Finally, the ML provides the user interface to the customers and includes several *Management Points* (MPs). The *Virtual Resource Orchestrator* (VRO) is a distributed control system that can selectively and simultaneously invoke *Domain Leaders*, in order to assemble an end-to-end path and several computational resources. The VRO will be described in Chapter 3.

When *Local Controllers* are powered on, they are assigned to *Domain Managers* according to a round-robin algorithm. The *Domain Leader* is responsible for this assignment. Moreover, the *Domain Leader* is also in charge of handling user's resource deployment requests; in detail, the *Domain Leader* distributes these requests among the *Domain Managers*, and each VM will have an IP address at boot time. The assignment is done by considering the current resource utilization of the *Domain Managers*, which is an information stored in the *Domain Leader* database. The mapping between network/IT resources and *Domain Managers* is one of the available "customer-side" information. Therefore, by interacting with the suitable *Domain Managers*, users are able to send power commands with the aim of controlling virtual machine operations: start (power on), stop (power off), reset (reboot), suspend, live migration. The *Local Controllers* are in charge of interact with network routers and physical servers, with the aim of executing commands coming from the *Domain Manager* (see Figure 2.6). The mapping between network resources and *Local Controllers* is an information stored in the

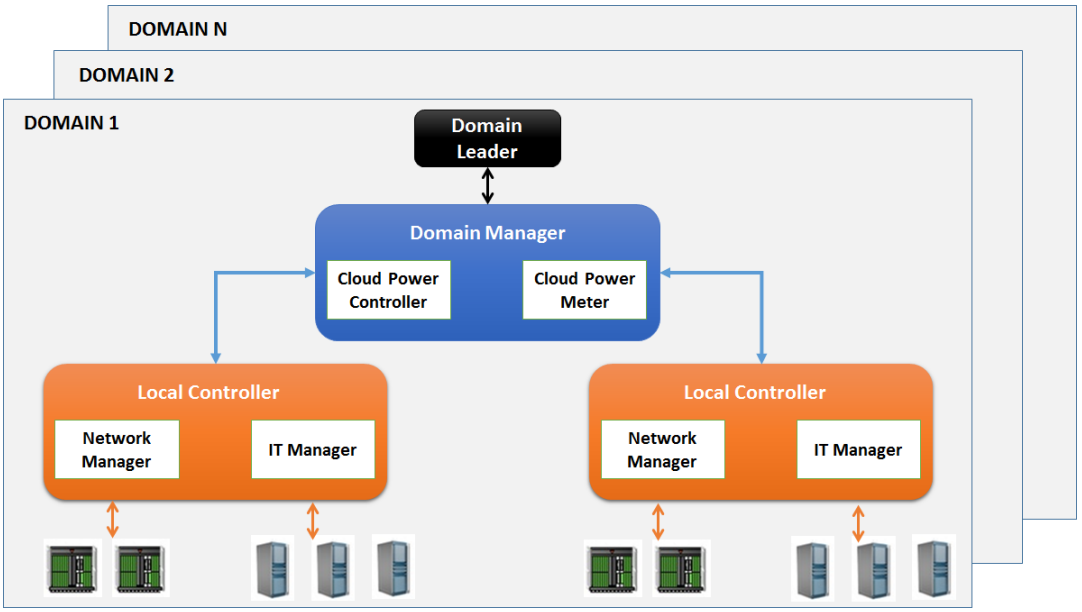


FIGURE 2.6: Interactions between Domain Leader, Domain Managers and Local Controllers

Local Controller database. Moreover, *Local Controllers* monitor the resource utilization of network devices and detect underloaded and overloaded hosts. Finally, resource utilization data are sent by *Local Controllers* to the *Domain Manager*. Based on the information received, the *Domain Manager* takes decisions about VM allocation and consolidation (in order to solve host overload/underload problems), and computes the appropriate network path in order to satisfy energy-efficient objectives. VM allocation and consolidation decisions contribute to the creation of a *Green Migration Plan*, which specifies the new mapping between network/IT resources and *Local Controllers*. Finally, the *Local Controllers* interact with network routers and physical servers, with the aim of executing commands that are received from the *Domain Manager*. Virtual machines are periodically consolidated into a lower number of physical hosts, in order to turn off the idle devices, thus saving energy consumption. Finally, the *Domain Leader* might change over time; therefore, the *Management Points* are used to discover the current active *Domain Leader*, and this information is sent to the client. In the next section, the *Domain Manager* and its components are described.

2.5 Cloud Power Monitor and Control

The *Domain Manager* includes two key elements of the *Green Cloud Enabler* architecture: the *Cloud Power Controller* and the *Cloud Power Meter*.

2.5.1 Cloud Power Controller

The *Cloud Power Controller* (CPC), as part of the *Domain Manager*, is specialized in complex VM migration and path computation strategies, while satisfying energy saving objectives. Therefore, by relying on information collected about resource state, the CPC outputs a migration plan that takes into account the resource energy costs. In our scenario, the CPC is based on the PCE standard architecture, and it uses path selection algorithms and VM consolidation strategies in order to minimize network energy consumption.

A PCE, as stated in Section 2.3, is a functional element that cooperates with similar entities and with network devices in order to compute the best path through multiple domains, according to network constraints and green requirements. Our proposal assumes that network and IT devices are able to provide PCEs with energy consumption information. In this context, we consider the opaque *Link State Advertisements* (LSAs) of the OSPF protocol [95] to distribute power consumption values of network resources. The structure of a standard LSA packet is shown in Figure 2.7: the payload consists of *Type-Length-Value* (TLV) objects.

Two types of top-level TLVs are defined: Router-Address-TLV (type 1) and Link-TLV (type 2). New TLVs fields to the traditional *Traffic Engineering* extensions for OSPF-TE have been added. In detail, the energy information is carried by setting up new sub-TLVs inside the Link-TLV. In the sub-TLVs, the *Type* field is chosen in the range (32768,32777), which is reserved for experimental uses. The *Length* field specifies the extension of the *Value* field (in octets). The *Value* field is used to distribute energy information. The new sub-TLVs include the following additional "resource markers":

1. The resource type identifier: network or IT.
2. The power state of each resource. It is defined according to the standard approach proposed in EMAN [107], which identifies 12 power states for a device.

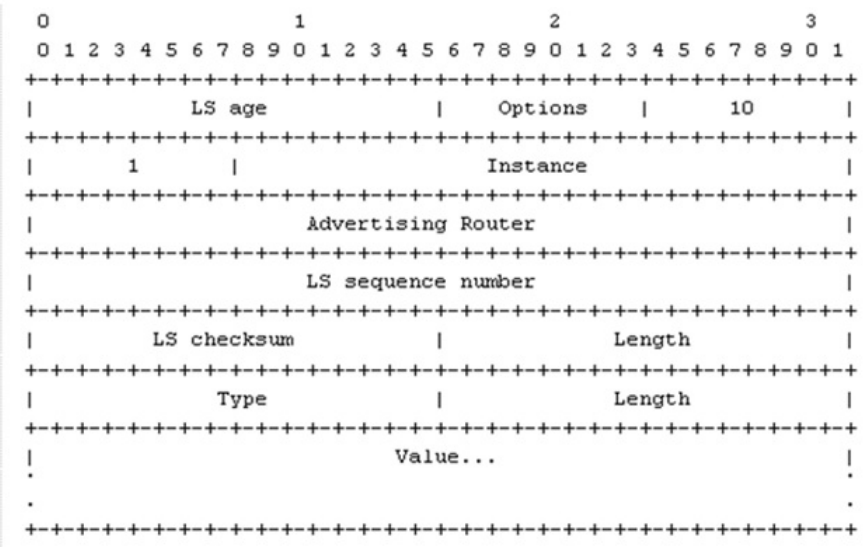


FIGURE 2.7: OSPF-TE LSA format

3. The energy consumption (expressed in kWh) of the resources.

4. The CO_2 emissions of the resources (CO_2/kWh), which depend on the energy source type (dirty/green) used for powering them.

5. The transition time matrix, which consists of the time required to transit from a resource state to another one.

6. The resource localization composed of geographic coordinates that univocally identify the position of a well-defined resource within the Cloud infrastructure (latitude and longitude).

7. The resource utilization percentage (0-100%).

The flooding procedure follows the standard OSPFv2 flooding. After receiving a new LSA, the node decides whether to forward the LSA or to discard it according to the carried timestamp. If a node does not support the aforementioned sub-TLVs, the node forwards the LSA to its neighbours. Once valid LSAs have been received, the *Link State Database* (LSD) is updated. The PCE takes a routing decision that minimizes energy consumption, by also taking into account both minimum guaranteed bandwidth and maximum activation time for the node interfaces (as will be discussed in Chapter 3).

2.5.2 Cloud Power Meter

In a complex Cloud infrastructure, resource monitoring allows to collect information so as to take appropriate scheduling decisions. In order to orchestrate network and IT resources, we introduce new resource markers indicating the power consumption, CO_2 emissions, power state, position and utilization of the resources within the Cloud infrastructure. The *Cloud Power Meter* (CPM) is able to collect resource markers related to both IT and network devices. The main components of the CPM are:

- **Resource Power MAnager (RPMA)**: the RPMA is in charge of the power management of IT and network resources. In particular, the RPMA sends the following power commands: node shutdown, suspend to ram, suspend to disk, suspend network port/channel.
- **Resource Power MEter (RPME)**: the RPME monitors the *Local Controller* utilization and computes the power consumption of each resource. This information contributes to the creation of a *Green Migration Plan*.

2.6 Cloud Enabler Logic

This section introduces the *Cloud Enabler* logic, which consists of a *Green Migration Plan* and several dynamic consolidation algorithms.

2.6.1 Related Work

The VM live migration is a key point of dynamic VM consolidation techniques. Indeed, thanks to the live migration it is possible to dynamically consolidate VMs into the minimum number of physical hosts, so that the idle servers can be turned off in order to reduce the typical energy consumption of a Cloud-based data center. However, VM migrations might create overhead by increasing the network traffic, and the load of both source and destination physical machines. Therefore, it is important to take into account QoS constraints, because an inadequate VM consolidation would degrade performance. Several consolidation algorithms have been proposed in the literature that deal with the problem of optimizing resource usage while reducing power consumption. In [108], two

algorithms related to VM allocation and consolidation are combined in order to reach a common objective that is the minimization of the data center energy consumption. Authors propose an allocation algorithm that is based on an *Integer Linear Programming* (ILP) model, and it aims at energy consumption reduction in the Cloud. The allocation algorithm takes into account VM constraints, such as CPU, memory and storage. Instead, the migration algorithm aims at migration cost mitigation and it is designed as a best-fit heuristic: the VMs are sorted in a decreasing order of energy consumption, and the algorithm tries to allocate the most energy intensive VM into the server with fewer residual resources. Then, idle servers are put into sleep mode. Hence, the migration algorithm aims to minimize the number of running physical machines. Finally, the two algorithms are merged together. The best-fit heuristic is compared with the optimal allocation provided by a linear solver; simulations show that the convergence time of the algorithms grows exponentially when the number of servers or the number of VM requests increases, thus making the approach inadequate for large-scale data centers. Murtazaev and Oh [109] present a consolidation algorithm that is specifically designed to be used with live migration. Given an allocation scheme, the proposed algorithm tries to reduce the number of active servers while minimizing the number of live migrations. The physical nodes are sorted in decreasing order according to their utilization, and then the VMs are sorted in decreasing order too, starting from the least loaded server. Finally, the algorithm tries to reallocate the VMs from the least loaded server to the most loaded one. The reallocation of a group of VMs is triggered only if their migration allows to turn off the server, otherwise the VMs are left on it and the next least loaded server is considered. Moreover, the idea of simultaneous migrations is introduced in order to reduce the overall migration time. Authors compare their algorithm to the *Fist-Fit Decreasing* (FFD) heuristic, and they show that the proposed approach is able to consolidate VMs and to minimize the number of migrations. The algorithm also considers RAM and CPU associated with VMs and servers. However, the time complexity is very high and the algorithm is inadequate for large-scale data centers. Moreover, the energy efficiency problem is not addressed. In [110], a dynamic consolidation algorithm in a virtualized environment is described; the algorithm is based on a FFD heuristic. The future VM resource needs are predicted by considering their history: authors perform a trace analysis on commercial web servers in order to elaborate a method that selects the most suitable servers on which the VMs can be reallocated. The proposed algorithm minimizes the number of physical resources while providing

SLA guarantees. However, the algorithm takes into account neither the current VM placement nor energy-efficient approaches, and the inter-relationship between various workloads is not specified. In [111], the VM consolidation is modelled as a *Constraint Satisfaction Problem* (CSP), and it is solved by using constraint programming. A resource manager (called Entropy) for homogeneous clusters is introduced, which takes into account the migration overhead, memory and CPU requirements. The proposed approach assumes that VM demands are known in advance. Moreover, the VMs are considered as static boxes: the objective is to avoid that the allocation of new demands implicates the opening of a new box. The approach is compared to the FFD heuristic: simulations have shown that Entropy is able to minimize the number of physical machines and migrations. Marzolla et al. [112] present the V-MAN approach, which is based on the *Peer-to-Peer* (P2P) paradigm. The proposed approach aims to solve the consolidation problem. V-MAN uses a simple gossip protocol that allows neighbouring servers to exchange messages about their state (i.e, the number of VMs running on each server); then, VMs are migrated from the least loaded servers to the most loaded ones, so that the idle servers can be switched to low-power mode. V-MAN proved to be scalable, efficient and fault-tolerant by means of simulations. However, the proposed approach only takes into account the number of VMs regardless of their resource needs, therefore servers that have free resources always accept new VMs. This assumption is not realistic. In [113], authors address the problem of the host overload detection by considering a Markov chain model and a control algorithm. The algorithm aims to minimize the *Overload Time Fraction* (OTF) that is the percentage of time during which the host is overloaded. As a consequence, SLA violations and decreased performances are avoided, and QoS requirements are satisfied. Moreover, an host is considered overloaded only when necessary, in order to prevent superfluous migrations. The algorithm is effective only when stationary workloads are taken into account. Then, authors assume a discrete state space, which is represented as intervals of CPU utilization. The transition between several states is described by a probability matrix. Simulation results obtained through PlanetLab traces show that the proposed approach outperforms the best benchmark algorithms. Dhiman et al. [114] present vGreen, an energy-efficient multi-tiered software system; it manages VM scheduling across distinct physical nodes in a virtualized environment. vGreen is implemented according to a client-server paradigm. Authors claim that VM characteristics (memory accesses, instructions per cycle) must be taken into account during a VM consolidation, in order to minimize power consumption. Therefore,

vGreen tries to migrate VMs with similar characteristics to different hosts, because they could create contention if hosted by the same server, thus reducing the performance. To this end, two metrics that capture power and performance information related to VMs are introduced: *Memory accesses Per Cycle* (MPC) and *Instructions Per Cycle* (IPC). vGreen has been implemented on a real tested, and results show that this approach improves both performance and energy savings by 20% and 15% compared to state of the art policies. Finally, authors in [115] propose a framework for green Clouds in order to model both VM allocation and consolidation. In particular, they present a *Modified Best-Fit Decreasing* (MBFD) algorithm, which is an extension of the best-fit decreasing heuristic, in order to achieve a power-aware VM allocation. Then, they describe adaptive threshold-based migration algorithms so as to optimize the VM allocation while minimizing the number of migrations. Two static lower and upper thresholds are used to detect underloaded and overloaded hosts; in the first case, when the host utilization falls below the lower threshold, VMs are migrated from the underloaded server in order to turn it off; instead, if the host utilization exceeds the upper threshold, the VMs with the lowest usage of the CPU are migrated in order to avoid SLA violations. The performance of the proposed approach is evaluated by using the CloudSim simulator.

2.6.2 Green Migration Plan

Network and IT resource orchestration requires a migration plan that specifies a new mapping between resources and *Local Controllers*. The migration plan is achieved by applying the resource relocation algorithms that will be discussed in Section 2.6.3. Migrations can happen either sequentially or in parallel: in the first case, a VM is moved from the source LC to the destination LC one at a time, while in the second case multiple VMs are migrated concurrently. Since modern hypervisors (e.g. KVM) support parallel migrations, these are considered in our scenario. The resource orchestration process consists of the following steps:

1. The *Domain Leader*, included in a *Virtual Infrastructure Provider* (VIP) management system, starts an orchestration process in order to satisfy the requirements of a customer.

2. The *Domain Leader* selects the *Domain Managers* that will be responsible for executing the orchestration process; each *Domain Manager* includes a *Cloud Power Controller* and a *Cloud Power Meter*:

   (a) The *Cloud Power Meter* collects resource markers related to both IT and network devices.

   (b) The *Cloud Power Controller* is concerned with VM migration and path computation strategies, in order to minimize energy consumption.

3. The *Domain Managers* select the appropriate *Resource Relocation Algorithms* (RRA) to satisfy customer's requirements.

4. The RRA are executed according to overload and underload policies, and they produce as output the *Green Migration Plan* (GMP), which gives instructions to compute the best path through multiple domains, according to network constraints and green management objectives. Moreover, the *Green Migration Plan* specifies the new placement for the VMs.

5. The orchestration process is completed and the *Local Controllers* execute commands that are received from the *Domain Manager*.

2.6.3 Resource Relocation Algorithms

The *Cloud Power Controller* takes as input the resource markers collected by the *Cloud Power Meter*; then, the CPC computes a *Green Migration Plan* by taking into account the dynamic consolidation algorithms presented in [116], which will be discussed below. In this section, we describe the techniques that will be used to relocate IT resources (virtual machines), while for a more detailed discussion about the orchestration of network equipment, the reader should refer to Chapter 3. The consolidation problem consists of four steps:

1. Establishing when a host is overloaded so as to migrate VMs from it, in order to reduce the load of the host;

2. establishing when a host is underloaded so as to migrate VMs from it, in order to turn off the host;

3. choosing the VMs that should be migrated from the overloaded and underloaded hosts;

4. finding a new placement for the VMs to be migrated.

The Algorithm 1 summarizes the aforementioned consolidation problem. The VM placement algorithm is invoked only when the list of VMs to be migrated from the overloaded hosts is completed; then, the new placement of the VMs is added to the *Green Migration Plan*.

Algorithm 1 VM Consolidation

```
Input: nodes_list, VM_list
Output: Green Migration Plan
for each node in nodes_list do
    if node is overloaded then
        Select VMs to be migrated
    end if
end for
Find a new placement for VMs
Add VMs to Green Migration Plan
for each node in nodes_list do
    if node is underloaded then
        Select VMs to be migrated
        Find a new placement for VMs
        Add VMs to Green Migration Plan
    end if
end for
return Green Migration Plan
```

2.6.3.1 Detection of Overloaded Hosts

In this section, four overload detection techniques are described. According to these techniques, the upper utilization threshold dynamically adapts to the CPU utilization of the host. In detail, the upper threshold is modified according to the deviation of the CPU utilization: when the CPU is fully utilized, the deviation is higher, therefore the upper threshold must be lowered.

Median Absolute Deviation (MAD)

In order to dynamically adapt the upper bound, the *Median Absolute Deviation* (MAD) is considered; it is an estimator of the statistical dispersion. MAD shows better performance than the standard deviation; indeed, the standard deviation is heavily influenced by values that are distant from the mean, therefore the outliers in a dataset have a strong impact. Instead, by considering the MAD, the outliers are less relevant. Given a dataset $X_1, X_2, ..., X_n$, MAD is defined as the median of the absolute deviations from the dataset's median:

$$MAD = median_i \left(|X_i - median_j \left(X_j \right)| \right) \tag{2.1}$$

The upper utilization threshold T can be written as:

$$T = 1 - \eta \cdot MAD \tag{2.2}$$

where η is a calibration parameter that allows to regulate the trade-off between energy-savings and SLA violations.

Interquartile Range (IQR)

The *Interquartile Range* (IQR) is the difference between the third and first quartiles:

$$IQR = Q_3 - Q_1 \tag{2.3}$$

The upper utilization threshold T can be written as:

$$T = 1 - \eta \cdot IQR \tag{2.4}$$

Local Regression (LR)

The *Local Regression* is used to build a curve that approximates the original data. It is considered local because each smoothed value is determined by neighboring data points defined within a span. The local regression describes a relation between a predictor

variable x_i and a response variable y_i. We consider the following model:

$$y_i = f(x_i) + \varepsilon_i \tag{2.5}$$

where $f(x)$ is an unknown function, and ε_i represents random errors in the observations of data. The Loess (LOcal regrESSion) method is used to estimate the function f. We assume that $f(x)$ can be locally approximated by a function that belongs to a simple class of parametric functions (polynomials of degree 1 or 2). We now describe the method used to achieve a Loess smoothed value for a target covariate x_0. A Tukey's tricube function is considered:

$$T(u) = \begin{cases} \left(1 - \left|u^3\right|\right)^3 & \text{if } |u| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The weight sequence for the observations (x_i, y_i) is defined by the function $w_i(x_0)$:

$$w_i(x_0) = T\left(\frac{\Delta_i(x_0)}{\Delta_{(q)}(x_0)}\right) \tag{2.7}$$

where $\Delta_i(x_0) = |x_0 - x_i|$ is the Euclidean distance between x_i and x_0, and we suppose that $\Delta_{(i)}(x_0)$ are the ordered values of these distances from smallest to largest. Therefore, $\Delta_{(q)}(x_0)$ is the maximum of the distances computed by considering q observations located around x_0 (or the distance from x_0 to the farthest predictor in its local window, the q-th nearest neighbor).

The observations x_i usually have different distances from x_0, therefore a span for each value x_0 is defined in order to select the observations that must have a non-zero weight. Hence, only the values located within a smoothing window $\left[x_0 - \Delta_{(q)}(x_0), x_0 + \Delta_{(q)}(x_0)\right]$ will be considered in order to estimate $f(x_0)$. $\Delta_{(q)}(x_0)$ is the size of the window and is called bandwidth or span: $\Delta_{(q)}(x_0)$ is not fixed but depends on the target x_0. A span of α means that $\alpha \cdot 100\%$ of the values will be used for each interval.

Moreover, we suppose that $f(x)$ is approximated by a polynomial of degree 1:

$$f(x) = y = a + bx \tag{2.8}$$

In order to achieve the local regression estimate $\hat{f}(x_0)$, we have to compute the values of a and b that minimize:

$$\sum_{i=1}^{n} w_i(x_0)(y_i - a - bx_i)^2 \tag{2.9}$$

By using this approach, it is possible to fit a trend polynomial to the last $k = \lceil q/2 \rceil$ observations of the CPU utilization. Let $x_1, x_2, ..., x_k$ be the observations; for each new observation x_k, the proposed approach is used to estimate the next observation $\hat{f}(x_{k+1})$. The *Local Regression* algorithm detects an overloaded host if the following inequalities are satisfied:

$$\eta \cdot \hat{f}(x_{k+1}) \geq 1 \ \cup \ x_{k+1} - x_k \leq t_{max} \tag{2.10}$$

where η is the calibration parameter, while t_{max} is the maximum migration time allowed for each VM allocated to the host.

Local Regression Robust (LRR)

The *Local Regression* approach shows some problems when there are outliers related to heavy-tailed distributions. Therefore, a robust estimation method called *bisquare* is proposed. The method begins with the aforementioned estimate $\hat{f}(x)$. Then, the residuals $\hat{\varepsilon_i}$ are computed:

$$\hat{\varepsilon_i} = y_i - \hat{f}(x_i) \tag{2.11}$$

The *bisquare* weight function is introduced:

$$B(u;b) = \begin{cases} \left(1 - \left(\frac{u}{b}\right)^2\right)^2 & \text{if } 0 \leq |u| < b \\ 0 & \text{if } |u| \geq b \end{cases} \tag{2.12}$$

Let $m = median(|\hat{\varepsilon_i}|)$. New robust weights are computed:

$$r_i = B(\hat{\varepsilon_i}; 6m) \tag{2.13}$$

The local regression technique is repeated, but a new estimate $\hat{f}(x_i)$ is computed with the new weights $r_i w_i(x)$. By using this approach, a new estimate of the next observation $\hat{f}(x_{k+1})$ is computed. The *Local Regression Robust* algorithm detects an overloaded host if the inequalities 2.10 are satisfied.

2.6.3.2 Selection of Virtual Machines

After having determined that an host is overloaded, it is necessary to select the VMs that should be migrated from the host. In this section, four VM selection techniques are described.

Minimum Migration Time (MMT)

This policy selects a VM to be migrated that requires the minimum amount of time to complete a migration, compared to other VMs allocated to the host. The migration time is the ratio of RAM used by the VM to available network bandwidth for the host. A virtual machine i is selected if the following condition is satisfied:

$$\forall j \in M_h, \ \frac{RAM(i)}{BW_h} \leq \frac{RAM(j)}{BW_h} \tag{2.14}$$

where M_h is the set of VMs allocated to the host h, $RAM(i)$ and $RAM(j)$ are the amount of RAM used by the VMs i e j respectively, while BW_h is the available network bandwidth for the host h.

Random Selection (RS)

This policy selects a VM to be migrated according to a uniformly distributed discrete random variable $X = U(0, |M_h|)$, where M_h is the set of VMs allocated to the host h.

Maximum Correlation (MC)

This policy selects a VM to be migrated that has the highest correlation of the CPU utilization with other VMs. Indeed, according to Verma et al. [117], *"The higher the correlation between the resource usage by applications running on an over subscript server, the higher the probability of the server being overloaded"*. Let $X_1, X_2, ..., X_n$ be random variables that represent the CPU utilization of the n VMs allocated to a host, while Y is the VM candidate to be migrated. We have to evaluate how strong is the correlation between Y and $n-1$ remaining random variables. Let $\mathbf{X}$ the augmented matrix of the

$n-1$ independent random variables, while $\mathbf{y}$ is the vector of the observations for the variable Y:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1,1} & & x_{n-1,n-1} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{2.15}$$

Let $\hat{\mathbf{y}}$ the vector of the predicted values of the variable Y:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{b} \qquad \mathbf{b} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{2.16}$$

The multiple correlation coefficient between observed values y_i and predicted values $\hat{y}_i$ is:

$$R^2_{Y,X_1,X_2,\ldots,X_{n-1}} = \frac{\sum_{i=1}^{n}\left(y_i - m_Y\right)^2\left(\hat{y}_i - m_{\hat{Y}}\right)^2}{\sum_{i=1}^{n}\left(y_i - m_Y\right)^2\sum_{i=1}^{n}\left(\hat{y}_i - m_{\hat{Y}}\right)^2} \tag{2.17}$$

where m_Y and $m_{\hat{Y}}$ are the means of Y and $\hat{Y}$ respectively. $R^2_{Y,X_1,X_2,\ldots,X_{n-1}}$ is evaluated for each VM. A virtual machine i is selected if the following condition is satisfied:

$$\forall j \in M_h,\ R^2_{i,X_1,X_2,\ldots,X_{i-1},X_{i+1},\ldots,X_n} \geq R^2_{j,X_1,X_2,\ldots,X_{j-1},X_{j+1},\ldots,X_n} \tag{2.18}$$

where M_h is the set of VMs allocated to the host h.

Minimum Utilization (MU)

This policy selects a VM to be migrated that has the lowest CPU utilization.

2.6.3.3 Placement of Virtual Machines

The placement of virtual machines is modelled as a bin-packing problem: the physical machines are the bins, the VMs to be allocated are the items, the CPUs of the nodes are the bin sizes, and the power consumption of the nodes is the price. The VM placement problem is solved by using an extension of the *Best-Fit Decreasing* (BFD) algorithm, namely the *Power-Aware Best-Fit Decreasing* (PABFD), which was proposed by Beloglazov and Buyya [116]. The algorithm sorts the VMs according to their CPU utilization in decreasing order, then each VM is allocated to the host in which the power consumption increase would be the minimum.

Algorithm 2 Power Aware Best-Fit Decreasing (PABFD)

```
Input: nodes_list, VM_list
Output: VM Allocation
VM_list.sortDecreasingUtilization()
for each VM in VM_list do
    minPower ← MAX
    allocatedNode ← NULL
    for each node in nodes_list do
        if node has enough resources for the VM then
            power ← estimatePower(node, VM)
            if power < minPower then
                allocatedNode ← node
                minPower ← power
            end if
        end if
    end for
    if allocatedNode ≠ NULL then
        allocation.add(VM, allocatedNode)
    end if
end for
return VM Allocation
```

2.6.3.4 Detection of Underloaded Hosts

The host with the least utilization is found. Then, all the VMs are migrated from this host to other hosts only if the VM migration is useful to turn off the server, otherwise no migration is activated. This process is repeated for all the hosts that were not considered overloaded.

Chapter 3

Green Resource Management in a VRO-based Infrastructure

3.1 Context and Motivations

The emergence of the *Network Function Virtualization* (NFV) paradigm allows network service providers to extend their business models by combining the traditional service portfolio with innovative Cloud computing services [118]. By relying on the virtualization of resources, service providers can automate the highly dynamic delivery of virtualized network services and create on-demand multiple isolated virtual infrastructures for their customers. This concept, previously known as network virtualization [119], has recently been further extended to include both communication and computational resources. A *Virtualized Network Function* (VNF) is the virtualization of a network function in legacy non-virtualized networks: network functions are decoupled from the underlying hardware in order to reduce the dependence on dedicated physical resources, and they run as software images by using standard virtualization technologies. In this way, service providers can use a common physical infrastructure to deploy applications and services by allocating virtual resources only when needed. NFV increases business agility, reduces costs, enables faster service delivery and allows service providers to react dynamically to changing market demands. The NFV Infrastructure is the totality of all hardware and software components that build up the environment in which VNFs are deployed, managed and executed. In this context, we use the term *Virtual*

Infrastructure (VI) to denote a set of computational resources (i.e., virtual machines and virtual disk volumes) deployed in a number of distributed data centers and connected by guaranteed-bandwidth virtual links [120]. The aim of creating such Virtual Infrastructures is to provide a given service to a known and variable population of end users. Network Function Virtualization is the best paradigm to implement the idea of virtual and distributed service infrastructures. For this paradigm to be effective, infrastructure providers need more powerful management platforms to efficiently combine management procedures for both communication and IT resources. As a matter of fact, *European Telecommunications Standards Institute* (ETSI) has identified the necessity of *"a consistent management and orchestration architecture"* as one of the challenges to be addressed for successfully implementing NFV.

NFV orchestration has the following requirements:

- Dynamic configuration, provisioning and chaining of VNFs, in order to create innovative services.
- Intelligent service placement by selecting the optimal location (network nodes, data centers) for placing VNFs, and by using real-time analysis and performance monitoring.
- Setting and enforcement of policies based on SLAs.
- Elastic scaling of services based on fluctuating demands.
- Creation, provisioning and monitoring of VNFs.

Hereinafter, we assume that the underlying networking infrastructure supports *Generalized Multi-Protocol Label Switching* (GMPLS) [92], and conforms to the *Path Computation Element* (PCE) architecture [81]. GMPLS enables dynamic topology reconfiguration, while PCE establishes LSPs as virtual TE links for the allocated virtual GMPLS control plane. Such an infrastructure is able to automatically provision and operate guaranteed-bandwidth network connections between given end-points (e.g., the *Wide Area Network* (WAN) core routers of distributed data centers [121]). Based on this assumption, we propose an architecture that assigns to a central management entity, named *Virtual Resource Orchestrator* (VRO), the responsibility for optimally allocating the resources needed to deploy a requested Virtual Infrastructure. Figure 3.1 shows how

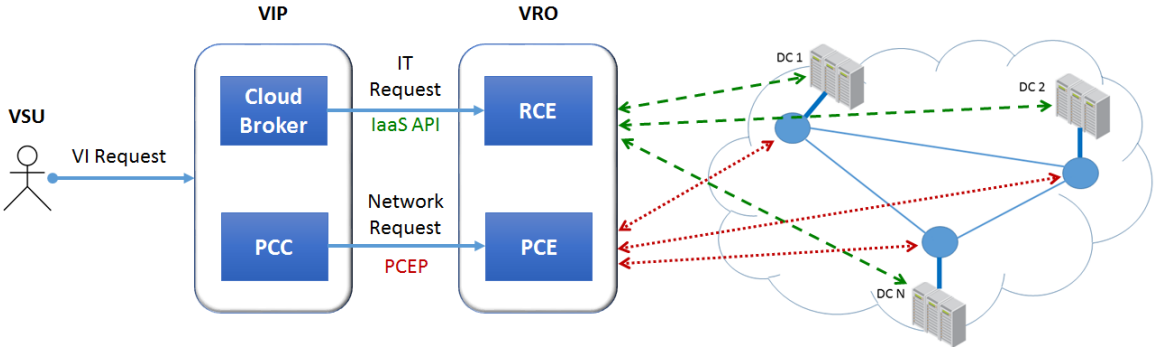


FIGURE 3.1: Creation of a virtual infrastructure on a VRO-based infrastructure

a VI request, issued by a *Virtual Service User* (VSU), is taken by a *Virtual Infrastructure Provider* (VIP), which tasks VRO with deploying virtual machines, virtual volumes and virtual links needed to build the Virtual Infrastructure. The resulting VI must satisfy a set of requirements expressed in the initial VI request. Proper mechanisms (e.g., based on transparent migration of VMs across data centers [122]) to provide recovery functionalities for the created infrastructure could also be implemented, as proposed in [123]. As of today, the creation of a Virtual Infrastructure would be accomplished by statically allocating resources. This process is lengthy, costly, error-prone, and in the case of long-term contracts there is a heavy underutilization of over-committed resources. The purpose of the VRO is to translate a VI request into a number of coordinated network planning and provisioning actions. Being solicited only when creating VIs, the VRO is not subject to heavy load, as we expect only a few VI requests per hour to be processed. Besides assuring contractual SLAs, the purpose of the VRO is to pursue optimization objectives that are compatible with the requirements negotiated in the SLAs. The VRO design has been derived by extending the standard PCE architecture.

One of the most important goals for the network management procedures is to minimize energy consumption of IT and networking infrastructures; it has been estimated that data centers accounted for 1.3% of worldwide electricity use in 2010, as they are one of the major sources of energy consumption of the whole ICT sector [82]. Therefore, finding a compromise between power consumption and the perceived Quality of Service is one of the main objectives in Cloud data centers.

Several techniques to introduce energy-awareness in the resource management of virtualized data centers have been proposed in literature. An energy reduction mechanism such as *Dynamic Voltage Frequency Scaling* (DVFS) is considered in [69]. The key idea

of DVFS is to reduce the CPU frequency and voltage, and consequently power consumption, during periods of low utilization at the cost of a performance degradation; this technique is particularly useful when used in conjunction with workload consolidation [124]. In [125], the DVFS combined with communication overhead and leakage power has been studied; in particular, the problem of simultaneous dynamic voltage scaling of processors and communication links is addressed by the authors. Power aware task scheduling algorithms for real-time embedded systems are developed, so as to achieve 37.4% of power reduction compared to scaling on processors alone.

Several approaches make use of proper task scheduling algorithms in order to reduce energy consumption. In [126], authors present energy-efficient techniques to schedule multiple real-time tasks (with uncertain execution time) in multiprocessor systems that support DVFS, while meeting their deadline constraints, by combining intra- and inter-task voltage scheduling. The authors perform an off-line analysis of the scheduling problem by considering different execution profiles, assuming that the probabilistic distributions of tasks' execution time are available; moreover, they emphasize the benefits of inter-task DVFS to take advantage of slack time. Finally, they compute the optimal voltage scheduling with the *Earliest Deadline First* (EDF) policy, by setting a job with an imminent deadline as the highest-priority job. The drawback of this approach is that the system model is homogeneous, and energy constraints are not considered.

Several techniques in order to minimize the network traffic caused by communications between virtual machines have also been proposed. In [127], authors propose a decentralized affinity-aware migration technique, which takes into account the network transmission traffic between each pair of VMs in order to find affinities and conflicts between co-placed virtual machines. VM placement is dynamically adapted for optimizing communications: virtual machines are migrated to the same physical location when similarities are found, minimizing the communication cost between all the VMs. This approach incorporates heterogeneity and dynamism in network topology, lowers operational costs for Cloud providers and improves hosted application performance.

Energy-efficiency has also interested data center infrastructures and the Internet network topology, which significantly contribute to the overall energy consumption of Cloud computing. Indeed, according to [77], the energy consumption of communication networks will grow 2.5 times by 2018 compared to 2009. Different approaches in order

to reduce network power consumption are available, and they can be classified into: Re-engineering, *Dynamic Power Scaling* (DPS) and Sleeping/Standby. Re-engineering aims to improve the design of network devices, optimizes their internal organization, and reduces complexity levels, by introducing more energy-efficient technologies for network device architectures. Novel technologies for network links are also considered. DPS approaches are used to dynamically adjust packet processing and network interface capacity of network/IT resources, in order to meet the current traffic loads; DPS involves the use of dynamic voltage scaling and idle logic, with a trade-off between packet routing performance and power consumption. Finally, Sleeping/Standby approaches are designed to put unused network devices into stand-by mode, and wake them up only if necessary. Saunders [128] claims that an increase in energy efficiency will result in an increase of the aggregate energy consumption; this makes energy cheaper and leads to economic growth, which in turn might involve an increase in energy demand. Therefore, the CO_2 emissions resulting from data centers must be reduced both by using and generating energy efficiently. Several approaches promote the use of green nodes, which are powered by renewable energy sources [129] [130] [131].

In this chapter, we describe the VRO and its prototype implementation, with an experimental evaluation that shows how the VRO can be configured to pursue significant energy savings, by combining green management procedures in data centers with a green management of the geographical networking infrastructure [132].

3.2 Related Work

The concept of Virtual Infrastructures provided as a service is the basis of the emerging Cloud computing paradigm. In this chapter, we refer to a scenario in which users have control over the geographical location of their virtual machines; moreover, users are provided with contractual guarantees regarding the geographical connectivity among VMs located in different data centers. This scenario has been already studied in literature [120]. Virtual Infrastructures created on-demand might be *Content Delivery Networks* (CDNs), as proposed in recent papers [133] [134]. We claim that NFV might play a significant role in this context. As a matter of fact, ETSI envisions dynamic creation of on-demand Content Delivery Networks as a relevant use case for NFV in [135].

Xia et al. [136] investigate the placement of *Virtual Network Function* (VNF) chains in optical data centers, in order to minimize *optical-electrical-optical* (OEO) conversions. VNF chaining is a carrier-grade process for continuous delivery of services based on network function associations. Authors consider an optical backbone to interconnect several pods, which are performance optimized data centers: modular containers with servers, networking, storage, and cooling. OEO conversions are required because VNF chaining within a pod is based on packet switching, while between pods optical technologies are needed. These conversions are minimized by placing the VNFs of the same chain into fewer pods, in order to reduce the inter-pod traffic. In [137], the authors discuss the role of high-performance dynamic optical networks in Cloud computing environments. Central to the proposed architecture is the coordinated virtualization of optical network and IT resources of distributed data centers, enabling the composition of virtual infrastructures. During the composition process of the multiple coexisting but isolated virtual infrastructures, the unique characteristics of optical networks are addressed and taken into account. Several algorithms are evaluated over various network topologies and scenarios. The results provide a set of guidelines to data center infrastructure providers; in this way, providers can effectively and optimally provide virtual infrastructure services to users, and satisfy their requirements.

The use of virtualization at the edges of a network infrastructure has been proposed in [138]. In the context of wireless-optical broadband access networks (named WOBAN), the authors describe an architecture that places Cloud services (such as processing and storage) in facilities located at the edges of a wireless access network. This approach has multiple benefits: it offloads traffic over wireless links, reduces bottleneck from the gateways of WOBAN, reduces delays, and it allows providers to implement location-dependent Cloud services. Cloud network and resource orchestration requires complex management procedures to guarantee performance, energy efficiency, security, robustness and reliability. Furthermore, there are not so many accepted standards or open source solutions for resource orchestration problems. Therefore, the problem of resource management in virtualized data centers has been largely investigated in the last few years [139]. In order to assess the effectiveness of novel resource management algorithms, several simulation tools have been recently developed, such as CloudSim [140] and GreenCloud [141].

Defining the mapping of virtual resources to physical ones is commonly known as *embedding*, and this problem has been extensively addressed in the context of network virtualization in wide area networks [142] [143]. More recently, the same formulation has been applied to the data center context [144]. Within a single data center, resource management procedures typically rely on live migration techniques for dynamic relocation of VMs. In [145], authors provide some examples related to the dynamic consolidation of virtual machines in a data center, determining which VMs should be migrated from an overloaded host. In this context, authors design a VM selection policy, where not only the CPU utilization is considered, but also a variable that represents the degree of resource satisfaction is defined in order to select the VMs. In addition, a novel VM placement policy that prefers placing a migratable VM on a host that has the minimum correlation coefficient is also presented. In [146], authors propose an effective sequencing technique named CQNCR (read as sequencer) for determining the execution order of massive VM migrations within data centers. Specifically, given an initial and a target resource configuration, CQNCR manages VM migrations in order to efficiently reach the final configuration with minimal time and impact on performance. Experiments show that CQNCR can significantly reduce total migration time by up to 35% and service downtime by up to 60%. Both solutions are very promising, but they do not provide any support for network resource orchestration.

As pointed out in the introduction, energy efficiency is becoming an important objective for IT and networking management. Readers may refer to [147] for an extensive analysis of green management techniques in Cloud computing. At the beginning, the problem of energy-efficient resource management in Cloud data centers was usually not combined with networking management: green networking was usually considered as a distinct issue to be addressed separately [148] [149]. For example, in [150] authors present a survey to design an energy efficient IT infrastructure for Cloud services; they explain the solutions that can be applied at data center and network level, allowing considerable energy savings. They discuss the best practices for energy efficient data centers by considering the power consumption associated with the operation, management and maintenance of computing resources; in particular, they focus on hardware, power supply and cooling of the infrastructure. From the perspective of network energy consumption, authors discuss the solutions proposed in literature in order to increase power savings for access and core networks, such as energy efficient packet forwarding, green routing,

dynamic load balancing and lightpath-bypass strategies. The joint minimization of the energy consumption of network and data centers is not considered.

Data centers that provide Cloud services are subject to an increasing demand of bandwidth with a resulting need for network energy awareness; indeed, the bandwidth demands of new applications are doubling every 18 months [151]. As a result, network energy efficiency has become one of the most important data center design concerns [152]. In [153], a new metric is defined, i.e. *Network Power Effectiveness* (NPE), which is the ratio of the aggregate network throughput to the total network power consumption. It represents the end-to-end *bps per Watt* (or *bit per Joule*) in data transmissions; this parameter is very important for Cloud providers because it reflects the trade-off between power consumption and network throughput in data centers.

The typical evaluation of the network efficiency (focused on network throughput) has been revolutionized, stressing out the importance of the power consumption of the network devices; therefore, there is a need for energy efficiency of the legacy networking equipment, which will be used for many years [154]. Energy efficient protocols for networking devices are emerging, such as IEEE 802.3az [155]: when packets are not transmitted, a *Low Power Idle* (LPI) mode is used to reduce the energy consumption of a link in Ethernet-based communications. In detail, when all packets in the transmit queue are transmitted and the buffer becomes empty, the link switches to LPI mode. Significant energy savings (around five TWh) are achieved compared to the traditional approaches, and sleeping links are not deleted from the topology, maintaining network reliability.

Green networking techniques can be classified into [151]:

- **Consolidation**: network load and traffic are consolidated on a subset of devices to shut down underutilized ones, exploiting path diversity. In [156], network traffic aggregation is considered in order to put idle links into sleep-mode. Authors reported 22% of energy savings for link loads of 50%.
- **Selective Connectedness**: idle devices are put into *Low Power Idle* or sleep-mode as transparently as possible. In [157], authors found that the arrival pattern for the traffic in data centers can be characterized as a log-normal arrival process having on-off periods, so it follows a heavy tailed distribution and it is highly

volatile and bursty. Moreover, the average link utilization in the connections to the aggregate switches is only 8% of the capacity for 95% of the time. Therefore, the Selective Connectedness approach is useful in data center networks.

- **Proportional Computing**: energy is consumed proportionally to device utilization. Proportional Computing methods can be mainly classified into: DVFS (see Section 3.1) and *Adaptive Link Rate* (ALR). ALR methods reduce interface capacity (data rate) as a function of network link load, with 85% of energy reduction compared to fully active devices.

In [158], authors consider network topology along with network traffic demands with the aim of reducing the total energy consumption associated with powered on switches and paths. They propose a *Network-Aware Virtual Machine Placement* (NAVP) problem and a greedy heuristic solution called VMFlow, in order to optimize VM placement and routing of traffic demands. VM allocation occurs by considering servers with enough resources to host a virtual machine, and subsequently the algorithm computes the path that minimizes energy consumption. NAVP tries to consolidate as much traffic demands as possible over the same set of network links, so as to reduce the total energy consumption. This approach has some drawbacks: time-awareness and bandwidth guarantees are not provided; moreover, power consumption of an element does not depend on its utilization, and VM placement mapping is one-to-one (at most one VM is mapped to a server). ElasticTree is proposed in [159] to save energy in data center networks. It is a network-level energy optimizer that utilizes SDN to find a subset of links and devices to serve the current traffic, and turns off the idle switches. Energy savings up to 50% are estimated. However, in high performance networks this solution is not recommended because of performance overhead.

Authors in [160] show that the energy efficiency of the wired Internet is lower than that of a typical 802.11 wireless LAN. They propose to put network interfaces, links, switches and routers that are idle into sleep-mode; then, routers wake up automatically when they sense incoming traffic. An interface sends packets to its neighbours to communicate its status. Finally, a traffic trace is analyzed to evaluate the accuracy of the approach. Authors do not take into account the possibility of VM replacement enabled by live migration. In [161], authors propose heuristics to turn off network links in a WAN scenario, where several service and content providers cooperate in order to achieve an

optimal allocation of network paths and computational resources, so as to minimize energy consumption in case of performance constraints. In [162], data center network architectures have been investigated; authors propose optimization techniques that can be applied only at the design time of data centers, and not when they are running.

Authors in [162] address a flow assignment problem in a data center network by proposing an offline formulation; they also develop an on-line path-consolidation algorithm to reduce utilized paths and save energy, by dynamically turning off idle switches and links, and by turning them on when network is over-utilized. Significant energy savings are achieved compared to the ElasticTree algorithm. In [163], authors address the problem of automatic mapping of virtual elements to physical ones in an acceptable time. They present a technique to consolidate VMs and improve the utilization of physical hosts in a data center; their heuristic is able to map virtual machines and links to physical machines and paths, in order to have an optimal load balancing for the CPU utilization, when there are constraints on hosts resources (memory, storage) and links (bandwidth, latency). Network communication between VMs is optimized, but nothing is said about energy efficiency.

Basmadjian et al. [164] describe power consumption prediction models of the most meaningful ICT resources of data centers (servers, storage devices and network equipment). These models are the rationale of energy optimization algorithms [165]: by applying these policies, through the aforementioned prediction models, it is possible to decrease by 20% energy consumption in a private Cloud data center. Authors propose a control loop for the energy optimization, which consists of three modules: Optimization, Reconfiguration and Monitoring. The state of a data center is periodically monitored; the Optimization module examines this state, then it ranks energy-saving configurations with respect to their power consumptions, which are predicted by the Power Calculator module. Finally, once a configuration is chosen, the data center is reconfigured. To validate these mechanisms, authors create an eight-node private Cloud based on a Cloud controller (running on Red Hat Linux). This mechanism triggers optimization algorithms every time a new virtual machine is created or terminated. The policies are not clearly specified and an analysis is required to determine whether policies are exhaustive. It is also interesting to investigate the case of federal data centers.

In [166], authors perform a survey about methods and technologies deployed for the

energy-efficient operation of computer hardware and network infrastructures, particularly in Cloud contexts; authors review the impact of energy-saving strategies for integrated systems and Cloud management. Then, authors discuss energy aware scheduling in multiprocessor and grid systems. Finally, they suggest to reduce software and hardware energy costs, to improve load balancing and to consider the CO_2 emissions resulting from data centers. Savings of 20% can be achieved in server and network energy consumptions. These policies only focus on a single part of the Cloud infrastructure, either the network or the data center, but it would be interesting to combine them. The authors indicate possible improvements, such as reducing energy consumption due to communications. However, this method depends on the application involved.

Nonde et al. [167] propose an energy-efficient virtual network embedding approach in Cloud networks by consolidating network and data center resources. A MILP framework is presented for modelling their approach in an IP over WDM network, so as to minimize the power consumption by reducing the number of activated nodes and links. Energy savings of 60% compared to other approaches existing in literature are achieved. A heuristic for real-time energy optimization is developed too. Finally, authors show the effects of delay and location constraints on the node embedding problem; for example, this happens when a service requires running applications on virtual machines in a given fixed location (e.g., company headquarters or branch). They conclude that co-location of virtual machines that belong to the same enterprise customer in a data center will save power and costs. Very interesting is the involvement of the optical layer in the link embedding problem.

In the last years, several works that analyze the joint minimization of the energy consumption of network and data centers have been published. For example, in [168] authors propose MILP-based optimization models to accommodate Cloud services with minimum power consumption increase in the backbone network, while introducing minimum power consumption overhead at the data centers. In this chapter, we evaluate the benefits deriving from an energy-aware management of both IT and network resources.

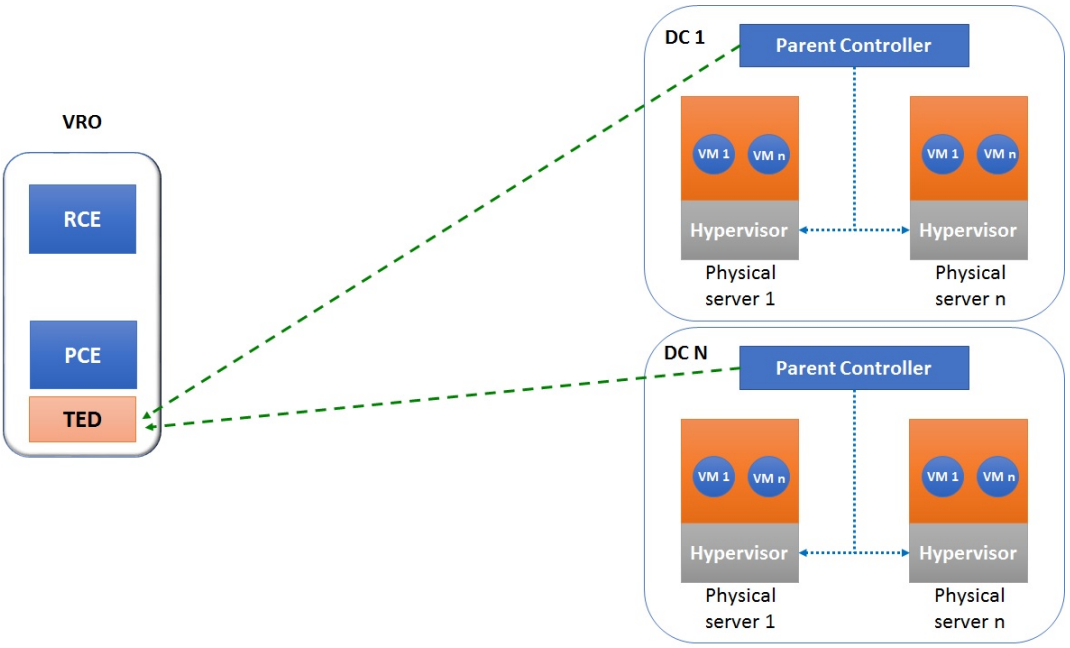


FIGURE 3.2: VRO interaction with IT resources

3.3 Design, Operation and Implementation of a VRO

The flexible intra-domain and inter-domain architecture of the PCE, and the extensibility of the network control plane protocols (OSPF-TE and RSVP-TE) permit to define a new class of network controllers, useful to configure both the network resources and the virtual machines of a typical virtual infrastructure. In this context, we propose an innovative controller, named Virtual Resource Orchestrator, for the provision of network resources and the migration of virtual machines in virtualized infrastructures.

The VRO is a distributed control system that manages network and IT equipment, and it configures the best mix of resources (i.e., links, virtual machines, disks, CPU) by using appropriate configuration protocols and interfaces. VRO inherits the inter-domain/intra-domain Path Computation Element logic related to routing protocols (OSPF-TE) and signalling protocols (RSVP-TE). It monitors the network and IT resource state in real-time, and based on collected data it makes the appropriate decisions during the virtual machine and network life cycles. The VRO implements several functionalities that allow the reconfiguration of the network and IT infrastructure by taking into account design constraints and objectives. For example, in a green Cloud environment the VRO manages the migration of virtual machines and the configuration of network infrastructures, minimizing the energy consumption of physical equipment.

Figure 3.2 shows how the VRO interacts with an IT infrastructure made up of N distributed data centers. Such an interaction allows VIP to deploy virtual IT resources by means of Cloud computing API (e.g., Apache jclouds[1] or OpenStack4j[2]). Furthermore, the VRO collects information regarding the current level of utilization and power consumption of physical resources available at each data center. By collecting this information, the VRO is able to properly decide how the virtual resources should be deployed on top of the available physical infrastructure.

Defining the mapping of virtual resources to physical ones is commonly known as embedding problem; the VRO solves this optimization problem in two steps. In the first step, the VRO decides in which geographical data center the VM has to be allocated. In the second step, it selects the physical server where the VM has to be activated. With regard to the first problem, the VI request might contain an explicit geographical location that limits the available solutions; with reference to the second problem, we assume that VM allocation is delegated to the data center scheduler. Finally, the VRO has to decide the mapping of virtual links to networking infrastructure: the connectivity problem between VMs and data center core routers is solved by data center scheduler, while connectivity at geographical level is addressed starting from the solution presented in [169].

The interaction between Virtual Resource Orchestrator and network elements is illustrated in Figure 3.3. The PCE is generally confined within the control plane to elaborate explicit optimal routes (with related costs), which will be configured as GMPLS tunnels. The resulting route costs are Traffic Engineering indicators used by the network administrator (carrier) to optimize the utilization of network resources. The interaction between VRO and network elements is based on the OSPF-TE protocol.

We have implemented the VRO by extending some of the open source software components that were originally developed within the DRAGON project [170], which is supported by the *National Science Foundation* (NSF). In particular, we have added new features to OSPF-TE/Quagga[3] and RSVP-TE/KOM[4] modules. The VRO could be embedded in each network/IT device (distributed scenario) or held centrally on a distinct node that supports the PCE (centralized scenario), in order to orchestrate resources

[1] https://jclouds.apache.org/
[2] http://www.openstack4j.com/
[3] http://www.quagga.net/
[4] http://www.isi.edu/div7/rsvp/release.html

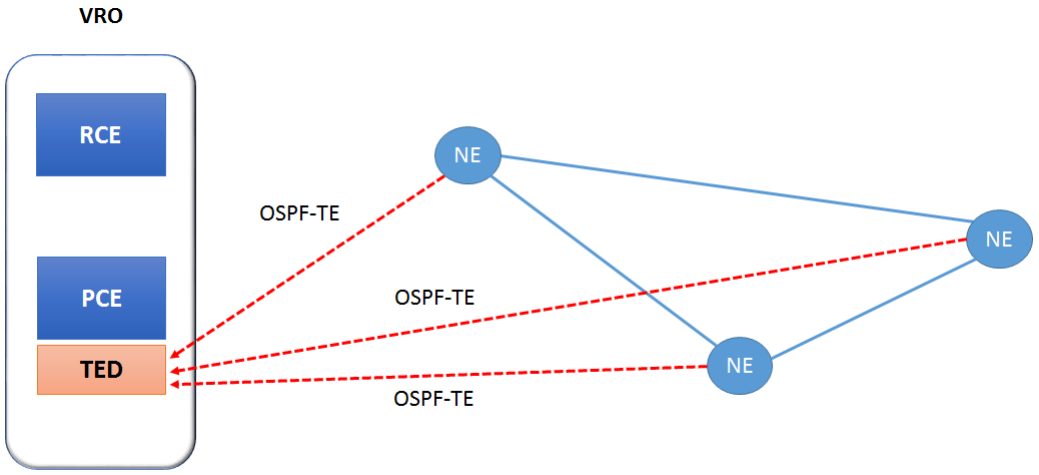


FIGURE 3.3: VRO interaction with network elements

through multiple domains. In this thesis, a centralized computational model is selected. VRO's inter-domain communication and provisioning mechanisms are based on two pillars: a link-state protocol for inter-domain topology exchanges, and a multi-constraint path computation approach to determine suitable network routes.

3.3.1 VRO Functional Components

The VRO is a computational entity that enables network and IT resource orchestration in order to provide virtualized network functions. The main components of the VRO are:

- **Cloud Broker/PCC modules**. They are usually implemented on a *Network Management System* (NMS). They request a "path and IT resource computation" to be performed by the proper PCE/RCE. The request is sent by using the PCEP [105], and it includes information about the desired Virtual Infrastructure. For example, during the specification phase, the PCC might request virtual resources that usually include end points (source and destination addresses), network nodes and links, while the Cloud Broker might ask for a number of virtual machines, OS images, IP address ranges.

- **PCE module**. It is specialized in complex path computation for satisfying the requests received by a PCC. The PCE computes a path based on network state information, network constraints and policies.

- **RCE module**. The *Resource Computation Element* (RCE) module is specialized in virtual machine placement for satisfying the requests received by a Cloud Broker.

- **TED module**. This repository stores information that will be sent to PCE and RCE, such as topology information about nodes and links that connect them, node and link status, virtual machine status, available hypervisor resources and network constraints. In our model, the TED is created from traffic engineering information distributed by the OSPF-TE routing protocol.

- **Policy component module**. It provides PCE and RCE with the policy that impacts resource computation in response to a virtual infrastructure request. A PCE/RCE might apply policies to decide what algorithm to use while performing resource computations.

3.3.2 VRO Orchestration Process

The sequence diagram in Figure 3.4 shows the resource orchestration process performed by VRO with the objective of minimizing the power consumption of the overall infrastructure. The resource orchestration workflow consists of the following steps:

1. Out-of-band TED synchronization. As described in Chapter 2 (see Section 2.2.4 and Section 2.5.1), OSPF-TE uses *Link State Advertisements* (LSAs) packets to exchange information about network topology between routers; moreover, OSPF-TE is properly extended to include energy information in the *Value* field, which is contained in the payload of LSAs. Each router stores the received LSAs in the *Link State Database* (LSD), where each LSA is an entry of the database. When an OSPF router has just been connected to the network, an initial LSD synchronization phase starts: a new neighbour is found and an OSPF router synchronizes its LSD with that of the neighbours, so that they have the same LSA entries. After this procedure was completed, the "asynchronous LSA flooding" mechanism (on a fixed time basis) guarantees the LSD synchronization is maintained between the routers whenever a change to the topology occurs. In our scenario, the TED stores topology information about the GMPLS domain and its synchronization process is not intrusive. Indeed, the VRO monitors the OSPF-TE traffic and places the traffic engineering information into its TED by means of a stateful inspection

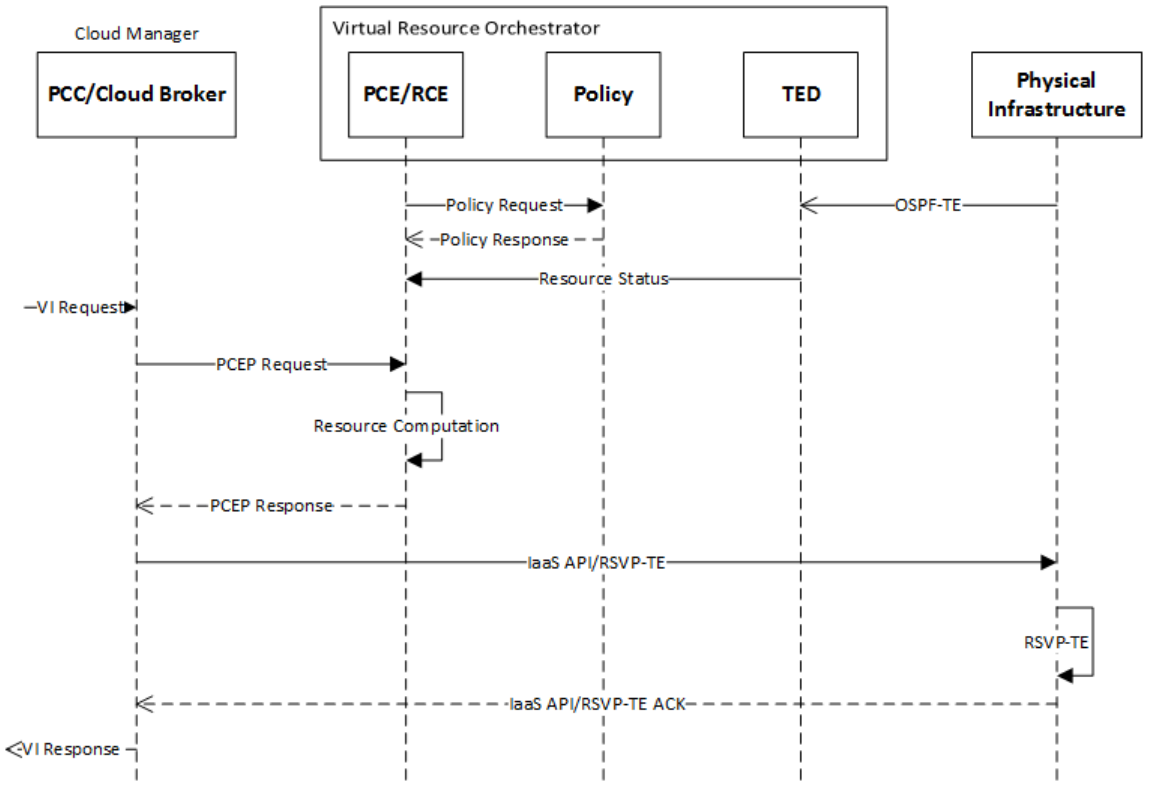


FIGURE 3.4: VRO orchestration process

of OSPF-TE LSAs. At the end of the process, the TED will include optical network topology information and energy-related data associated with nodes and links, and resource (nodes, links and virtual machines) status. At the same time, policies (existing traffic engineering constraints, bandwidth reservations, explicit path inclusions/exclusions, objective functions) may be configured and managed by a network operator, and interpreted in real time by the PCE/RCE.

2. In case of multiple VROs available to serve a particular Virtual Infrastructure request, the Cloud Manager must select a VRO according to VRO's capabilities. Once the Cloud Manager has selected a VRO, a request/response protocol is required for the Cloud Manager to send the Virtual Infrastructure request to the VRO, and for the VRO to send back the resource computation response. The request is analyzed by the PCE/RCE by using current TED information.

3. Based on current TED information, the PCE computes traffic-engineered paths while the RCE performs VM consolidation tasks. The resource computation response is sent back to the Cloud Manager.

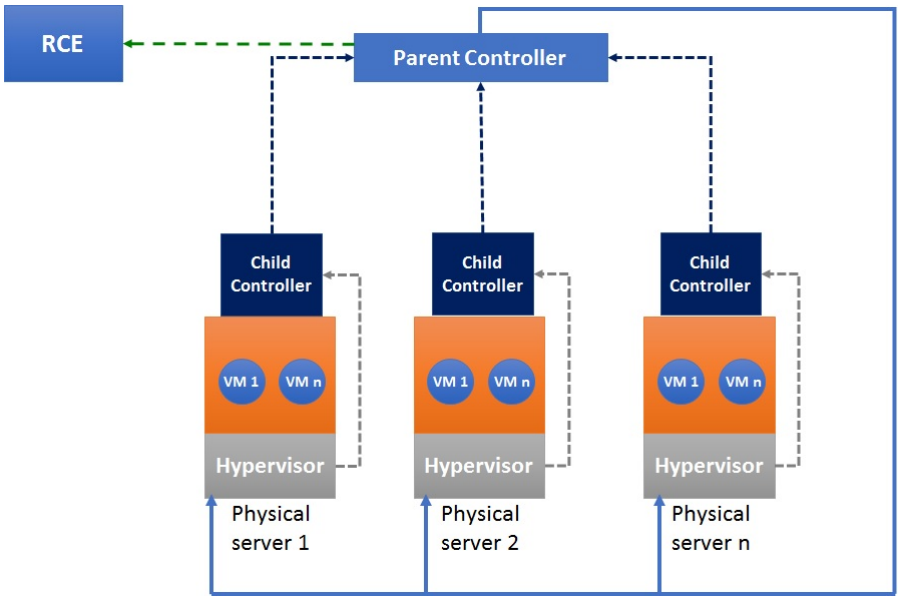


FIGURE 3.5: Computational resource orchestration

4. The PCC sends a request to the head node of the physical infrastructure for starting the resource reservation phase: the RSVP-TE protocol is responsible for setup, maintenance and tear-down of computed paths. Finally, RCE orchestrates the virtual machines through the involvement of a *Parent Controller* that resides in each data center and solves VM placement optimization (see Figure 3.5). In particular, a *Child Controller* is located on each physical server and it monitors its current utilization. The Parent Controller collects the status of individual servers from the Child Controllers and it sends data to RCE, which creates a green migration plan. Finally, the Cloud Broker gives instructions to the Parent Controller, which sends commands to hypervisors in order to migrate VMs [171].

The data considered during the resource orchestration process are

1. The resource type identifier: network or IT.
2. The power state of each resource. It is defined according to the standard approach proposed in EMAN [107], which identifies 12 power states for a device.
3. The energy consumption (expressed in kWh) of the resources.
4. The CO_2 emissions of the resources (CO_2/kWh), which depend on the energy source type (dirty/green) used for powering them.

5. The transition time matrix, which consists of the time required to transit from a resource state to another one.

6. The resource localization composed of geographic coordinates that univocally identify the position of a well-defined resource within the Cloud infrastructure (latitude and longitude).

7. The resource utilization percentage (0-100%).

3.4 Experimental Assessment

In this section, we present a case study simulation aimed at showing how the VRO can be used to take into account green management objectives in the orchestration of communication and computational resources. We envision a scenario where a number of WAN core routers of geographically distributed data centers are interconnected by means of a PCE-enabled networking infrastructure, which provides guaranteed-bandwidth paths between routers.

3.4.1 Green Resource Management Problem

The problem to be addressed is twofold: from the network-side, a green management of the geographical networking infrastructure must be pursued, in order to lower the energy consumption and the CO_2 emissions; from the IT-side, an energy efficient consolidation of virtual machines in data centers needs to be reached. The innovative idea is to provide an automatic control system for Cloud infrastructures, considered as the combination of optical networks and IT resources.

The energy-efficient management of Cloud computing environments can be pursued through green networking and virtual machine placement solutions, which might be developed both in the context of intra and inter-datacenter networking architectures. In this work, we specifically address two sub-problems: inter-datacenter green networking and intra-datacenter VM placement.

In order to assess the overall energy consumption of the physical infrastructure, we conduct a separate evaluation of both networking infrastructure and data center energy

costs. The first part of the evaluation is carried out by using MATLAB [172] and it is based on the network energy models discussed in [149] [169]; a novel traffic-engineering algorithm that minimizes energy consumption and CO_2 emissions of networks connecting multiple distant data centers is implemented. The second part of the evaluation is based on data center energy models taken from [116], and the simulations are implemented in CloudSim [140]. In this work, we use CloudSim simulator as the main platform to perform our tests. CloudSim offers several benefits ranging from simple configuration of different VM scheduling policies, to fast evaluation of efficiency, performance and reliability of several resource provisioning policies within a large heterogeneous Cloud infrastructure.

The aim is to find energy-aware scheduling solutions for Cloud data centers. Indeed, several studies demonstrate that the optimization of data center architectures, energy-aware allocations and scheduling will lead to significant energy savings.

3.4.2 A Case Study Simulation

To conduct our evaluation, we consider a test scenario consisting of

- a network infrastructure composed of eight WAN routers connected by 15 bidirectional links whose capacity is either 1-Gbps or 10-Gbps (see Figure 3.6);
- a computational infrastructure composed of 8 data centers, each one co-located with a WAN router aimed at geographically extending layer 2 networks over multiple distant data centers.

The network topology assumed in our study is comparable with those adopted by midsize Internet Service Providers for their backbone (e.g., Abilene Internet2 backbone consists of 14 links and 11 nodes). It is assumed that a service provider makes a request for an adequate number of resources to deploy healthcare applications in the Cloud. The objective is to provide consumers with enough resources in order to use the provider's applications, which are deployed on a Cloud infrastructure. In particular, we consider applications designed for specific needs of breast, prostate and liver *Magnetic Resonance Imaging* (MRI) processing and analysis; applications are placed on one server and they are accessible from various client devices through a thin client interface such as a web

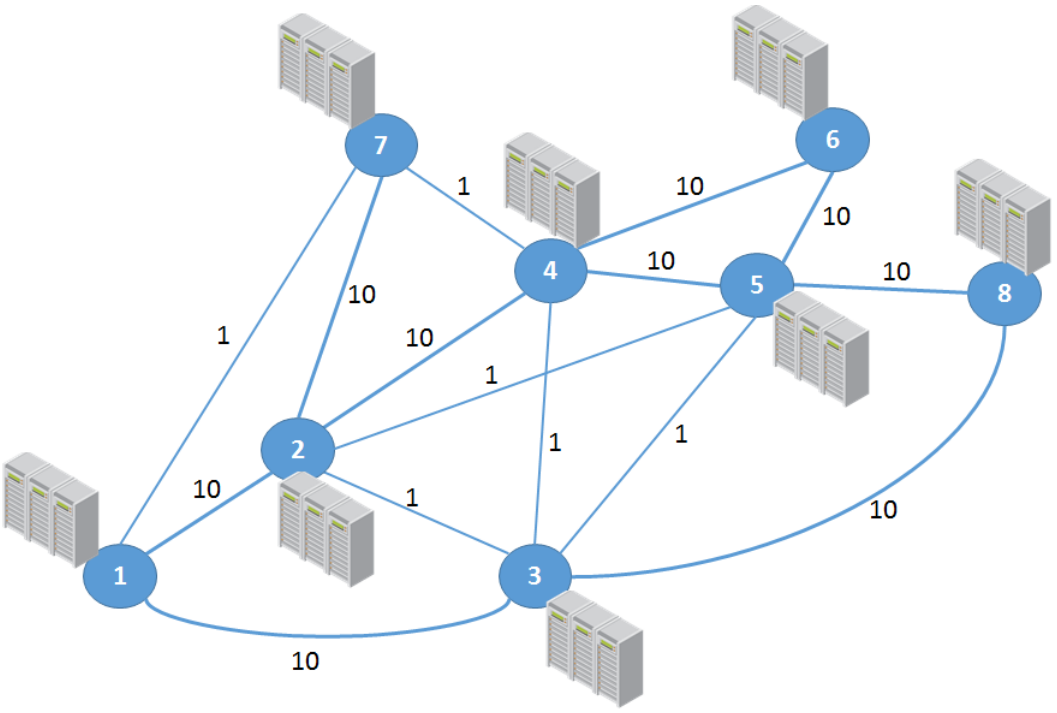


FIGURE 3.6: Use case scenario: network topology

browser. The most elementary request involves the allocation of two virtual machines, each on a different data center, so that an application in one VM is used to capture information sent from the other VM for data synchronization, providing benefits such as increased availability and a better user experience across wider geographical areas. Moreover, a network path is established between both VMs, and other pairs of virtual machines (that will be turned on later) will share it. We set three constraints:

- To satisfy SLA demands between service provider and customers, the individual request cannot be taken over by more than two data centers.
- Traffic load associated with each request is known.
- Data center's computing power is limited.

For the sake of simplicity, we assume that an individual Virtual Infrastructure consists of a variable number of virtual machine pairs, which must be deployed on two different data centers in accordance with the first above-mentioned constraint (related to SLA). The number of VM pairs is randomly selected between 2 and 14. Once activated, each pair of VMs produces an expected traffic up to 100 Mbps from a source-VM to a destination-VM. Let us clarify this last point with a simple example. A provider receives a request

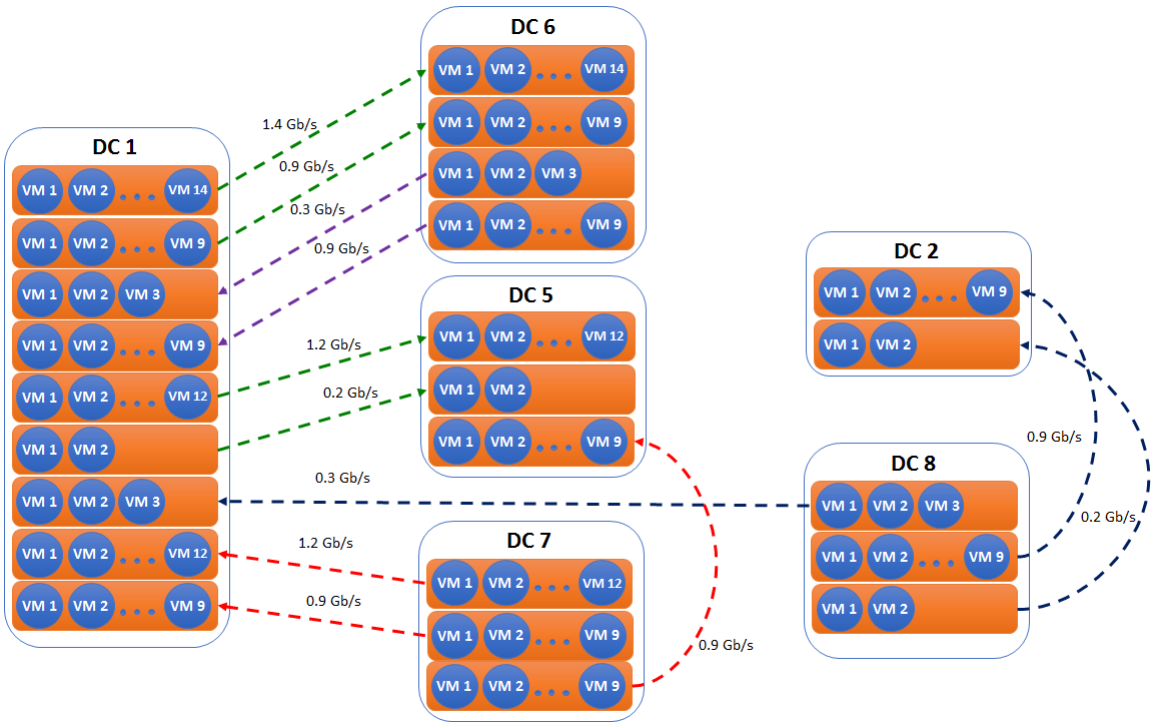


FIGURE 3.7: Traffic between VM pairs at time $t = 60'$

from a customer that implies the activation of four VMs to be satisfied; specifically, VMs will be deployed on the source data center connected to node 1. At the same time, four VMs will be deployed on the destination data center connected to node 6 for reliability purposes, and a 400-Mbps traffic volume (100 Mbps between each source-destination couple of VMs) will flow from data center 1 to data center 6. Therefore, traffic load grows proportionally to VMs deployed.

For each VI, the VRO establishes a bidirectional network path between the two data centers hosting the virtual machines. This path is assigned to a VI, hence its bandwidth is shared between all the VMs belonging to the same VI. Since the number of communicating VM pairs is between 2 and 14, the bit-rate of the links ranges from 0.2 Gbps (traffic generated by 2 VMs) to 1.4 Gbps (traffic generated by 14 VMs) (see Figure 3.7).

Each data center consists of 40 servers of two types, arranged into two racks (each rack can accommodate up to 20 dedicated servers). The internal network of each data center is organized according to a classical three-tier topology (consisting of access, aggregation and core layers). In our simulations, we assume that two access switches, two aggregation switches and one core switch are available in each data center. New power models have

been added to the CloudSim power package, concerning the following real servers used in the simulations:

- HP ProLiant MicroServer G7 N54L with AMD® Turion® II Neo N54L processor (dual-core, 2.2 GHz, 2 MB, 15 W), RAM 8 GB;
- HP ProLiant DL320e Gen8 v2 with Intel® Core™ i3 4150 (dual-core, 3.5 GHz, 3 MB, 54 W), RAM 8 GB.

Each server is equipped with 500 GB of storage and 1 Gbps of network bandwidth. The values of MIPS (*Million Instructions Per Second*), PEs (*Processing Elements*), and RAM of the VMs are chosen according to four Amazon *Elastic Compute Cloud* (EC2) instances (see Table 3.1), and they have been faithfully reproduced in CloudSim.

TABLE 3.1: Amazon EC2 Characteristics Reproduced in CloudSim

Instance Type	**vCPU**	**Memory** (GB)	**Clock Speed** (GHz)
t2.micro	1	1	2.5
t2.small	1	2	2.5
t2.medium	2	4	2.5
m3.large	2	7.5	2.5

Each VM requires 10 GB of storage and 100 Mbps of network bandwidth. The Cloud Broker, responsible for mediating negotiations between SaaS and Cloud providers, undertakes on-line negotiations to deploy a number of VMs randomly selected from 2 to 14. The Broker distributes cloudlets (tasks) among the VMs in which they will be hosted; afterwards, virtual machines will process cloudlets. The cloudlet length (number of instructions that the processor is going to execute) was set to 9×10^6 *Million Instructions* (MI); each task required 60 minutes of processor time with inter-arrival time of 5 minutes. Each VM has been randomly assigned a workload trace from the workload data.

In the simulations, we assume that 12 different VI requests are consecutively sent to the provider at 5-minute intervals. Each VI lasts 60 minutes (1 hour). Afterwards, all the resources assigned to the VI are released and the infrastructure is terminated. We

locate source-VMs in data centers connected to nodes 1, 6, 7, 8, while destination-VMs are located in data centers connected to nodes 1, 2, 5, 6. Therefore, packets are sent from nodes 1, 6, 7, 8 to nodes 1, 2, 5, 6 (see Table 3.2).

TABLE 3.2: Traffic Flow Characteristics

Start	**5'**	**10'**	**15'**	**20'**	**25'**	**30'**	**35'**	**40'**	**45'**	**50'**	**55'**	**60'**
From → to	1 → 6	1 → 6	6 → 1	6 → 1	1 → 5	1 → 5	8 → 1	8 → 2	8 → 2	7 → 1	7 → 1	7 → 5
Bit rate [Gbit/s]	1.4	0.9	0.3	0.9	1.2	0.2	0.3	0.9	0.2	1.2	0.9	0.9

Figure 3.8 and Figure 3.9 show the number of VMs running on the eight data centers of our scenario, throughout the whole simulation, assuming that all the data centers have an initial load of 10 VMs. Traffic flows last for the whole duration of a VI. Hence, the network load consists of 12 traffic flows, which start every 5 minutes and last 60 minutes. This produces an increasing network load in the first hour and a decreasing load in the second hour (see Figure 3.10).

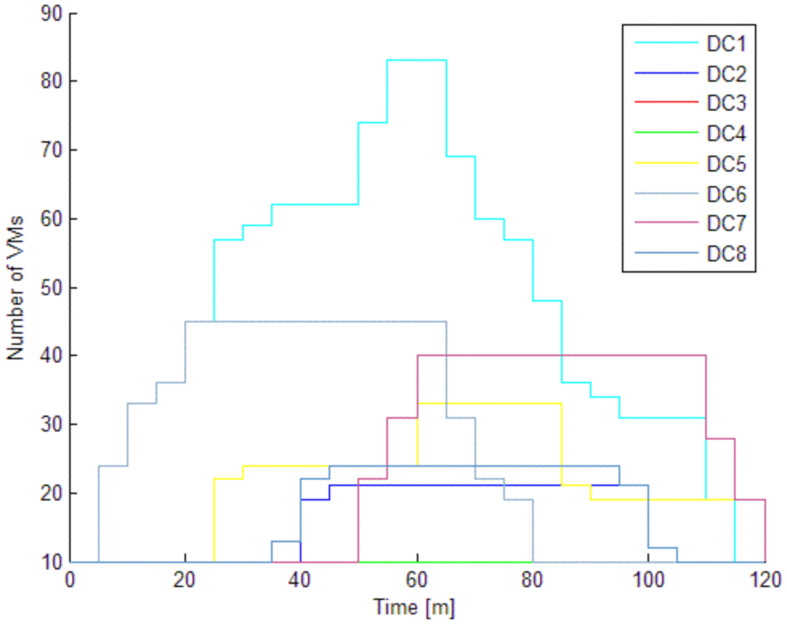


FIGURE 3.8: Number of VMs per data center during simulations

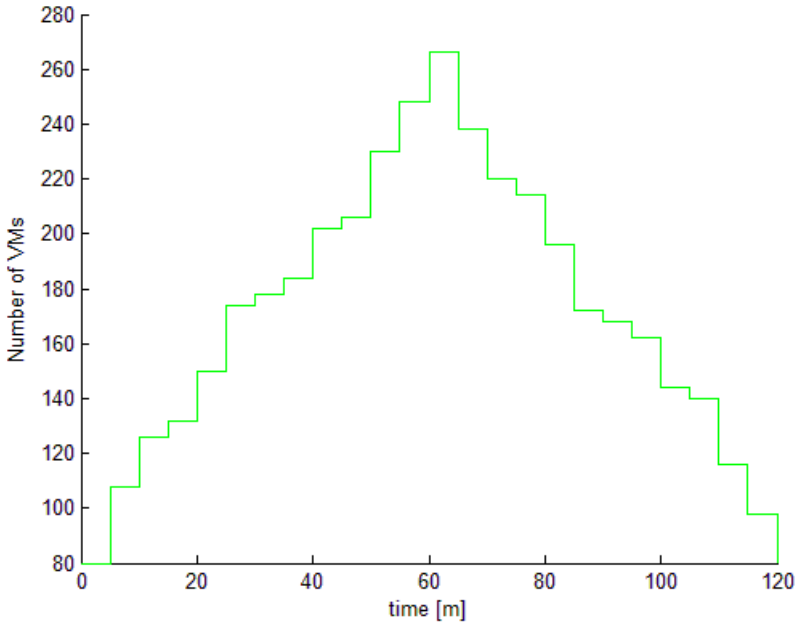


FIGURE 3.9: Total number of VMs during simulations

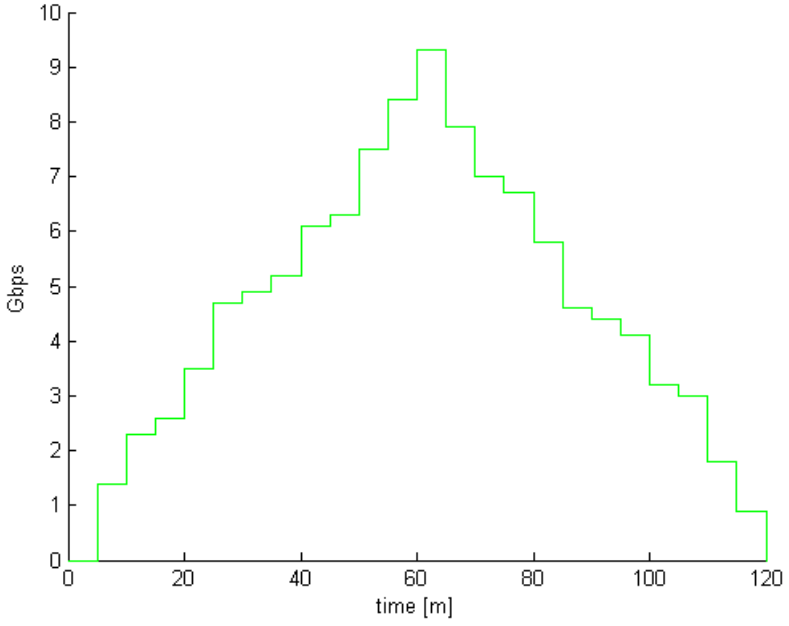


FIGURE 3.10: Overall network traffic during simulations

3.4.3 Modeling of Network Power Consumption and CO_2 Emissions

In our simulations, we consider a simple *Optical Transport Network* (OTN) that consists of eight nodes interconnected by fifteen bidirectional links whose capacity is either 1-Gbps or 10-Gbps, with a given number of wavelengths for each link. When the optical signal propagates through the fiber, it is attenuated because of absorption and scattering; moreover, signal is distorted and broadened due to dispersion mechanisms that limit fiber bandwidth. The combined effect of attenuation and dispersion makes the signal weaker and indistinguishable when it reaches its destination [173]. Therefore, a way of amplifying the optical signal when it is transmitted over long distances ($\geq$ 80 km) is needed to regenerate its strength and shape, before the *Signal to Noise Ratio* (SNR) (or, respectively, *Bit Error Rate* (BER)) becomes very small (or, respectively, high). This can be done by using several methods. At the beginning, systems used electro-optic methods, which required an optoelectronic module composed of an optical receiver, a regeneration and equalization system, and an optical transmitter. These systems are called regenerators and they require optical-electrical-optical signal conversions: the optical signal is converted to an electrical one, amplified, and then converted back to an optical signal that propagates along the fiber. The regenerators remove the noise from the signal, so as to obtain a clean one. Electro-optic methods have several drawbacks: they make the system more complicated, installation and maintenance costs are higher, and they require a separate amplifier for each WDM channel. Later, optical amplifiers were developed in order to replace the electro-optic regeneration systems. The optical amplifiers make O-E-O conversions unnecessary: signals can be amplified optically without energy dispersion. Furthermore, optical amplifiers offer better performance than regenerators. In our scenario, several optical amplifiers for each communication link are considered; even with intermediate optical amplification, the length of such an optical path is limited due to attenuation and increasing BER. As a result, optical paths exceeding a typical length of 1000 km will require a 3R regeneration in an intermediate node, where 3R regenerators re-amplify, re-shape and re-time the signal.

Moreover, we suppose that real-time information about the energy sources currently powering network devices is available. Several primary energy sources can be exploited in order to produce the electrical energy needed to power telecommunication equipment: fossil fuels (coal, oil, gas) and renewable energy (solar, wind, ocean, geothermal,

biomass). Fossil-based sources are considered dirty because when they are burned to make electricity, the carbon is released back into the atmosphere and it contributes to the greenhouse effect. The greenhouse gases such as carbon dioxide make the temperature rise, and they contribute to global warming and pollution. In 2020, it is estimated that fossil fuels will decrease from 83% to 76% of the world's energy use, because of the increasing demand for nuclear and renewable energy. Nuclear energy emits small amounts of CO_2, but it produces radioactive material. Nuclear is usually considered another non-renewable energy source because the uranium and plutonium, which are the elements most often used to fuel nuclear power plants, are non-renewable resources. Instead, the renewable energy is generated from natural sources and it is considered green and environment friendly. While the non-renewable energy sources are limited and they will soon run out, the renewable ones are constantly and sustainably replenished.

In the network model, several dirty and green energy sources that supply the network devices (nodes, optical amplifiers and 3R regenerators) are considered. Moreover, we assume that all the amplifiers on the same communication link have the same characteristics (the static part of power consumption is the same), and they are powered by the same energy source. Similar considerations apply to 3R regenerators. Therefore, each node or link is characterized by a power consumption (that changes under different loads), and it is assigned to an energy source. Hereinafter, the power consumption and CO_2 emissions of all the network components considered in our scenario are discussed.

3.4.3.1 Switch Power Consumption

A switch is characterized by a chassis, which is a frame/housing for mounting the circuit components, and several line cards that provide interfaces to the network. The power consumption of a generic switching node is dependent on its configuration and current usage: switch type, number of ports, port transmission rate and employed cabling solutions [174]. Therefore, switch power consumption can be written as [175]:

$$P_{sw} = P_{ch} + \sum_{i=1}^{conf_{lc}} \left(n_{lc}^{conf_{lc}^{i}} \cdot P_{lc}^{conf_{lc}^{i}} \right) + \sum_{j=1}^{conf_{if}} \left(n_{if}^{conf_{if}^{j}} \cdot P_{if}^{conf_{if}^{j}} \right) \tag{3.1}$$

where P_{ch} is the power consumed by the chassis, $conf_{lc}$ is the number of different line card configurations (there is a configuration for each line card rate, e.g. 1 Gbps, 10 Gbps),

$n_{lc}^{conf_{lc}^i}$ is the number of line cards that operate at a specified rate i, $P_{lc}^{conf_{lc}^i}$ is their power consumption. Instead, $conf_{if}$ is the number of different interface configurations (there is a configuration for each interface rate, e.g. 10 Mbps, 100 Mbps, 1 Gbps), $n_{if}^{conf_{if}^j}$ is the number of interfaces that operate at a specified rate j, $P_{if}^{conf_{if}^j}$ is their power consumption. In detail, for a given configuration of line cards and interfaces:

$$P_{lc} = P_{S_{lc}} + \beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}} \tag{3.2}$$

where P_{lc} is the line card consumption, $P_{S_{lc}}$ is the static part, β_{lc} is the dynamic part that scales with the rate, $currRate_{lc}$ is the current transmission rate of the line card (it is the sum of the transmission rate of all its own interfaces), and $maxRate_{lc}$ is its maximum transmission rate. Similarly:

$$P_{if} = x_{if} \cdot \left(P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}} \right) \tag{3.3}$$

where P_{if} is the interface consumption, x_{if} is a Boolean variable representing the interface state (1 if the interface is turned on, 0 otherwise), $P_{S_{if}}$ is the static part of interface power consumption, β_{if} is the dynamic part of interface power that scales with rate, $currRate_{if}$ is the current transmission rate of the interface, and $maxRate_{if}$ is its maximum transmission rate. More generally, the Equation 3.1 can be written as:

$$P_{sw} = P_{sw}^{fixed} + P_{sw}^{prop} \tag{3.4}$$

$$P_{sw}^{fixed} = P_{ch} + \sum_{i=1}^{conf_{lc}} \left(n_{lc}^{conf_{lc}^i} \cdot P_{S_{lc}}^{conf_{lc}^i} \right) + \sum_{j=1}^{conf_{if}} \sum_{k=1}^{n_{if}^{conf_{if}^j}} \left(x_{if}^k \cdot P_{S_{if}}^{conf_{if}^j} \right) \tag{3.5}$$

P_{sw}^{fixed} is the static part of power consumption and it is constant if the switch (and all its interfaces and line cards) is powered on, $P_{S_{lc}}^{conf_{lc}^i}$ is the static power of line cards with configuration $conf_{lc}^i$, k is the number of interfaces that operate at a specified rate j, x_{if}^k is a Boolean variable representing the state of each interface k that operates at a specified rate j, and $P_{S_{if}}^{conf_{if}^j}$ is the static power of the interfaces with configuration $conf_{if}^j$.

$$
\begin{aligned}
P_{sw}^{prop} &= P(\rho, c) = P(\rho) \\
&= \sum_{i=1}^{conf_{lc}} \left(n_{lc}^{conf_{lc}^i} \cdot \beta_{lc}^{conf_{lc}^i} \cdot \frac{currRate_{lc}^{conf_{lc}^i}}{maxRate_{lc}^{conf_{lc}^i}} \right) \\
&+ \sum_{j=1}^{conf_{if}} \sum_{k=1}^{n_{if}^{conf_{if}^j}} \left(x_{if}^k \cdot \beta_{if}^{conf_{if}^j} \cdot \frac{currRate_{if}^k}{maxRate_{if}^{conf_{if}^j}} \right)
\end{aligned} \tag{3.6}
$$

P_{sw}^{prop} is the dynamic part of power consumption. It depends on ρ, which is the switch utilization, and c that is a coefficient related to the power consumption overhead induced by routing protocols [176]. In our scenario, there is only a negligible increase of the power consumption when OSPF-TE packets are temporarily exchanged between routers, thus we can ignore the parameter c. Therefore, P_{sw}^{prop} scales only with the network load. Furthermore, we assume that an interface can be put into low-power mode independently from the other interfaces on the same router or line card, and only when there is not inbound and outbound traffic. Conversely, we suppose that low-power mode is not available at line card and node level: indeed, even idle nodes are characterized by a fixed power consumption.

3.4.3.2 Link Power Consumption

Each optical fiber requires optical amplifiers and 3R regenerators (see Section 3.4.3). As previously stated, we assume that all the amplifiers on the same communication link have the same characteristics (the static part of power consumption is the same); similar considerations apply to 3R regenerators. The power consumption of a link can be written as [169]:

$$
P_{link} = P_A + P_R \tag{3.7}
$$

where P_{link} is the link power consumption, P_A is the power consumption of the optical amplification, and P_R is the power consumption of the regeneration. In detail:

$$
P_A = \left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor \cdot x_A \cdot (P_A^S + \gamma_A) \tag{3.8}
$$

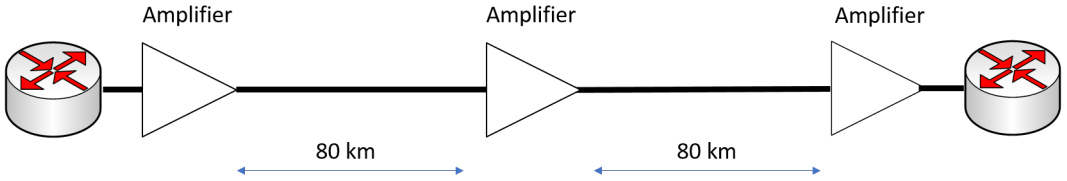


FIGURE 3.11: Optical amplifiers on a fiber

where L_{link} is the length of the link, $L_{A_{max}}$ represents the maximum distance an optical signal can cover without optical amplification ($\approx$ 80 km), x_A is a Boolean variable representing the optical amplifier state (1 if al least a traffic demand passes through it, 0 otherwise), P_A^S is the fixed power of the optical amplifier, γ_A is the proportional power of the amplifier. In our scenario, we suppose that $\gamma_A = 0$ because optical amplifiers are able to amplify simultaneously all the available wavelengths on the link, regardless of the traffic demands. A constant value (2) is added because each fiber requires at least two optical amplifiers at the end points (see Figure 3.11). Ultimately, the power consumption of an optical amplifier will be P_A^S if it is used by at least one lightpath, zero otherwise. Therefore, Equation 3.8 becomes:

$$P_A = \left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor \cdot x_A \cdot P_A^S \tag{3.9}$$

The power consumption of the 3R regeneration can be defined as:

$$P_R = \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot x_R \cdot (P_R^S + \delta_R \cdot currRate_\lambda) \tag{3.10}$$

where L_{link} is the length of the link, $L_{R_{max}}$ represents the maximum distance an optical signal can cover without regeneration ($\approx$ 1000 km), x_R is a Boolean variable representing the regenerator state (1 if the regenerator is turned on, 0 otherwise), P_R^S is the fixed power of the regenerator, δ_R is the proportional power (W/Gbps) required for each wavelength that must be regenerated on the link l, and $currRate_\lambda$ is the current transmission rate on the wavelength λ. A 3R regenerator for each wavelength of the link is required. More generally, the Equation 3.7 can be written as:

$$P_{link} = P_{link}^{fixed} + P_{link}^{prop} \tag{3.11}$$

$$P_{link}^{fixed} = P_A^{fixed} + P_R^{fixed} = \left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor \cdot (x_A \cdot P_A^S) + \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot (x_R \cdot P_R^S) \tag{3.12}$$

$$P_{link}^{prop} = P_{R}^{prop} = \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot x_R \cdot \delta_R \cdot currRate_{\lambda} \tag{3.13}$$

where P_{link}^{fixed} is the static part of link power consumption, while P_{link}^{prop} is the dynamic part.

3.4.3.3 Network Power Consumption and Emissions

By considering the above-mentioned assumptions, the overall power consumption of the network can be calculated by:

$$P_{net} = \sum_{n=0}^{N} [P_{sw}]_n + \sum_{l=0}^{L} [P_{link}]_l \tag{3.14}$$

where P_{net} is the network power consumption, N is the number of switches, $[P_{sw}]_n$ is the power consumption of the switch n, L is the number of links, and $[P_{link}]_l$ is the power consumption of the link l. In particular:

$$P_{net} = P_{net}^{fixed} + P_{net}^{prop} \tag{3.15}$$

$$\begin{aligned} P_{net}^{fixed} = & \\ & = \sum_{n=0}^{N} [P_{sw}^{fixed}]_n + \sum_{l=0}^{L} [P_{link}^{fixed}]_l \\ & = \sum_{n=0}^{N} \left\{ P_{ch} + \sum_{i=1}^{conf_{lc}} \left(n_{lc}^{conf_{lc}^i} \cdot P_{S_{lc}}^{conf_{lc}^i} \right) + \sum_{j=1}^{conf_{if}} \sum_{k=1}^{n_{if}^{conf_{if}^j}} \left(x_{if}^k \cdot P_{S_{if}}^{conf_{if}^j} \right) \right\}_n \\ & + \sum_{l=0}^{L} \left\{ \left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor \cdot \left(x_A \cdot P_A^S \right) + \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot \left(x_R \cdot P_R^S \right) \right\}_l \end{aligned} \tag{3.16}$$

$$
\begin{aligned}
P_{net}^{prop} = & \\
& = \sum_{n=0}^{N} [P_{sw}^{prop}]_n + \sum_{l=0}^{L} [P_{link}^{prop}]_l \\
& = \sum_{n=0}^{N} \left\{ \sum_{i=1}^{conf_{lc}} \left(n_{lc}^{conf_{lc}^i} \cdot \beta_{lc}^{conf_{lc}^i} \cdot \frac{currRate_{lc}^{conf_{lc}^i}}{maxRate_{lc}^{conf_{lc}^i}} \right) \right. \\
& \left. + \sum_{j=1}^{conf_{if}} \sum_{k=1}^{n_{if}^{conf_{if}^j}} \left(x_{if}^k \cdot \beta_{if}^{conf_{if}^j} \cdot \frac{currRate_{if}^k}{maxRate_{if}^{conf_{if}^j}} \right) \right\}_n \\
& + \sum_{l=0}^{L} \left\{ \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot x_R \cdot \delta_R \cdot currRate_\lambda \right\}_l
\end{aligned} \tag{3.17}
$$

Therefore, the overall network power consumption can be spit into the static part P_{net}^{fixed}, and the dynamic part P_{net}^{prop}.

Finally, the overall carbon footprint of the network during T hours can be calculated as:

$$
\Gamma_{net} = \int_0^T \left[\left(\sum_{n=0}^{N} \varphi_n \cdot [P_{sw}]_n \right) + \left(\sum_{l=0}^{L} \varphi_l \cdot [P_{link}]_l \right) \right] dt \tag{3.18}
$$

where Γ_{net} is the total carbon footprint (g CO_2), φ_n represents the mean CO_2 emissions of the switch n (g CO_2/kWh), and φ_l represents the mean CO_2 emissions of the link l (g CO_2/kWh). The possible values of φ_n and φ_l are shown in Table 3.3 [169].

TABLE 3.3: CO_2 Emissions

Energy Source	**Renewable**	**Mean Value (g CO_2/kWh)**
Solar, Wind, Hydro-electric	Yes	0
Nuclear	No	20
Geothermal	Yes	107
Biomasses	Yes	180
Natural gas	No	370
Fuel	No	880
Coal	No	980

3.4.4 Cost Function

Routing is implemented in the proposed network graph by using a constrained shortest-path Dijkstra's algorithm. Our goal is to take routing decisions in order to minimize both power consumption and CO_2 emissions, by also taking into account both minimum guaranteed bandwidth and maximum activation time for the switch interfaces (QoS constraints). The time required for on/off switching is often ignored. We claim that complex network infrastructures require an energy-aware control plane, which should be able to quickly reconfigure the network by taking into account both energy efficiency objectives and QoS constraints.

Routes are selected by considering a weighting function c_{ij} that is associated with each link (i, j), and it represents the link cost from the point of view of both its CO_2 emissions and power consumption. Therefore, link cost is evaluated as follows:

$$c_{ij} = c_{ij}^{P} + c_{ij}^{\Omega} + c_{ij}^{UB} + c_{ij}^{T} \tag{3.19}$$

where c_{ij} is the edge cost function of the link (i, j), c_{ij}^{P} is the power cost function of the link, c_{ij}^{Ω} is the carbon cost function of the link, while c_{ij}^{UB} and c_{ij}^{T} are respectively available bandwidth cost and interface activation time cost on link (i, j). In particular:

$$c_{ij}^{UB} = \begin{cases} 0 & UB_{ij} \geq B_{min} \\ \infty & \text{else} \end{cases} \tag{3.20}$$

$$c_{ij}^{T} = \begin{cases} 0 & T_{ij}^{ON} \leq T_{MAX}^{ON} \\ \infty & \text{else} \end{cases} \tag{3.21}$$

where:

- $UB_{i,j}$: available bandwidth of the link (i, j);
- B_{min}: minimum guaranteed bandwidth of the link (i, j);
- $T_{i,j}^{ON}$: time required to activate the interfaces connected to link (i, j);
- T_{MAX}^{ON}: maximum activation time allowed for the path.

If an edge has available bandwidth lower than the minimum guaranteed bandwidth, or the time required to activate its interfaces is greater than the maximum time allowed, the link must have an infinite cost and will be removed from the network. Therefore, the edge cost function given by Equation 3.19 can be defined as a linear combination of power consumption and CO_2 emissions:

$$c_{ij} = \alpha \cdot c_{ij}^{P} + (1 - \alpha) \cdot c_{ij}^{\Omega} \tag{3.22}$$

where c_{ij}^{P} and c_{ij}^{Ω} are normalized in order to be compared. The parameter $\alpha \in [0, 1]$ allows to assign different weights to the power efficiency and carbon cost functions, according to the provider's aims. In this way, two different objectives are considered together by assigning them a specific weight, according to their influence. The first term c_{ij}^{P} associated to the power consumption of the link (i, j) can be defined as:

$$c_{ij}^{P} = P_i + P_j + P_{ij}^{A} + P_{ij}^{R} \tag{3.23}$$

where P_i and P_j represent respectively the power consumption increase of the interfaces i and j, because in Dijkstra's algorithm the per-link cost function (also called incremental function) is used to increase the cost of the arc. Instead, P_{ij}^{A} and P_{ij}^{R} are respectively the incremental per-link power-related costs of the optical amplification and 3R regeneration on the link (i, j). The power consumption increase associated with the interface i of a line card n is obtained by summing two contributions (see Equation 3.2 and Equation 3.3):

$$P_i = \left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}}\right]_i + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}}\right]_n \tag{3.24}$$

where x_{if} is a Boolean variable representing the interface state (1 if the interface is turned on, 0 otherwise), and $P_{S_{if}}$ is the constant amount of power that is required to turn on the interface i. This term is taken into account only if the interface i had been previously put into low-power mode, otherwise $P_{S_{if}}$ is equal to 0. β_{if} is the dynamic amount of interface power that scales with rate, $currRate_{if}$ is the current transmission rate of the interface i, and $maxRate_{if}$ is its maximum transmission rate. Similarly, β_{lc} is the dynamic amount of line card power that scales with rate, $currRate_{lc}$ is the current transmission rate of the line card n, and $maxRate_{lc}$ is its maximum transmission rate. $P_{S_{lc}}$ is the constant amount of power that is required to turn on the line card n, and it is equal to zero because we suppose that low-power mode is not available at line card

level, therefore working line cards are never turned off. The power consumption increase related to the optical amplification is (see Equation 3.9)

$$P_{ij}^{A} = \left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor \cdot \overline{x_A} \cdot P_A^S \tag{3.25}$$

where $\left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor$ is the number of amplifiers on link (i, j), x_A is a Boolean variable representing the optical amplifier state (1 if al least a traffic demand passes through it, 0 otherwise), P_A^S is the fixed power of the optical amplifier. P_{ij}^A is taken into account once for each link, that is when the first wavelength of the link (i, j) is allocated to a lightpath and optical amplifiers have to be activated for the first time, otherwise its value will be zero in the computation of the power increase because amplifiers have been already turned on. The power consumption increase related to the 3R regeneration is:

$$P_{ij}^{R} = \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot (x_R \cdot P_R^S + \delta_R \cdot currRate_\lambda) \tag{3.26}$$

where $\left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor$ is the number of regenerators on link (i, j), x_R is a Boolean variable representing the regenerator state (1 if the regenerator is turned on, 0 otherwise), P_R^S is the fixed power of the regenerator, δ_R is the proportional power (W/Gbps) required for each wavelength that must be regenerated on the link l, and $currRate_\lambda$ is the current transmission rate on the wavelength λ. Therefore, the incremental per-link power-related cost of the regeneration is different from the optical amplification one, because 3R regenerators operate for single wavelength. Equation 3.23 can be written as:

$$\begin{aligned} c_{ij}^{P} = & \left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}} \right]_i + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}} \right]_n \\ & + \left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}} \right]_j + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}} \right]_m \\ & + \left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor \cdot \left(\overline{x_A} \cdot P_A^S \right) + \left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor \cdot (x_R \cdot P_R^S + \delta_R \cdot currRate_\lambda) \end{aligned} \tag{3.27}$$

Finally, the carbon cost function c_{ij}^{Ω} of the link (i, j) can be defined as:

$$c_{ij}^{\Omega} = \Omega_i + \Omega_j + \Omega_A + \Omega_R \tag{3.28}$$

where $\Omega_i, \Omega_j, \Omega_A, \Omega_R$ are respectively the carbon footprint increases (g CO_2) of interfaces i and j, optical amplifiers and 3R regenerators on link (i, j). In detail:

$$\Omega_i = E_i \cdot \varphi_i = \int_0^T \left(\left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}} \right]_i + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}} \right]_n \right) \cdot \varphi_i \; dt \tag{3.29}$$

$$\Omega_j = E_j \cdot \varphi_j = \int_0^T \left(\left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}} \right]_j + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}} \right]_m \right) \cdot \varphi_j \; dt \tag{3.30}$$

$$\Omega_A = E_A \cdot \varphi_A = \int_0^T \left(\sum_{a=1}^{\left\lfloor \frac{L_{link}}{L_{A_{max}}} + 2 \right\rfloor} \left(\overline{x_A} \cdot P_A^S \cdot \varphi_a \right) \right) dt \tag{3.31}$$

$$\Omega_R = E_R \cdot \varphi_R = \int_0^T \left(\sum_{r=1}^{\left\lfloor \frac{L_{link}}{L_{R_{max}}} \right\rfloor} \left[(x_R \cdot P_R^S + \delta_R \cdot currRate_\lambda) \cdot \varphi_r \right] \right) dt \tag{3.32}$$

where E_i, E_j, E_A, E_R are respectively the energy consumption increases (kWh) of interfaces i and j, optical amplifiers and regenerators, which can be calculated by integrating the aforementioned power consumption increases. Moreover, $\varphi_i, \varphi_j, \varphi_A, \varphi_R$ are respectively the mean CO_2 emissions (g CO_2/kWh) of interfaces i and j, optical amplifiers and regenerators on link (i, j). φ_a, φ_r are respectively the mean CO_2 emissions of amplifier a and regenerator r. The constrained shortest-path Dijkstra's algorithm runs in real-time by using the information contained in the Traffic Engineering Database (TED), and by considering the locally specified constraints on the link attributes; each time a VI request with QoS requirements is submitted to the provider and a traffic flow is successfully routed in the network between two nodes, the network status (decreased residual bandwidth and interface activation time) is updated and it is flooded between the neighbours by means of the OSPF-TE extension discussed in Section 3.3.2. Once the TE LSAs are flooded throughout the network, the aforementioned edge cost function is recomputed. Moreover, the edge cost functions of different links change according to power consumption and CO_2 emissions. The routing algorithm uses link weights to compute the best path, in order to minimize power consumption or carbon footprint, or their combination. The cost of a path P from a source node S to a destination node D is the sum of the costs of all its edges:

$$C_{sd} = \sum_{(i,j) \in P} c_{ij} \tag{3.33}$$

3.4.5 Energy-aware Network Management Results

The aforementioned energy model and the routing algorithm have been implemented in MATLAB, and the simulations have been performed on the case study infrastructure described in Section 3.4.2 (see Figure 3.6). This infrastructure is composed of eight core routers of geographically distributed data centers interconnected by 15 bidirectional links whose capacity is either 1-Gbps or 10-Gbps. Optical amplifiers and 3R regenerators are located on each link at every 80 km and 1000 km intervals respectively; moreover, each link requires two additional amplifiers at the end points. To consider a significant distribution of several energy sources, we refer to the U.S. *Energy Information Administration* (EIA) Energy Mapping System[5], shown in Figure 3.12, where energy sources are located throughout the country and can be displayed according to geographic, satellite, topographical, and street map views. We suppose that the eight data centers are spread all over the U.S. and that the data centers are co-located at eight different generation sources, so that they will use power produced locally (see Figure 3.13). Data centers will be powered by locally available renewable or alternative energy sources. In our scenario, data centers are located in eight cities whose availability of local energy sources may be inferred from the aforementioned U.S. Energy Map (see Table 3.4). Therefore, to evaluate the carbon footprint of nodes and links, we refer to Table 3.4 in order to know the energy sources currently powering them, while the corresponding mean CO_2 emissions are listed in Table 3.3 (see Section 3.4.3.3). The distances between the aforementioned cities are shown in Figure 3.14, where the lengths (km) of the fifteen bidirectional links of the illustrated network topology are specified.

TABLE 3.4: Data Center Locations and Available Energy Sources

Data Center	City	Available Energy Source
1	Inyokern (California)	Geothermal
2	Nucla (Colorado)	Coal
3	Earth (Texas)	Natural gas
4	Arnold (Nebraska)	Fuel
5	Des Moines (Iowa)	Biomasses
6	Welch (Minnesota)	Nuclear
7	Hardin (Montana)	Hydro-electric
8	Munster (Indiana)	Biomasses

[5] http://www.eia.gov/state/maps.cfm/

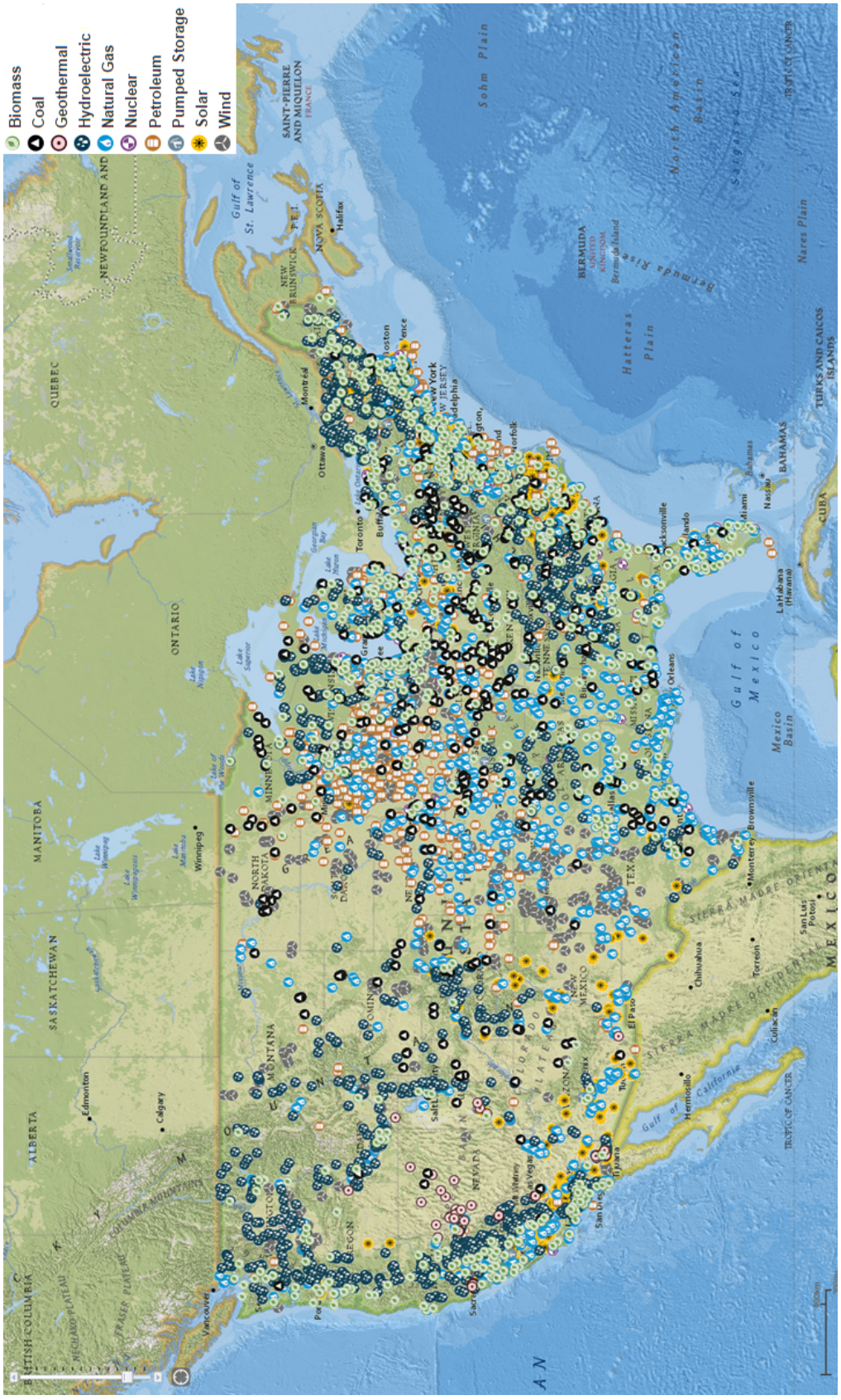


FIGURE 3.12: U.S. EIA Energy Mapping System

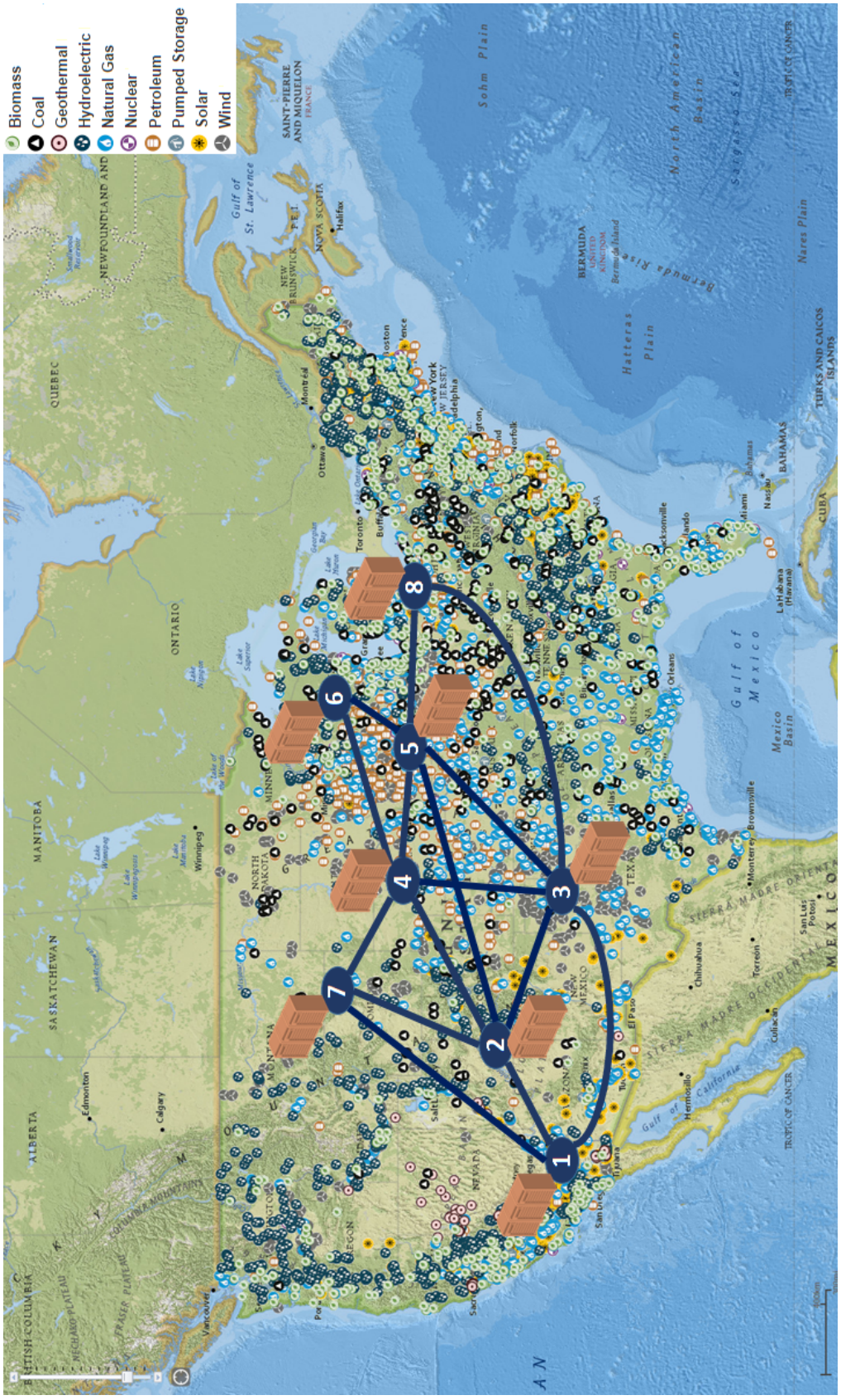


FIGURE 3.13: Data center locations

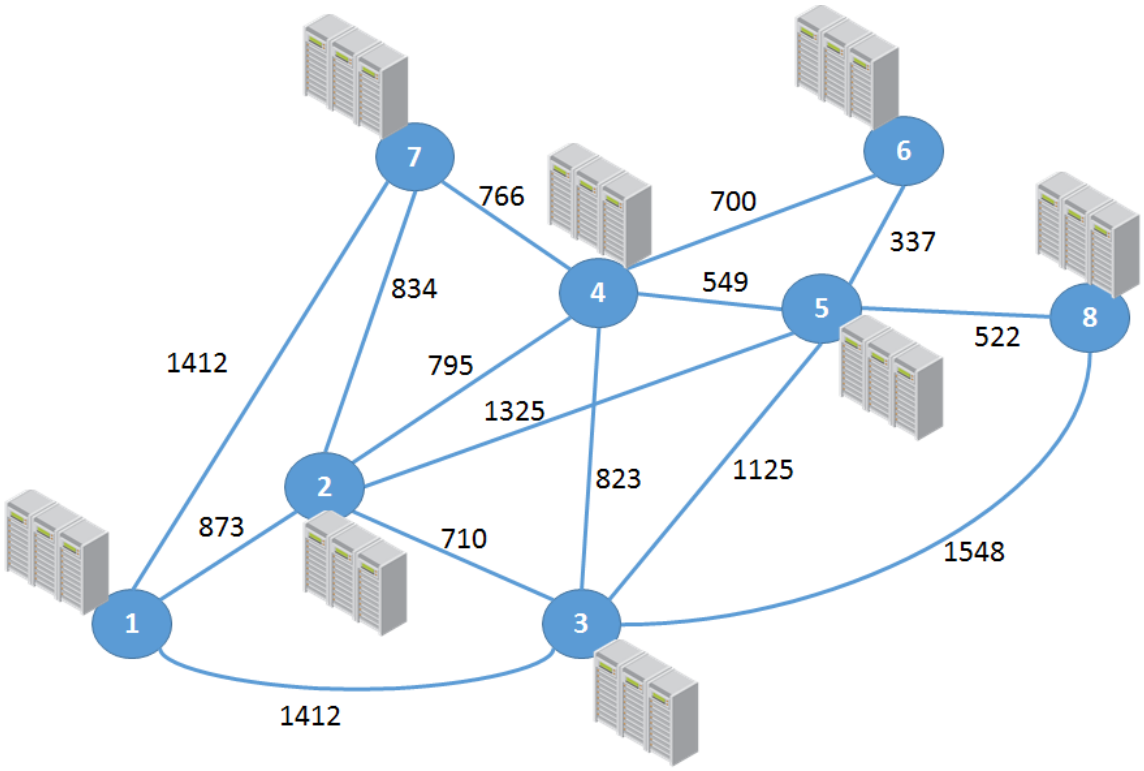


FIGURE 3.14: Length of the communication links expressed in km

To evaluate the power consumption of network nodes we refer to a simplified energy model of the Alcatel-Lucent Multiservice Switch 1850 TSS-320, shown in Table 3.5. In detail, the equipment considered for each node is shown in Table 3.6. The power demands of amplifiers and regenerators are reported in Table 3.7 [177].

TABLE 3.5: ALU-1850 TSS-320 Optical Switch

ID		**#**	**Description**	**MaxRate [Gbps]**	**Static power [W]**	**Dynamic power [W]**
	Node		TSS160C			
	Chassis				221,0	0,0
1,1	Line Card	1	2x10Gbps	20,0	30,0	20,0
1,1,1	Interface	1	SFP 1x10Gbps	10,0	20,0	10,0
1,1,2	Interface	2	SFP 1x10Gbps	10,0	20,0	10,0
1,2	Line Card	2	SFP 2x10Gbps	20,0	30,0	20,0
1,2,1	Interface	1	SFP 1x10Gbps	10,0	20,0	10,0
1,2,2	Interface	2	SFP 1x10Gbps	10,0	20,0	10,0
1,3	Line Card	1	SFP 10x1Gbps	10,0	30,0	20,0
1,3,1	Interface	1	SFP 1x1Gbps	1,0	5,0	5,0
1,3,2	Interface	2	SFP 1x10Gbps	1,0	5,0	5,0
1,3,3	Interface	3	SFP 1x1Gbps	1,0	5,0	5,0
1,3,4	Interface	4	SFP 1x10Gbps	1,0	5,0	5,0
1,3,5	Interface	5	SFP 1x1Gbps	1,0	5,0	5,0
1,3,6	Interface	6	SFP 1x10Gbps	1,0	5,0	5,0
1,3,7	Interface	7	SFP 1x1Gbps	1,0	5,0	5,0
1,3,8	Interface	8	SFP 1x10Gbps	1,0	5,0	5,0
1,3,9	Interface	9	SFP 1x1Gbps	1,0	5,0	5,0
1,3,10	Interface	10	SFP 1x10Gbps	1,0	5,0	5,0

TABLE 3.6: Nodes Equipment

Node	ID Line card	Number of Interfaces	maxRate [Gbps]
Node 1	1	1	10,0
	2	1	10,0
	3	1	1,0
Node 2	1	2	1,0
	2	2	10,0
	3	1	10,0
Node 3	1	3	1,0
	2	2	10,0
Node 4	1	2	1,0
	2	2	10,0
	3	1	10,0
Node 5	1	2	1,0
	2	2	10,0
	3	1	10,0
Node 6	1	1	10,0
	2	1	10,0
Node 7	1	2	1,0
	2	1	10,0
Node 8	1	2	10,0

TABLE 3.7: Power Consumption of Optical Amplifiers and 3R Regenerators

Device	Fixed power(W)	Proportional power (W/Gbps)
Optical amplifier	15	0
3R regenerator	285	3

The constrained shortest-path Dijkstra's algorithm computes network paths by taking into account both the power consumption of network devices and energy sources that are used for powering them. The objective is to minimize the power consumption and CO_2 emissions within the whole network. This is a multi-objective optimization problem in which two distinct objective functions are considered, namely the power consumption and the *Greenhouse Gas* (GHG) emissions, which are combined into a single total cost function. As shown in Equation 3.22, the edge cost function can be expressed as a linear combination of two link costs related to the objective functions, which are properly weighted. The weights can be tuned in order to reach the desired trade-off. At the beginning, one optimization objective is considered at a time, in order to evaluate separately the contribution of the individual objective functions to the multi-objective optimization problem, without taking into account trade-off situations. In detail, two extreme cases are discussed: the optimization of both power consumption ($\alpha = 1$) and carbon footprint ($\alpha = 0$). These limiting cases allow to obtain the lower bound of the objective function values, when the objective functions do not affect each other. Hereinafter, the results of the proposed energy model are discussed, with focus on the power consumption, GHG emissions, and average path length (hop count) resulting from the

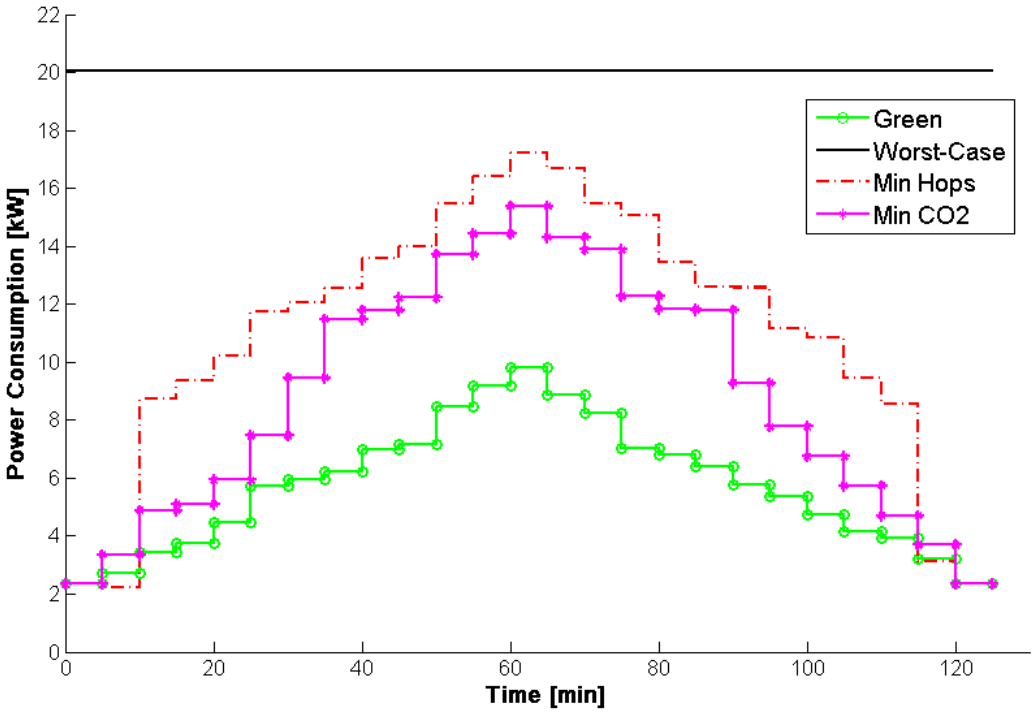


FIGURE 3.15: Power consumption for the Green, Min CO2, Min Hops, Worst-Case approaches

Green Routing Algorithm (Green, $\alpha = 1$) and *Minimum Emission Algorithm* (Min CO2, $\alpha = 0$). The results are compared to those obtained by using the *Shortest Path Algorithm* (Min Hops), and to the *Worst-Case*. *Green* and *Min CO2* minimize power consumption and GHG emissions respectively, by taking into account both minimum guaranteed bandwidth and maximum activation time for the switch interfaces (QoS constraints). Instead, *Min Hops* selects the shortest path between source and destination by minimizing the number of hops; moreover, *Min Hops* only considers the bandwidth constraint, assuming that each switch port can wake up quickly. Finally, the *Worst-Case* represents the case of today's networks in which devices are constantly powered on at full capacity but they are highly under-utilized most of the time. The *Min Hops* and *Worst-Case* approaches do not take into account energy-related parameters when computing network paths.

In Figure 3.15, we compare the overall power consumption resulting from the *Green*, *Min CO2*, *Min Hops*, and *Worst-Case* approaches, under the same load conditions. As expected, the lowest power consumption is achieved by the *Green* strategy, in which the cost function depends solely on the power consumption increases of network devices.

The energy savings achieved by using the *Green* approach are 35% compared to *Min CO2*, 47% compared to *Min Hops*, and 68% with respect to the *Worst-Case*. Instead, the energy savings achieved by using the *Min CO2* approach are 19% compared to *Min Hops*, and 55% with respect to the *Worst-Case*. The *Min CO2* algorithm selects longer paths in order to prefer nodes and links that are powered by renewable energy sources; therefore, when the traffic load increases, the power consumption resulting from the *Min CO2* strategy rises more quickly compared to the *Green* algorithm, because there will not be enough green energy sources, and the *Min CO2* is forced to select paths that are longer and more energy intensive. Indeed, when $0 \leqslant T \leqslant 30$ *min*. the energy savings achieved by the *Min CO2* and *Green* approaches are quite similar, instead when $30 \leqslant T \leqslant 60$ *min*. their distance grows. The *Min Hops* approach is more power consuming than *Green* and *Min CO2* algorithms for the entire duration of the simulations (except for $T \leq 10$ *min*, $T \geq 115$ *min*), demonstrating that energy-aware algorithms are able to reduce power consumption even at high traffic loads. As a general trend, the higher the bit-rate, the higher the power consumption; this happens because when traffic flows require less bandwidth, they can be routed over more energy-efficient paths, while flows that are more bandwidth-demanding must be routed over high capacity paths, so that the degrees of freedom are lower.

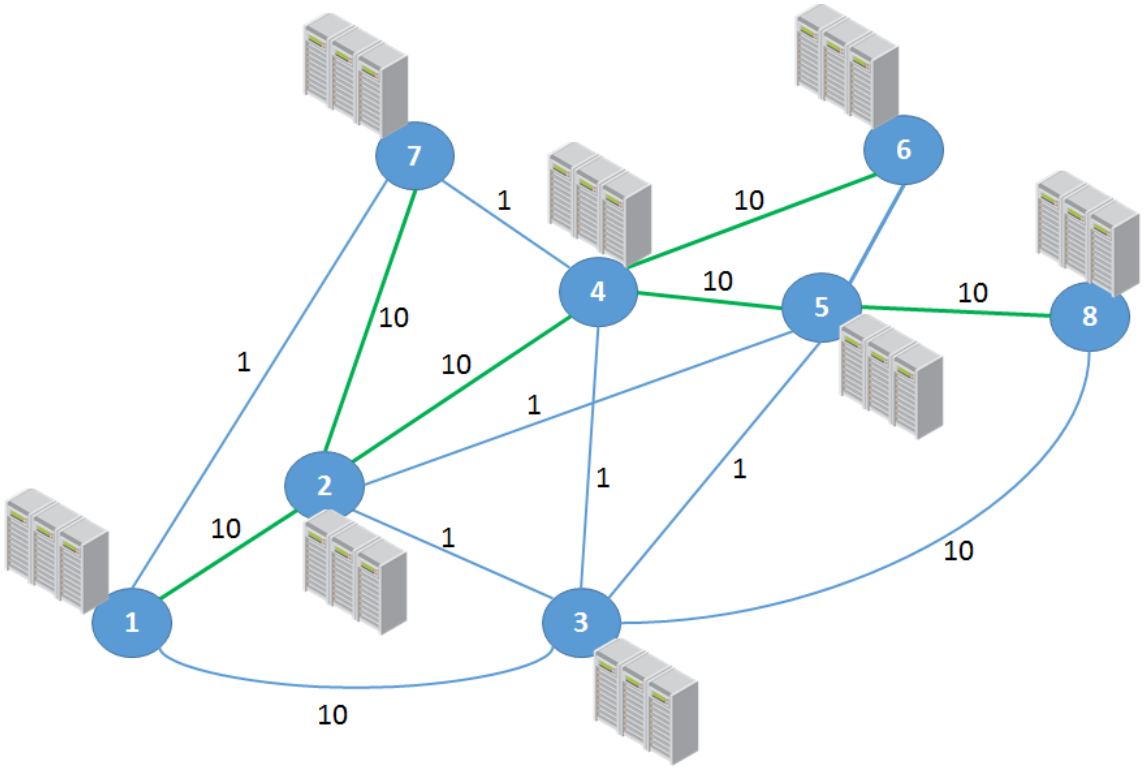


FIGURE 3.16: Source to destination paths chosen by green routing algorithm

Figure 3.16 shows the set of links used by source-to-destination paths that are chosen by the *Green* algorithm to process the twelve traffic demands: this set is made up of fewer links than those selected by the *Min Hops* approach (in which source-to-destination paths use all links). High path redundancy and low link utilization provide unique opportunities for power-aware traffic engineering. When there are multiple paths between the same source-destination pairs, and the traffic volume on some paths is low, it is possible to concentrate traffic on a lower number of paths. Switches having idle interfaces can turn them off for energy savings.

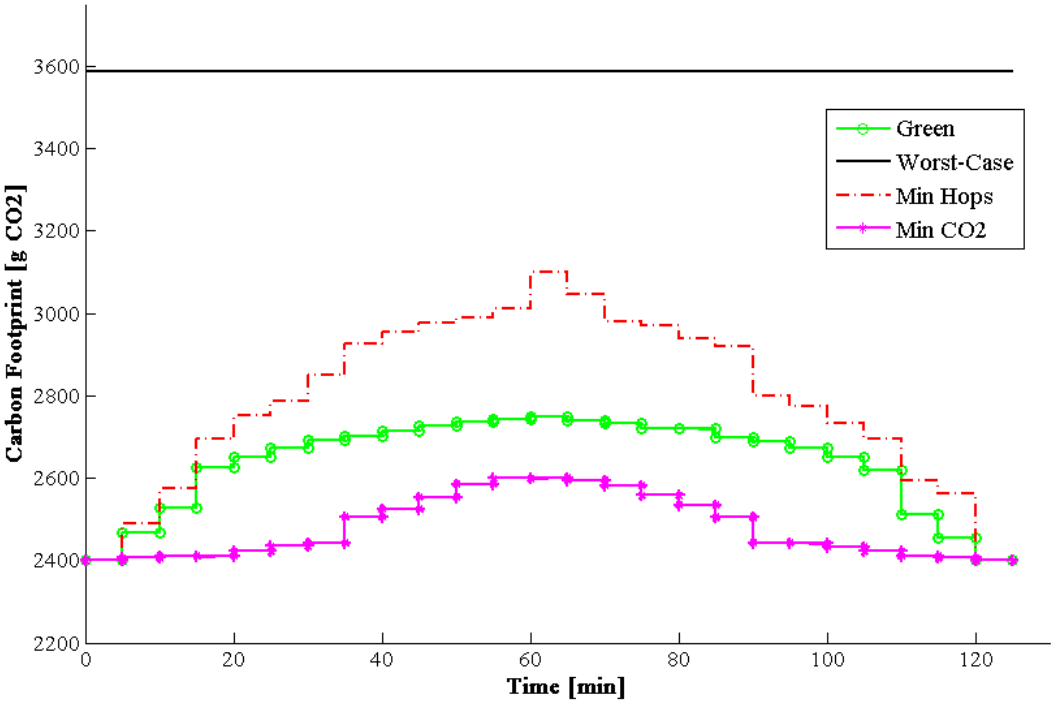


FIGURE 3.17: GHG emissions for the Green, Min CO2, Min Hops, Worst-Case approaches

In Figure 3.17, the total carbon footprint resulting from the *Green*, *Min CO2*, *Min Hops*, and *Worst-Case* approaches is compared under the same load conditions. The lowest GHG emissions are achieved by the *Min CO2* strategy, in which the cost function depends entirely on carbon footprint increases of the network devices. Therefore, besides reducing power consumption, the *Min CO2* algorithm achieves remarkable CO_2 savings by choosing network nodes and links that are powered by green energy sources. The CO_2 savings achieved by using the *Min CO2* approach are 9% compared to *Green*, 14% compared to *Min Hops*, and 33% with respect to the *Worst-Case*. Instead, the

CO_2 savings achieved by using the *Green* approach are 8% compared to *Min Hops*, and 29% with respect to the *Worst-Case*. The *Min CO2* approach is more power consuming compared to the *Green* algorithm, but this power produces lower emissions, being generated by greener energy sources. When the network traffic is low, the *Min CO2* algorithm achieves remarkable CO_2 savings compared to the *Green* algorithm; this happens because *Min CO2* selects longer paths in order to prefer nodes and links that are powered by renewable energy sources. Instead, when the traffic load increases, the distance between *Min CO2* and *Green* decreases because green energy sources will not be enough anymore, and the *Min CO2* is forced to select link and nodes that are powered by dirty sources. Hence, at high loads, the power consumption minimization also implies the GHG emission reduction. The *Min Hops* approach produces more carbon dioxide than *Green* and *Min CO2* algorithms for the entire duration of the simulations, demonstrating that energy-aware algorithms are able to reduce GHG emissions even at high traffic loads.

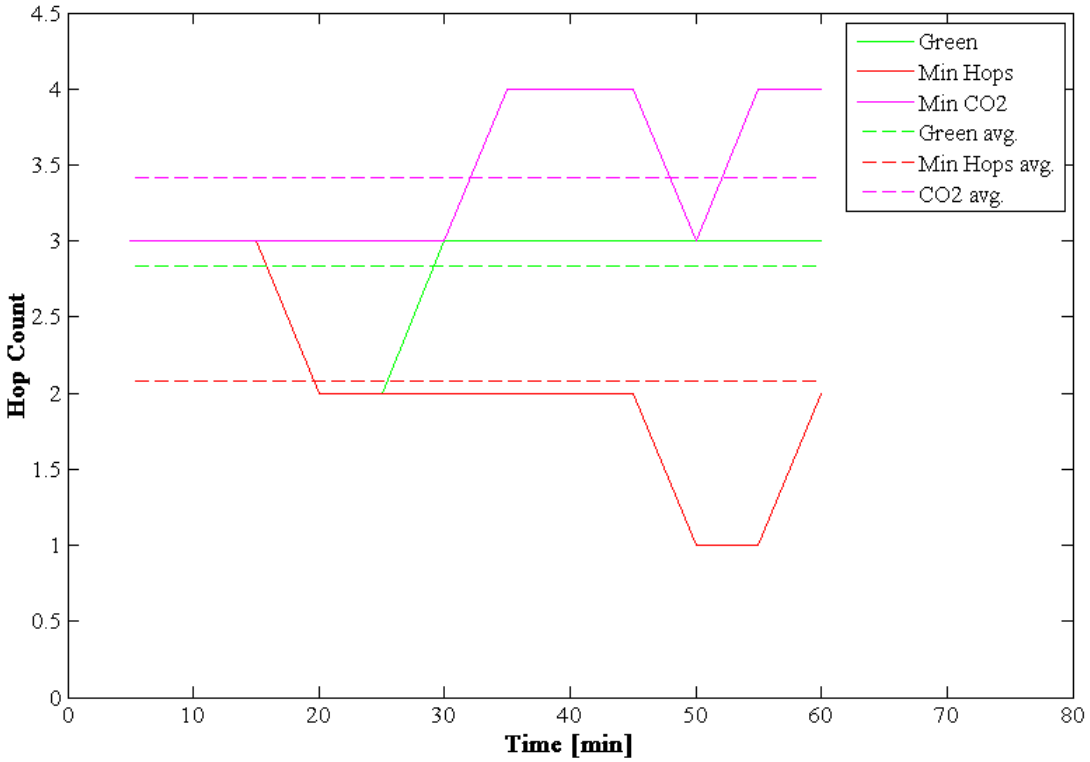


FIGURE 3.18: Number of hops for the Green, Min CO2, Min Hops approaches

Figure 3.18 shows the average path length that is achieved by using the proposed algorithms. As expected, the *Min Hops* algorithm allows to obtain the lowest hop count. The *Green* approach tries to minimize the power consumption and this involves an increase in the average path length compared to the *Min Hops* algorithm. Indeed, idle

links are avoided with respect to the busy ones, so that the average number of hops increases. Finally, the *Min CO2* strategy always tries to avoid dirty nodes while selecting longer paths than *Min Hops*; moreover, *Min CO2* chooses longer paths with respect to the *Green* algorithm too. This happens because power consumption is additive, so that for each traversed node and link the overall power consumption increases, while GHG emissions are not additive because green nodes do not produce emissions when they are traversed by network paths. Therefore, *Min CO2* always tries to avoid network devices that are powered by dirty energy sources, while increasing the average length path. The average hop count number is 2.1 for the *Min Hops* approach, while *Green* e *Min CO2* algorithms have an average path length of 2.8 e 3.4 hops respectively. Therefore, the proposed energy-aware algorithms reduce power consumption and GHG emissions, while showing only a marginal increase for the number of routing hops compared to the *Min Hops* approach.

In a multi-objective optimization problem, there are different objectives that are defined on a set of feasible decision variables, which must be properly allocated in order to provide an optimal solution for the given problem. Generally, there is not a single ideal solution that is an optimal value for every objective function at the same time. In particular, when there are several objectives, their concurrent optimization might create problems due to the conflicting relationship among the objective functions: *"There will always be a certain amount of sacrifice in one objective to achieve a certain amount of gain in the other"* [178]. Therefore, a new problem is how trade-offs between the objectives must be found, resulting in the so called *best compromise solution*. These solutions can not be found only by applying mathematical methods, but typically a *Human Decision Maker* (HDM), who is an expert in the domain of the problem, chooses the best compromise solutions from the results of a Pareto optimization, in order to achieve a trade-off between different objective functions. A *Pareto optimal solution* is one in which the value of an objective cannot be improved without degrading the value of at least one of the other objectives at the same time. Therefore, the decision maker has the task of identifying his preferred solution from a set of *Pareto optimal solutions*. In our scenario, the HDM is the network operator. He can assign different weights to the objective functions by tuning the α parameter in order to reach the desired trade-off. Hence, after evaluating the contribution of the individual objective functions to the multi-objective optimization problem, now we discuss the possible trade-offs.

Three distinct scenarios are considered, by assigning the following weights:

1. $\alpha = 0.25$ $(Green^{-}/Min\ CO2^{+})$
2. $\alpha = 0.50$ $(Green^{=}/Min\ CO2^{=})$
3. $\alpha = 0.75$ $(Green^{+}/Min\ CO2^{-})$

In the first case $(Green^{-}/Min\ CO2^{+})$, the objective of GHG minimization has a higher relative weight, while in the third case $(Green^{+}/Min\ CO2^{-})$ the power minimization objective is privileged. Finally, in the second case $(Green^{=}/Min\ CO2^{=})$ the aforementioned objectives are equally weighted in order to balance GHG emissions and power consumption. As shown in the following figures, the trade-off results are comprised between the graphs that have been previously reported for the *Green* and *Min CO2* algorithms, because limiting cases have defined the bounds of the objective function values. As expected, the lowest power consumption is achieved by $Green^{+}/Min\ CO2^{-}$, while the most power intensive strategy is $Green^{-}/Min\ CO2^{+}$ (see Figure 3.19 and Figure 3.20). In detail, the energy savings achieved by the $Green^{-}/Min\ CO2^{+}$, $Green^{=}/Min\ CO2^{=}$, and $Green^{+}/Min\ CO2^{-}$ strategies are 6%, 24% and 32% respectively, compared to the *Min CO2* approach. Regarding the GHG emissions, the lowest emitting strategy is $Green^{-}/Min\ CO2^{+}$, while the worst performance is shown by $Green^{+}/Min\ CO2^{-}$ (see Figure 3.21 and Figure 3.22).

In detail, the CO_2 savings achieved by the $Green^{+}/Min\ CO2^{-}$, $Green^{=}/Min\ CO2^{=}$, and $Green^{-}/Min\ CO2^{+}$ strategies are 2%, 4.4% and 5.7% respectively, compared to the *Green* approach. Finally, the average path length obtained by $Green^{-}/Min\ CO2^{+}$ is higher than $Green^{+}/Min\ CO2^{-}$ (see Figure 3.23 and Figure 3.24).

In detail, the $Green^{+}/Min\ CO2^{-}$, $Green^{=}/Min\ CO2^{=}$, and $Green^{-}/Min\ CO2^{+}$ have an average path length of 2.9, 3.1, and 3.3 respectively. Therefore, when the objectives are equally weighted $(Green^{=}/Min\ CO2^{=})$, the best compromise is achieved; in particular, the energy savings obtained by $Green^{=}/Min\ CO2^{=}$ are quite similar to those obtained by $Green^{+}/Min\ CO2^{-}$, and higher than $Green^{-}/Min\ CO2^{+}$. Indeed, in the $Green^{-}/Min\ CO2^{+}$ approach, the continuous selection of green nodes will lead to increased power consumption. In conclusion, the $Green^{=}/Min\ CO2^{=}$ strategy is the most tempting when an operator has to plan energy-efficient and eco-sustainable networks.

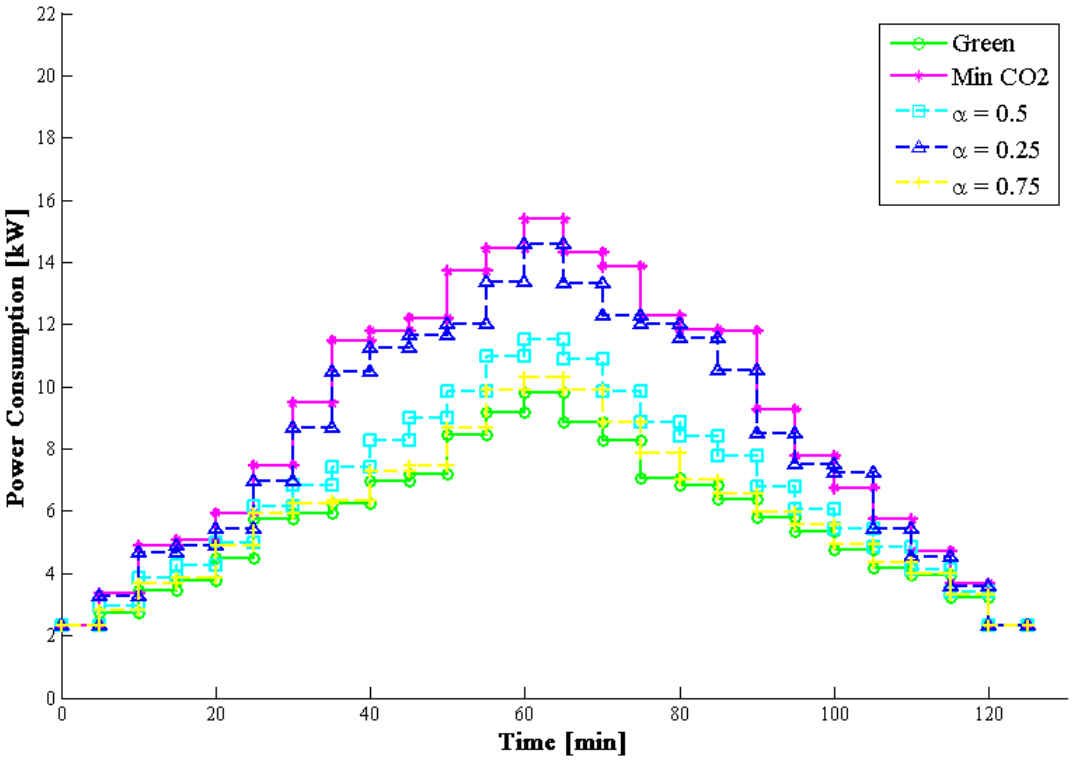


FIGURE 3.19: Power consumption for the Green, Min CO2, trade-off approaches

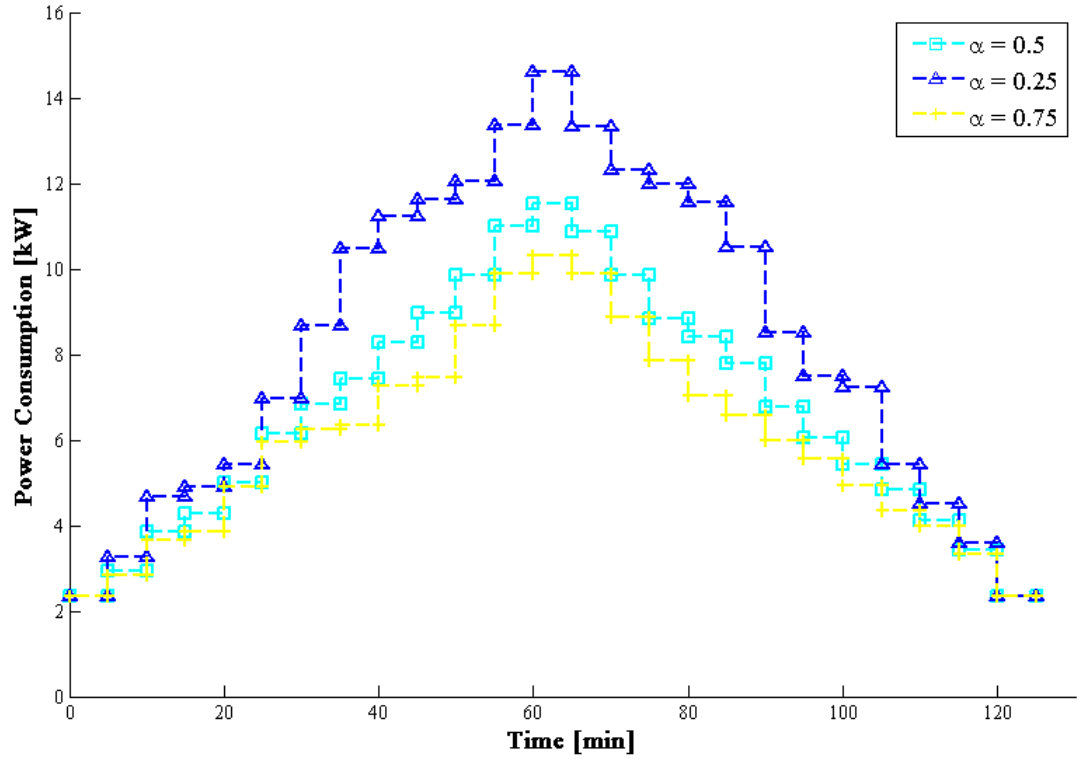


FIGURE 3.20: Power consumption for the trade-off approaches

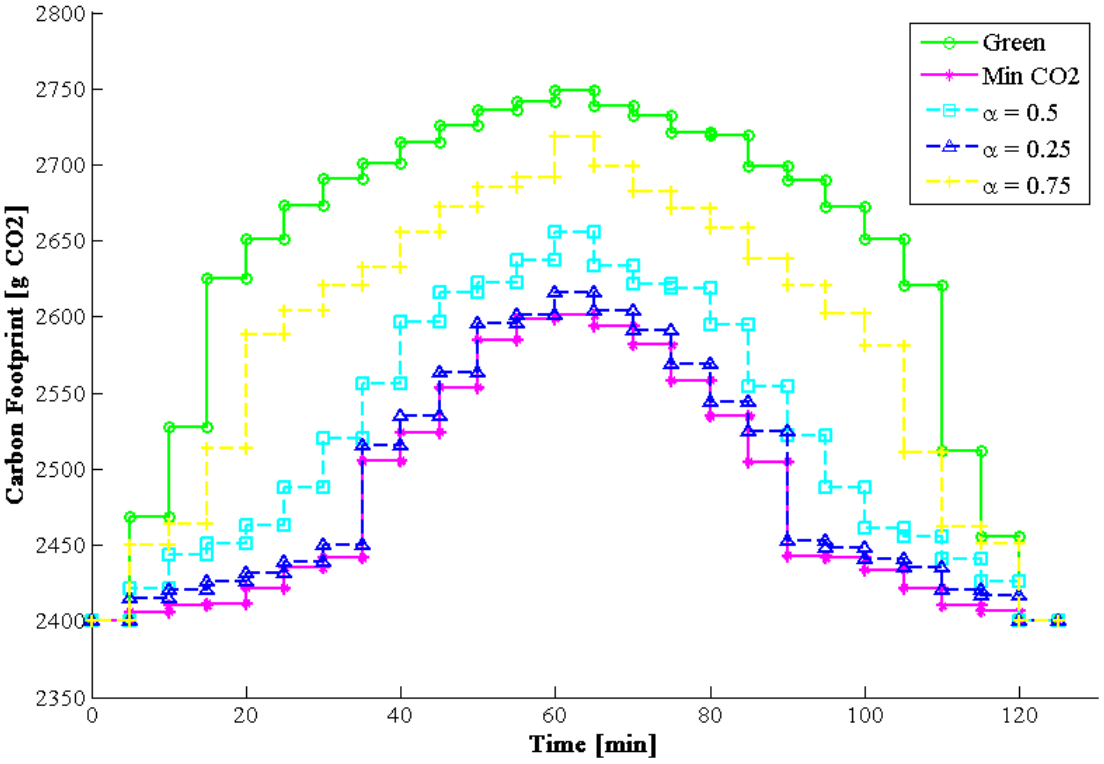


FIGURE 3.21: GHG emissions for the Green, Min CO2, trade-off approaches

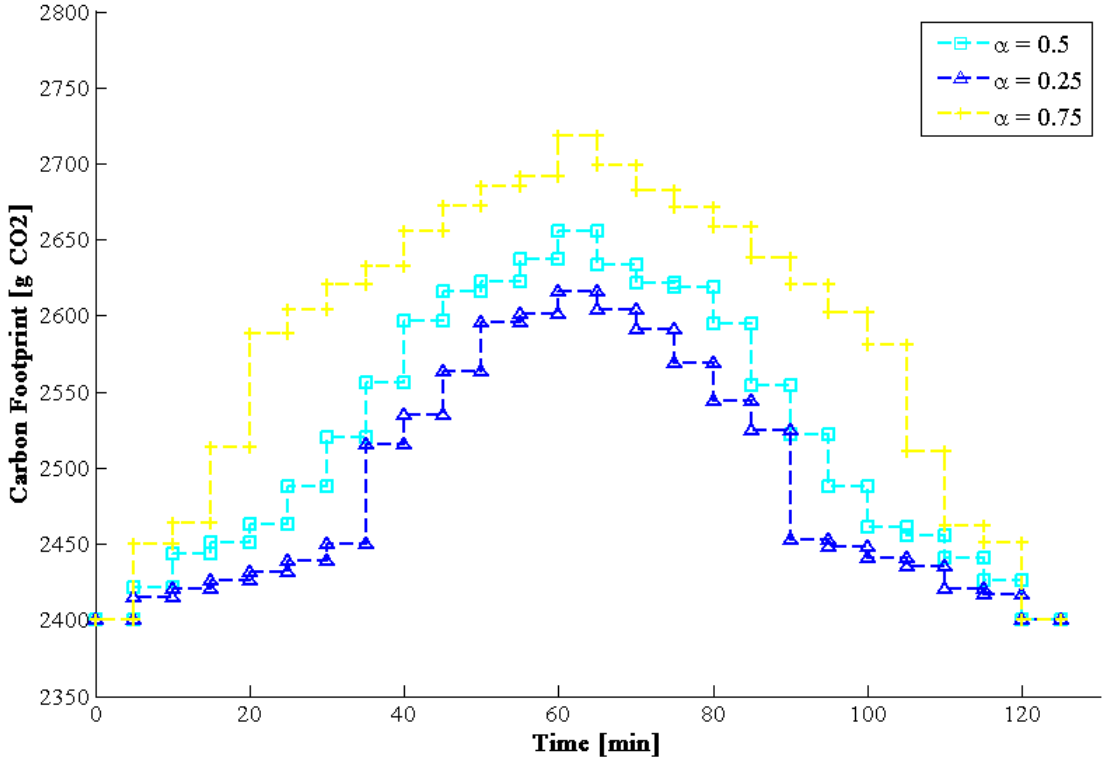


FIGURE 3.22: GHG emissions for the trade-off approaches

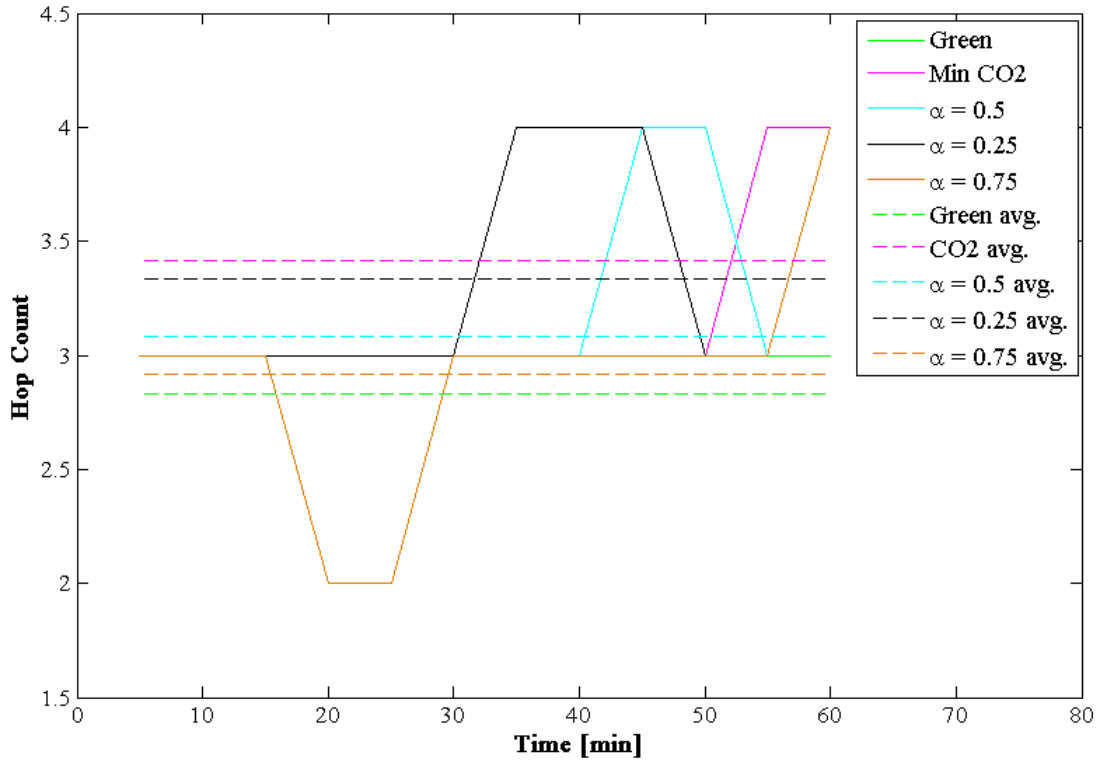


FIGURE 3.23: Number of hops for the Green, Min CO2, trade-off approaches

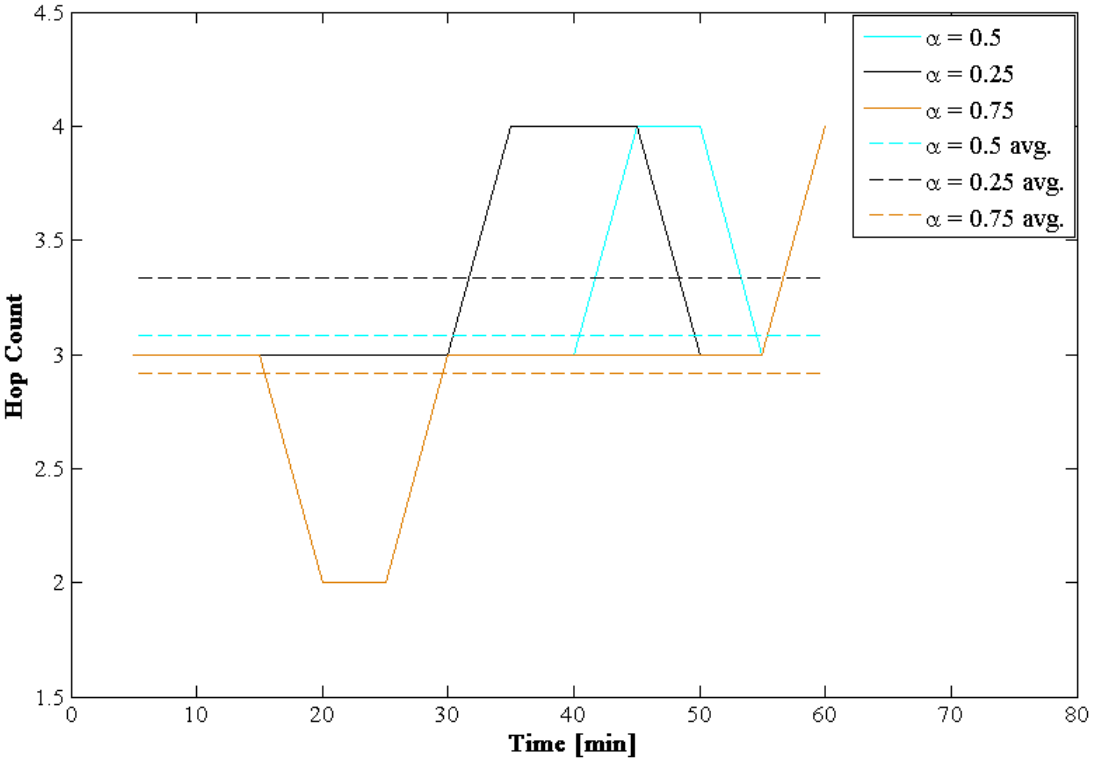


FIGURE 3.24: Number of hops for the trade-off approaches

3.4.6 Modeling of Data Center Power Consumption

To enhance energy efficiency in Cloud computing, it is important to study the distribution of power in data centers. The main source of power consumption in current data centers is their physical infrastructure (e.g. power and cooling systems), which is used to support the IT equipment (e.g. compute, storage, network). Indeed, physical infrastructure alone can amount to more then 50% of the total data center power usage. Servers consume 80% of the total IT load and 40% of total data center power consumption. Several studies identify the CPU utilization as the dominant source of energy consumption for a server [179] [166].

3.4.6.1 Server Power Consumption

According to [180] [181], the relationship between server power consumption and CPU utilization is approximately linear:

$$P_{server}[u(t)] = k \cdot P_{max} + (1 - k) \cdot P_{max} \cdot u(t) \tag{3.34}$$

where P_{max} is the maximum power consumed when the server is fully utilized, k is the fraction of power consumed by an idle server ($\approx$ 70%), and $u(t)$ is the CPU utilization that depends on time because of time-variant workloads. Instead, Fan et al. [182] proposed an empirical non-linear model of server power consumption:

$$P_{server}[u(t)] = P_{idle} + (P_{max} - P_{idle}) \cdot [2u(t) - u^r(t)] \tag{3.35}$$

where P_{idle} is the static consumption due to the elementary physical components, r is a calibration parameter that minimizes the square error. Beloglazov et al. [115] state that the proposed models can accurately predict the power consumption of a server with an error below 5% for the linear model, and an error below 1% for the empirical model. Therefore, in our simulations we consider the empirical model with the suggested value $r = 1.4$.

3.4.7 VM Consolidation Algorithms

Energy efficiency is a primary concern in data centers, which consume large amounts of electrical power leading to high costs and CO_2 emissions. Virtualization is a key component to reduce energy consumption, consolidating servers to one physical node in the form of VMs; however, this is not enough to solve energy issues, because during off-peak hours, many physical nodes run much below their average load. Several algorithms have been proposed to consolidate VMs into fewer servers, in order to switch off idle nodes and decrease energy consumption and cooling costs. Therefore, dynamic consolidation allows Cloud providers to optimize resource utilization by dynamically adjusting the number of active nodes to match resource demands, and by using live migration within a single data center, while providing required QoS for their customers.

VM allocation takes place in two steps: admission of a new VI request for VM provisioning, and dynamic VM consolidation. Dynamic consolidation of VMs consists of:

- host overload and underload detection by considering dynamic heuristics;
- VM selection, i.e. determination of VMs to be migrated from overloaded and underloaded hosts (that will be shut down);
- new placement for VMs to be migrated.

In this context, we present a comparative analysis of the energy efficient VM consolidation algorithms discussed in Chapter 2 (see Section 2.6.3). We run a set of simulations by means of CloudSim tool, and we estimate the overall data center energy consumption. We consider five methods to detect host overloading/underloading, based on the idea of setting upper and lower utilization thresholds for hosts, and keeping the total utilization of the CPU between these thresholds. The methods are:

1. **Interquartile Range** (IQR). It is a dynamic utilization threshold. It is a measure of statistical dispersion, being equal to the difference between the third and first quartiles.
2. **Local Regression** (LR). It is considered local because each smoothed value is determined by neighboring data points defined within a span. A regression weight

function is defined for the data points contained within the span, and weights are given by a tricube weight function.

3. **Local Regression Robust** (LRR). It makes LR process resistant to outliers.

4. **Median Absolute Deviation** (MAD). It is a dynamic utilization threshold. The upper threshold decreases when the CPU utilization increases, in order to prevent SLA violations.

5. **Static Threshold** (THR). This policy has a single utilization threshold that determines if a host is overloaded.

Four different VM selection policies were used:

1. **Minimum Migration Time** (MMT). This policy selects the VM that requires the minimum time to complete a migration, compared to other VMs allocated to the host.

2. **Random Selection** (RS). The VM to be migrated is chosen according to a uniformly distributed discrete random variable.

3. **Maximum Correlation** (MC). The selected VM has the highest correlation of the CPU utilization with other VMs.

4. **Minimum Utilization** (MU). The VM with the lowest CPU utilization is selected.

Finally, VM placement can be studied as a bin-packing problem, where bins are physical nodes and items are VMs [116]. In our simulations, MMT, RS, MC and MU selection policies are combined with MAD, IQR, LR, LRR and THR methods.

3.4.8 Energy-aware VM Consolidation Results

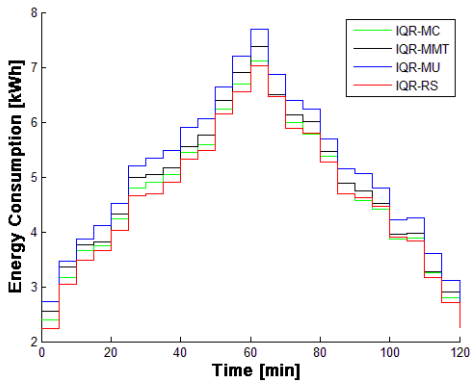
In a first set of experiments, we assumed IQR as host overloading detection policy and compared the four VM selection policies (MMT, RS, MC and MU). All the combinations will be compared in turn with *Non Power Aware* (NPA) and DVFS policies for benchmarking purposes. According to the NPA approach, all the hosts consume the maximum power throughout the whole simulation, while DVFS only applies a linear

relationship between the power consumption and the CPU utilization, without dynamic optimization of the VM allocation. Figure 3.25a shows the overall energy consumption, measured over 5-minute intervals, in the different simulated scenarios. The amount of energy consumed in each interval follows the computational load, hence it first increases and reaches its maximum after one hour, then it decreases. Nonetheless, Figure 3.25a shows that the four selection policies lead to different energy consumptions.

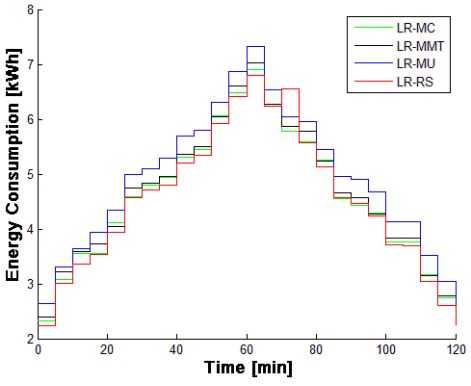
Figure 3.26 compares the different energy consumptions achieved by using the four selection policies, showing the energy consumed over 2-hours by the eight data centers; finally, the four combinations are compared in turn with NPA and DVFS. Similar simulations were run to evaluate the other four host overload/underload detection policies (LR, LRR, MAD, and THR). The results of the evaluation are shown in Figure 3.25b, Figure 3.25c, Figure 3.25d, and Figure 3.25e respectively.

Figure 3.26 reports the overall energy consumption in all the simulated scenarios. In particular, it shows that the following combinations achieve significant energy savings: LR-RS, LRR-RS, MAD-MC, and THR-MMT. In more detail, the THR-MMT policy achieves the best energy savings: 88.4% energy reduction compared to NPA policy and 36.1% energy reduction compared to DVFS policy. A comparison between non-green (NPA) approach, DVFS and all the green strategies is shown in Figure 3.27.

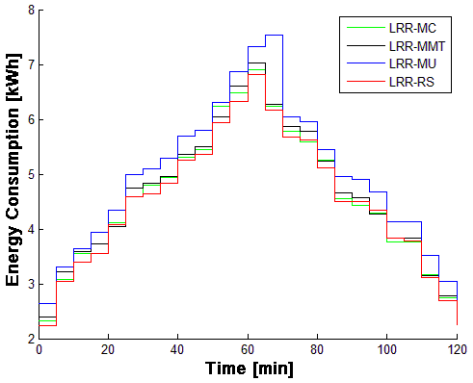
Finally, Figure 3.28 shows the comparison between non-green (NPA) and green IT strategies, where the green strategy is the result of an arithmetic mean between all the twenty combinations previously investigated. On average, dynamic consolidation policies on a large heterogeneous Cloud infrastructure guarantee 87.4% energy savings with respect to NPA, and 30.6% energy savings compared to DVFS. In conclusion, a Cloud infrastructure increases the energy consumption of the transport network, but data centers remain the most power hungry elements of a Cloud system [141]. Therefore, VM allocation and consolidation are combined in order to reach a common objective that is the minimization of the data center energy consumption. By consolidating the VMs, fewer racks and routers are employed, without compromising the SLAs previously negotiated between customers and vendor. Consequently, idle routers and cooling equipment can be turned off in order to reduce the energy consumption.



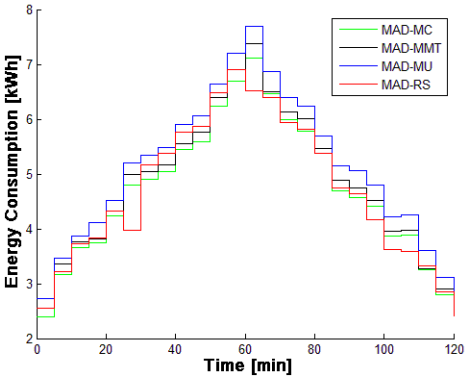
(A) IQR



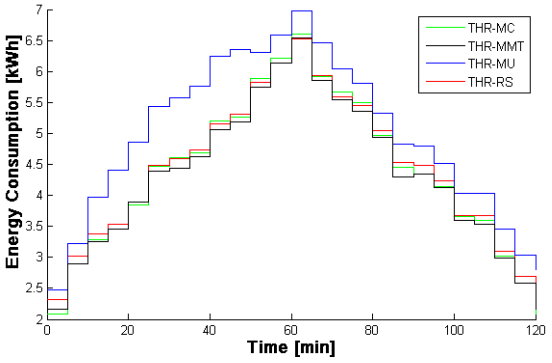
(B) LR



(C) LRR



(D) MAD



(E) THR

FIGURE 3.25: Host overloading/underloading detection policies combined with selection policies

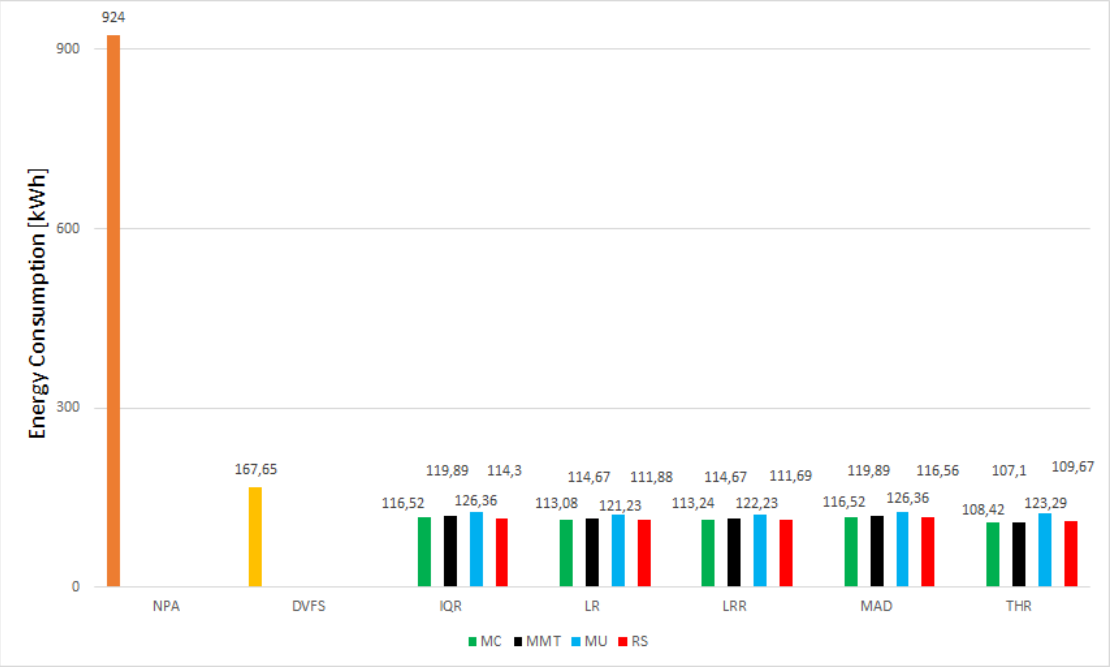


FIGURE 3.26: Energy consumption combining all detection and selection policies

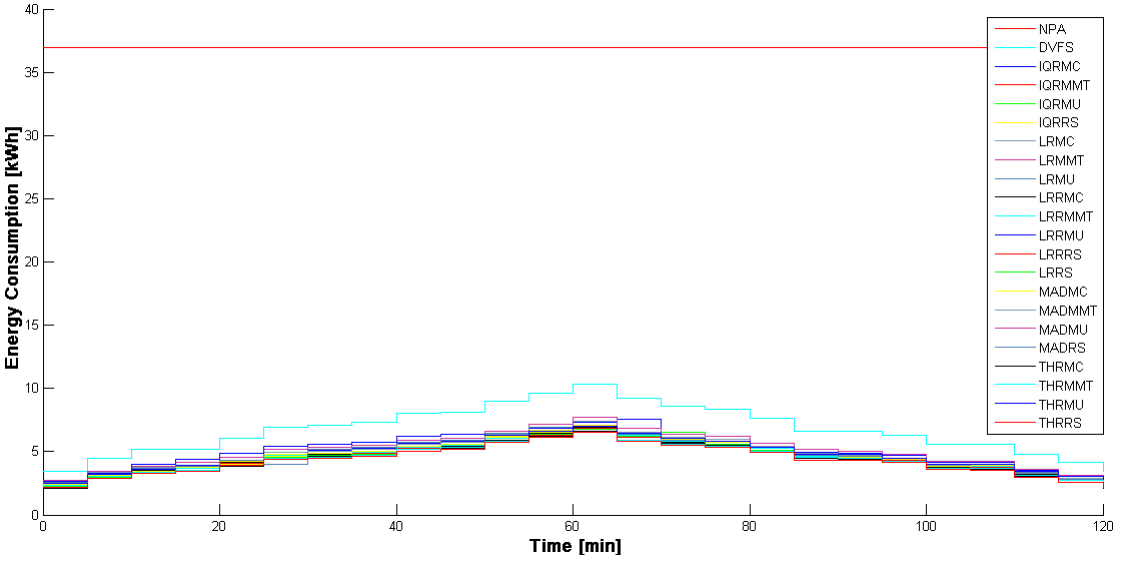


FIGURE 3.27: Comparison between non-green (NPA), DVFS and all green IT strategies

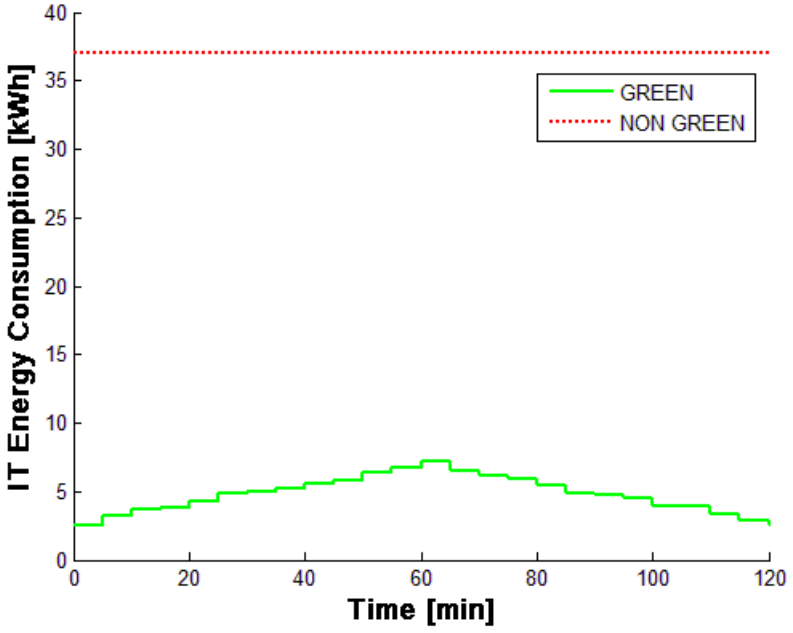


FIGURE 3.28: Comparison between non-green and green (calculated as arithmetic mean) IT strategies

3.5 Prototype Implementation

In this section, we describe the prototype implementation of the VRO architecture: we have reproduced on a testbed the network topology considered in Section 3.4.2. The testbed is implemented by means of five servers: a VMware ESXi server that is dedicated to *Network Address Translation* (NAT) and *Dynamic Host Configuration Protocol* (DHCP) services (based on pfSense), while an OpenStack environment is deployed on the other four servers. The OpenStack platform hosts ten virtual machines: eight Linux VMs are dedicated to the implementation of the virtualized network infrastructure, and the other two VMs include a software (DRAGON RCE) to compute green network paths. Below, some key features and the structure of the considered OpenStack platform and DRAGON architecture are briefly explained. Then, we present a Cloud management platform where resource management strategies are achieved by using object oriented API, so that the IaaS Cloud layer can be exploited to create virtualized testbeds. In this way, network administrators are able to manage testbed components and configure them in easy way through automatic approaches. Moreover, administrators can run testbeds for application deployment and optimization, with the objective to reproduce real operational environments in-house. Finally, the proposed platform has been experimentally evaluated.

3.5.1 OpenStack

OpenStack is an open IaaS platform. It provides a set of services that could be organized in a totally distributed and scalable manner. When this project began, in 2010, it was called OpenStack Austin and it has been characterized by several releases (Liberty is the 12th and the latest one). OpenStack proposes a series of services that use asynchronous communications to send and receive messages. The services are:

- **OpenStack Compute (Nova)** is a service that permits to create a series of virtual instances and it has the responsibility of the network configuration for each of them. The whole life-cycle of the instances within the Cloud infrastructure is controlled by Nova. Nova is agnostic to the hypervisor that guarantees the virtualization of services, interacting with it through the exposed APIs. Nova has some subcomponents:

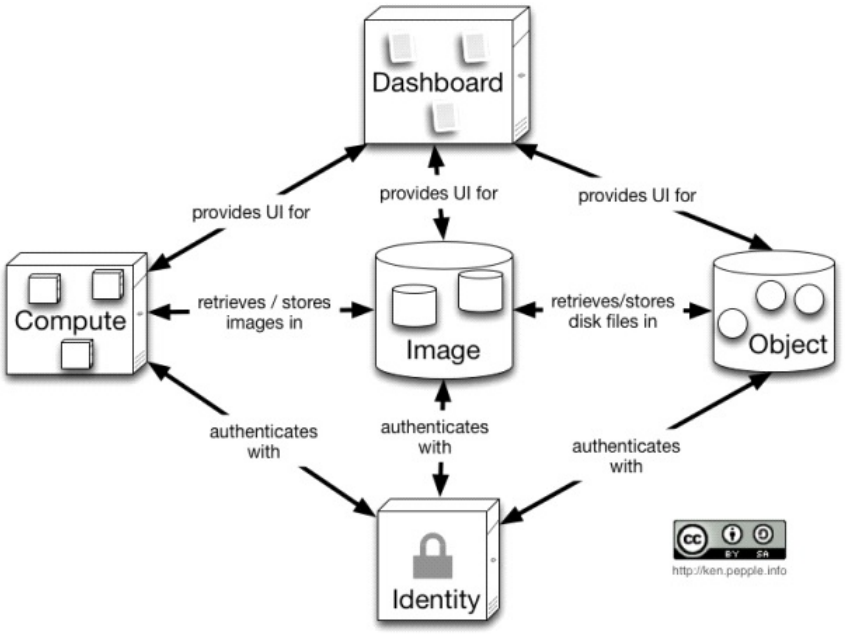


FIGURE 3.29: OpenStack Services

- *Nova-Network* manages the configuration of the virtual network of the VMs; it allocates IP addresses and it is able to create different networks (flat-based, VLAN-based, etc.).
- *Nova-API* is the interface to the outer world and it is employed by users to interact with the Cloud infrastructure. This service also enables to handle virtual appliances from Amazon, by using the *Elastic Compute Cloud* (EC2) APIs. This interface is designed as a RESTful web server, and through a message queue-based protocol it can communicate with the other components of the infrastructure. In detail, asynchronous request-response calls are employed, while callbacks are used as a trigger, which is activated when the output for a particular request is available.
- *Nova-scheduler* operates as a dispatcher of the Nova-API calls for the deployment of a VM. Through an algorithm based on configurable parameters, the scheduler chooses a compute server from a pool of available resources.
- *Nova-Compute* is a service that controls the entire life-cycle of a virtual appliance by interacting with the hypervisor.

- **OpenStack Object Storage (Swift)** is used to store objects and it provides fail-over and redundancy features. An object is a storage entity with meta-data that

describe and represent it. The files loaded in the object storage are not compressed or encrypted. Swift could be used to store and retrieve files, back up data or as an archive for the development of applications that need an integration with a storage system. The functions implemented by Swift include:

  - backup e scalability;
  - redundancy;
  - data container;
  - secure storage for a large number of objects.

  One of the fundamental components of Swift is the container that manages a series of objects stored in a particular structure. It could be considered as a folder in a file system, even if the containers cannot be nested. Before starting uploading a file, it is necessary to create a container. The object component is responsible for searching, storing and deleting objects. The users interact with the *Proxy* by using the exposed APIs. It receives requests to change meta-data, upload files or create containers. The information about the physical location of the objects stored in Swift is located in the *Ring*. Finally, the *Account* is the server that handles the list of containers.

- **OpenStack Image Service (Glance)** is another service that permits to search, store and retrieve original VM images. Glance can use a particular interface with the Amazon storage solution (S3), or it can be used with the Object Storage to stock images. In addition, Glance can employ the following methods to store images:

  - the local filesystem (default);
  - OpenStack Object Store;
  - S3 storage;
  - HTTP (read-only).

- **OpenStack Volume (Cinder)** is used to create further volume partitions, which can be added on the fly and exposed through the iSCSI protocol [183] to the VMs. These volumes are allocated as block storage resources and can be connected to the virtual machines as a secondary storage, or they can be employed as the root storage to boot instances. A snapshot could be considered as a read-only copy of

a volume at a given point in time, which can be created starting from a volume currently in use. A volume is requested via Cinder, which adds a logical volume into a specific group. Then, the volume is attached to an instance through an iSCSI session with the compute node.

- **Dashboard (Horizon)** provides users with a web-based platform to access and manage all the implemented services. It can be used to mount volumes, manage instances, manipulate containers, create a key pair etc. The main features available through Horizon are:
  - Volume Management: creating snapshots and volumes;
  - Object Store Manipulation: create, delete objects and containers;
  - Instance Management: view console logs, connect through VNC, terminate or create instances, etc.;
  - Manage users, quotas and usage for project;
  - Image Management: delete or edit images;
  - User Management: assign roles, create users etc.;
  - Access and Security Management: assign floating IPs, manage key-pairs, create security groups, etc.
- **Keystone** provides authentication and authorization processes for all the OpenStack services. Authentication establishes if a specific request is actually coming from the user who requested the service. There are two different types of authentication: the first one relies on the account (username-password), while the second one is based on token. Through the authorization procedure, an authenticated user can access to a specific service. This service also handles the user management functionalities by tracking users and what they are allowed to do. An user can be assigned different roles in different tenants, which can be considered as an organization, group or project. Keystone includes a token and a service catalogue that provides information about the endpoints of the services that can be accessed from a user.
- **Network (Neutron)** manages the creation of a network for the virtual appliances and it also could substitute Nova-network service. Its structure is modular, because Neutron allows to use a particular plugin that specifies the interface with a networking driver.

3.5.2 DRAGON

Establishing inter-domain paths is not trivial, because of the number of parameters involved, the diversity of network infrastructures and the proper configurations that must be done at both sides of the connection. Nowadays, configuring a lightpath across multiple domains is a lengthy process that needs to be automated, in order to make it scalable [184]. The *Dynamic Resource Allocation in GMPLS Optical Networks* (DRAGON) project concerns the dynamic provisioning of network resources; multi-domain TE paths are established by using a distributed control plane across heterogeneous network technologies, in response to end-user requests [185]. The main objective is to support e-Science applications with deterministic (defined and guaranteed service levels) network resources to link computational clusters, storage arrays, etc [186]. The DRAGON control plane architecture is able to provide connections on heterogeneous networks by using GMPLS, also providing wrappers for network equipment that do not support it. The key architectural components are (see Figure 3.30):

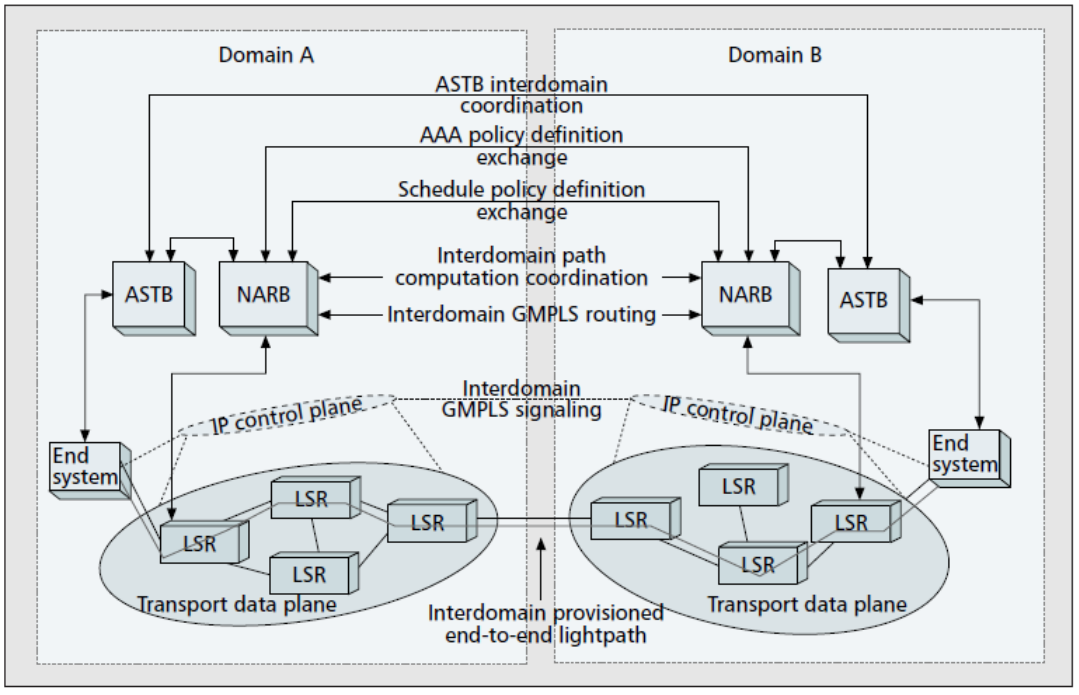


FIGURE 3.30: VRO interaction with IT resources

- **Virtual Label Switch Router (VLSR)**: in order to ensure end-to-end automated provisioning, VLSR provides a mechanism to include non-GMPLS devices that do not have native GMPLS protocols. VLSR translates GMPLS events into

device specific commands (SNMP, TL1, CLI), allowing dynamic reconfiguration of non-GMPLS equipment. A non-GMPLS device is converted to a VLSR by adding a UNIX-PC that runs a GMPLS-based control plane stack, which consists of OSPF-TE and RSVP-TE protocols. VLSR acts as a proxy agent for non-GMPLS network devices and enables the control of Ethernet, TDM and optical switches via GMPLS.

- **Network Aware Resource Broker (NARB)** and **Resource Computation Element (RCE)**: NARB enables routing, path computation and signalling across multiple different domains. It plays the part of a local Autonomous System and acts as a path computation engine that is queried by end-systems to have information about the availability of TE paths. The RCE is a stand-alone subcomponent of the NARB and it performs path computation tasks. NARB is also concerned with inter-domain routing by exchanging topology information to enable path computation. This exchange is based on the actual topology discovered by using the OSPF-TE protocol, or on an abstract view of the domain topology, which allows providers to hide the real topology of their domains (minimizing external updates). In the last case, path computation accuracy is highly reduced. The NARB makes use of algorithms to compute paths according to multiple constraints. This component first computes a path based on an abstract topology, and then a more accurate path is obtained through coordination of NARBs located in multiple domains. NARB/RCE has the following VLSR related features:
  - Dynamic resource state collection via OSPF-TE, and resource management;
  - domain topology abstraction and advertisement via OSPF-TE;
  - constraint-based path computation;
  - inter-domain routing.
- **Application Specific Topology Builder (ASTB)**: it receives requests from end-systems for establishing Application Specific Topologies (a set of LSPs related to an application domain). Moreover, the ASTB utilizes NARB services to evaluate the availability of the individual LSPs requested, mapping them to specific topologies.
- **Client System Agent (CSA)**: it is a software running on the end-system that terminates the data plane link of the provided service. In this context, a client is

any system that requests network services. In case of complicated topologies, CSA queries the local ASTB.

GMPLS TE information is advertised via OSPF-TE protocol, which sends LSAs to all the routers within the same area. Each router stores information in a local TED, learns network topology and uses *Shortest Path First* (SPF) algorithm to calculate the shortest path between that node and every other. If the TED includes constraints for *Authentication, Authorization, Accounting* (AAA) and scheduling, other than TE constraints, it is called 3D TED. The combination of path computation and 3D TED provides the 3D RCE, used by DRAGON to implement policy-based provisioning. For the GMPLS routing, the DRAGON project has extended the open-source ZEBRA software (adding TE extensions for OSPF), while for the GMPLS signalling, they have extended KOM RSVP (adding TE extensions for RSVP).

3.5.3 Network Management Application

Infrastructure providers need more powerful platforms to efficiently combine management procedures for both communication and IT resources. They have to ensure that the resources are able to scale with a growing number of participating services and that QoS requirements are met. Moreover, providers must face the problem of energy-efficient resource management in a system where the allocation decisions are subjected to energy consumption constraints. The above-mentioned characteristics can only be verified by testing the network and IT resources at runtime and in a multitude of realistic scenarios; however, resources must be deployed in these different designated environments to get meaningful test results. Unfortunately, it is often impossible to perform multiple tests at the same time during the running phase because the test infrastructure does not offer suitable resources.

To solve these issues, we have developed a framework for emulating energy-efficient network environments and for automatically deploying these in the Cloud. Network providers can customize and automatically generate running testbed instances based on the customization. By using the OpenStack APIs, we intend to automate the creation, deployment, and management processes of virtual machines in a Cloud environment, depending on Virtual Infrastructure requests issued by a Virtual Service User. OpenStack

nodes will host multiple virtual machines that can be allocated to different overlay experimental networks, according to a specific topology chosen by the user. In this way, a reliable and rapid communication over high-speed optical networks between data centers is achieved. Network providers can utilize our application program to create multiple paths with different criteria. A criteria could be a path with minimum end-to-end delay or a more energy-efficient path with increased latency. In other words, our framework can be used to generate customizable testbeds that enable simulations of GMPLS-based network scenarios, heading towards dynamic on-demand resource allocation and service provisioning. Several VLSRs, implemented by means of virtual Ethernet switches and DRAGON software, provide traffic engineered paths. On a higher plane, the proposed system provides users with a fully customizable testbed, in order to perform advanced experiments related to the distributed computing. Furthermore, by using the Cloud as a platform, we provide a cost-efficient way to setup arbitrarily large testbed instances on-demand.

A typical system usage scenario consists of three steps:

1. Users interact with a user friendly and intuitive GUI, in order to communicate with an OpenStack environment for testbed customization and network design. Two sets of JAVA APIs for graphics programming are used: AWT and Swing, which provide a huge set of reusable GUI components.

2. The system creates an infrastructure to provide and run flexible testbeds on-demand.

3. Our system deploys multiple DRAGON virtual machines according to different overlay networks, with a specific topology chosen by the user (not available with the traditional OpenStack dashboard).

The software libraries that have been used in order to develop our system are:

- **OpenStack4j**[6] **2.0.2**: an open source Cloud library that allows to exploit OpenStack APIs by using Java language, so that it is possible to interact with OpenStack services from within our system.

[6] http://www.openstack4j.com/

- **Jung**[7] **2.0.1**: a free and Java-based open source software library that is used for the analysis, manipulation, and the visualization of graphs and networks. It enables users to create directed and undirected graphs, and relations among the graph nodes.

After describing via XML the variables that affect the topology of the virtualized infrastructure, our application loads the XML topology document for the target OpenStack environment. Then, the XML is used to create configuration files that will be uploaded onto DRAGON virtual machines. The steps to be undertaken in order to obtain the above-mentioned files will now be detailed.

3.5.3.1 XML Format for Network Topology Representation

A network topology is made up of the interconnection of links and nodes:

- **Node**: virtual machine that runs on top of OpenStack. It represents a VLSR, implemented by using Ubuntu OS and DRAGON software. A GRE tunnelling mechanism is available: for each node, more tunnels can be configured. The configuration of a GRE tunnel involves creating a tunnel interface, which is a logical interface called "endpoint".
- **Link**: connection between two nodes. A tunnel interface is created in the node to support a connection. The tunnels behave as virtual point-to-point links that have two endpoints identified by the tunnel source and the tunnel destination addresses.

We have defined a customized XML format to represent the above-mentioned network topology information. The *topology* element contains information about network topology and it is composed of two sub-elements: *nodes* and *links* (see Figure 3.31). A *nodes* element contains a list of *node* elements, which have two attributes:

- **id**: a unique identifier of the node (integer).
- **type**: it identifies the role of the node in the topology (head, transit or tail). Transit nodes are dedicated to the implementation of the virtualized network infrastructure, while head and tail nodes include a special version of the DRAGON

[7] http://jung.sourceforge.net/

RCE, in order to compute green network paths and cover the VRO functionalities. Head and tail nodes are mandatory, and they are respectively connected to the source and destination (transit) nodes of the virtualized network infrastructure.

The *node* element contains the *endpoints* sub-element, which is a list of *endpoint* elements that have three attributes:

- **id**: a unique identifier of the endpoint (integer).
- **pwrCons**: power consumption of the tunnel interface.
- **status**: the power state is defined according to the standard approach that models the different power levels of a device, as proposed in EMAN [107].

```
- <topology name="myTopology">
  - <nodes>
    - <node type="head" id="0">
      - <endpoints>
          <endpoint id="1" status="high" pwrCons="140"/>
        </endpoints>
      </node>
    - <node type="transit" id="1">
      - <endpoints>
          <endpoint id="2" status="high" pwrCons="140"/>
          <endpoint id="3" status="low" pwrCons="250"/>
          <endpoint id="4" status="low" pwrCons="250"/>
        </endpoints>
      </node>
    - <node type="transit" id="2">
      - <endpoints>
          <endpoint id="5" status="high" pwrCons="250"/>
          <endpoint id="6" status="low" pwrCons="25"/>
        </endpoints>
      </node>
    + <node type="transit" id="3">
    - <node type="tail" id="4">
      - <endpoints>
          <endpoint id="5" status="high" pwrCons="98"/>
        </endpoints>
      </node>
  </nodes>
  - <links>
    - <link id="1" metric="5">
        <nodeId1>1</nodeId1>
        <nodeId2>2</nodeId2>
        <epId1>1</epId1>
        <epId2>2</epId2>
      </link>
    + <link id="2" metric="10">
    + <link id="3" metric="5">
    + <link id="4" metric="10">
    + <link id="5" metric="5">
  </links>
</topology>
```

FIGURE 3.31: Example of XML topology element

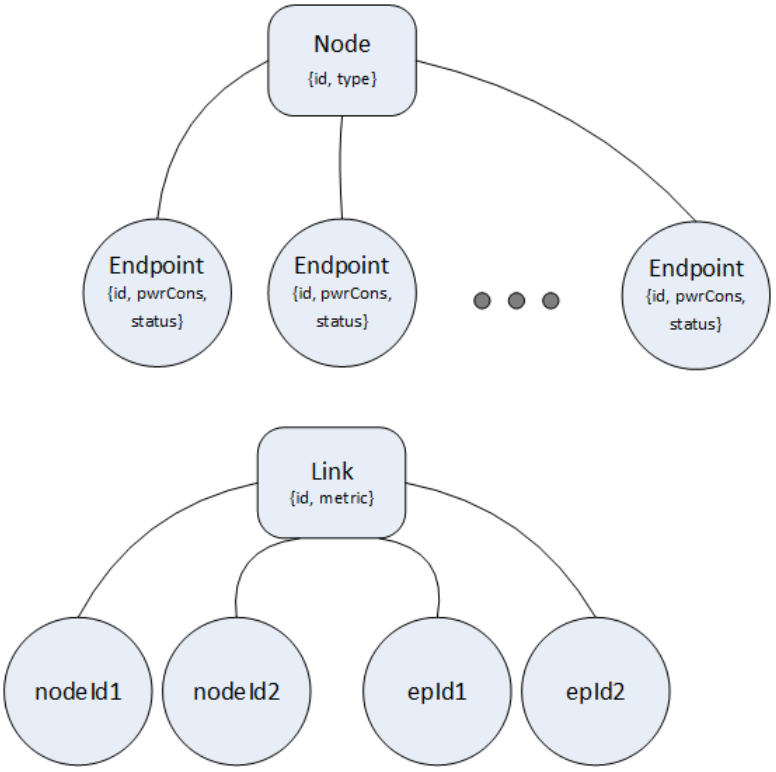


FIGURE 3.32: A taxonomy of topology elements

Finally, a *links* element is a list of *link* elements. Each *link* element contains the following attributes:

- **id**: a unique identifier of the link (integer).
- **metric**: it is the cost of an interface in OSPF. This cost indicates the overhead required to send packets across that interface. The cost of an interface is inversely proportional to the bandwidth of that interface.

A link joins two nodes and more precisely, two interfaces on two nodes. Therefore, the *link* element contains the following sub-elements: *nodeId1* and *nodeId2* identify the terminal nodes of the link, while *epId1* and *epId2* identify the terminal endpoints of the link (see Figure 3.32).

3.5.3.2 Generating DRAGON Configuration Files

We use W3C libraries to parse the XML topology file and map the elements into Java objects; by reading XML file, we also obtain an adjacency matrix that shows the connectivity of the network, and its related graph. In Figure 3.33, we present an example

of very simple network with three nodes and two links. A Java object is generated from each network element; in Figure 3.34 there is an example of Java object related to the *link* element. Each node is configured with an interface exclusively dedicated to the exchange of control plane messages (GRE interface), and another one for exchanging routing messages (LGRE interface): GRE and LGRE objects are associated with each link, as shown in Figure 3.34. Source and destination addresses of a GRE tunnel are part of the same subnet.

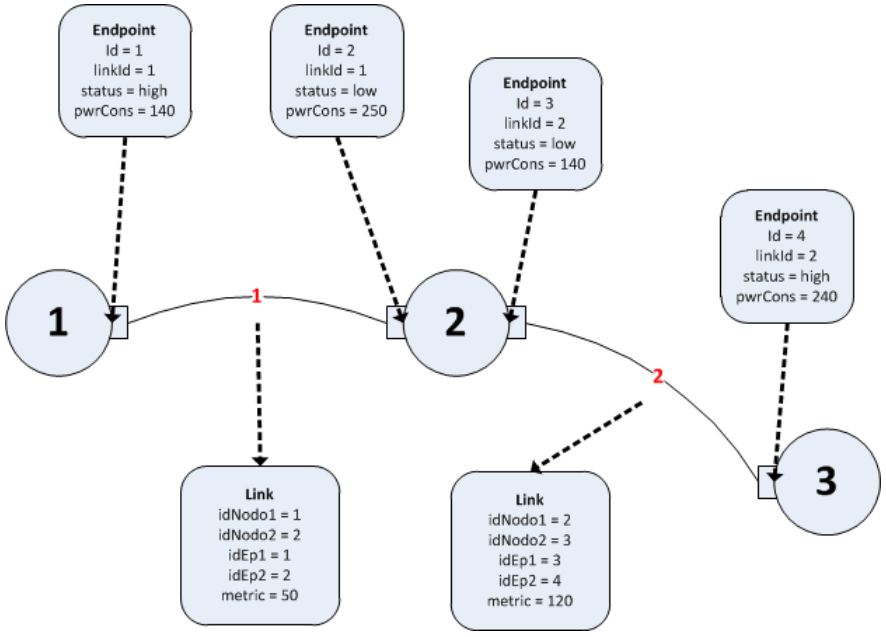


FIGURE 3.33: An example of simple topology

```
NodeList links = document.getElementsByTagName("link");

List<Link> myLinkList = new ArrayList<Link>();

for(int i=0; i<links.getLength(); i++){
    Element linkElement = (Element)links.item(i);
    int idLink = Integer.parseInt(linkElement.getAttribute("id"));
    int metric = Integer.parseInt(linkElement.getAttribute("metric"));
    int nodeId1 = Integer.parseInt(linkElement.getElementsByTagName("nodeId1").item(0).getTextContent());
    int nodeId2 = Integer.parseInt(linkElement.getElementsByTagName("nodeId2").item(0).getTextContent());
    int epId1 = Integer.parseInt(linkElement.getElementsByTagName("epId1").item(0).getTextContent());
    int epId2 = Integer.parseInt(linkElement.getElementsByTagName("epId2").item(0).getTextContent());

    String subnetGre = AddressPool.getAddressPool().getNewSubnet();
    String subnetLgre = AddressPool.getAddressPool().getNewSubnet();

    Gre gre = new Gre(idLink, epId1, epId2, subnetGre, "255.255.255.0");
    Lgre lgre = new Lgre(idLink, epId1, epId2, subnetLgre, "255.255.255.0");

    myLinkList.add(new Link(idLink, nodeId1, nodeId2, epId1, epId2, metric, gre, lgre));
}
```

FIGURE 3.34: Parsing XML topology

Finally, the following configuration files for each virtual machine are created:

- **init.sh**: this script creates connections with neighbouring virtual machines;
- **ospfd.conf**: VM credentials, router ID and its switching capabilities are defined. In addition, also power consumption information for each interface is specified (interface status, power consumption and transition time);
- **rsvpd.conf**: GRE tunnel interfaces are specified;
- **zebra.conf**: LGRE tunnel interfaces are defined.

The user interface of the application is shown in Figure 3.35: after reading the XML file, an adjacency matrix that shows the connectivity of the network and its related graph is displayed (see Figure 3.36). The "Generate Files" button allows to configure DRAGON virtual machines according to a specific topology chosen by the user.

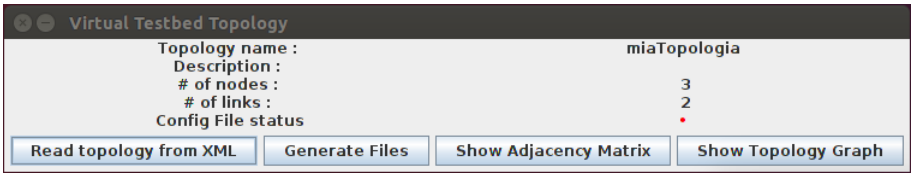


FIGURE 3.35: GUI of network management application

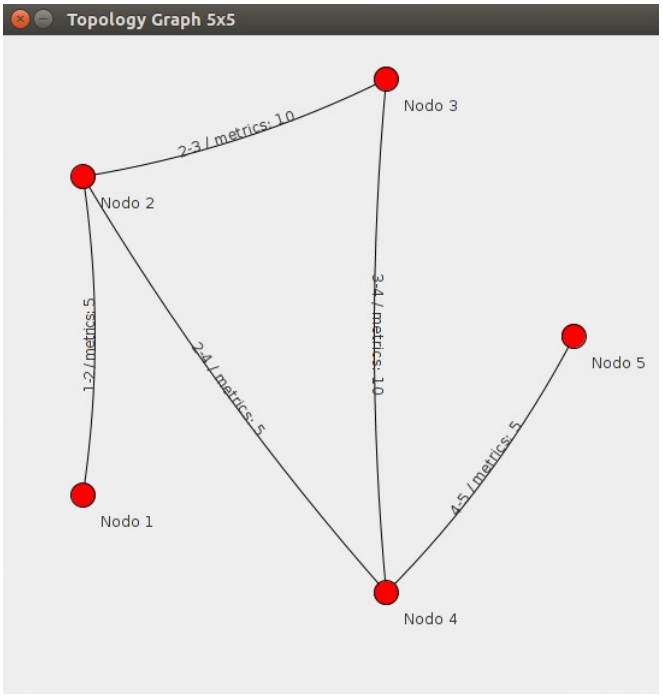


FIGURE 3.36: Network topology graph

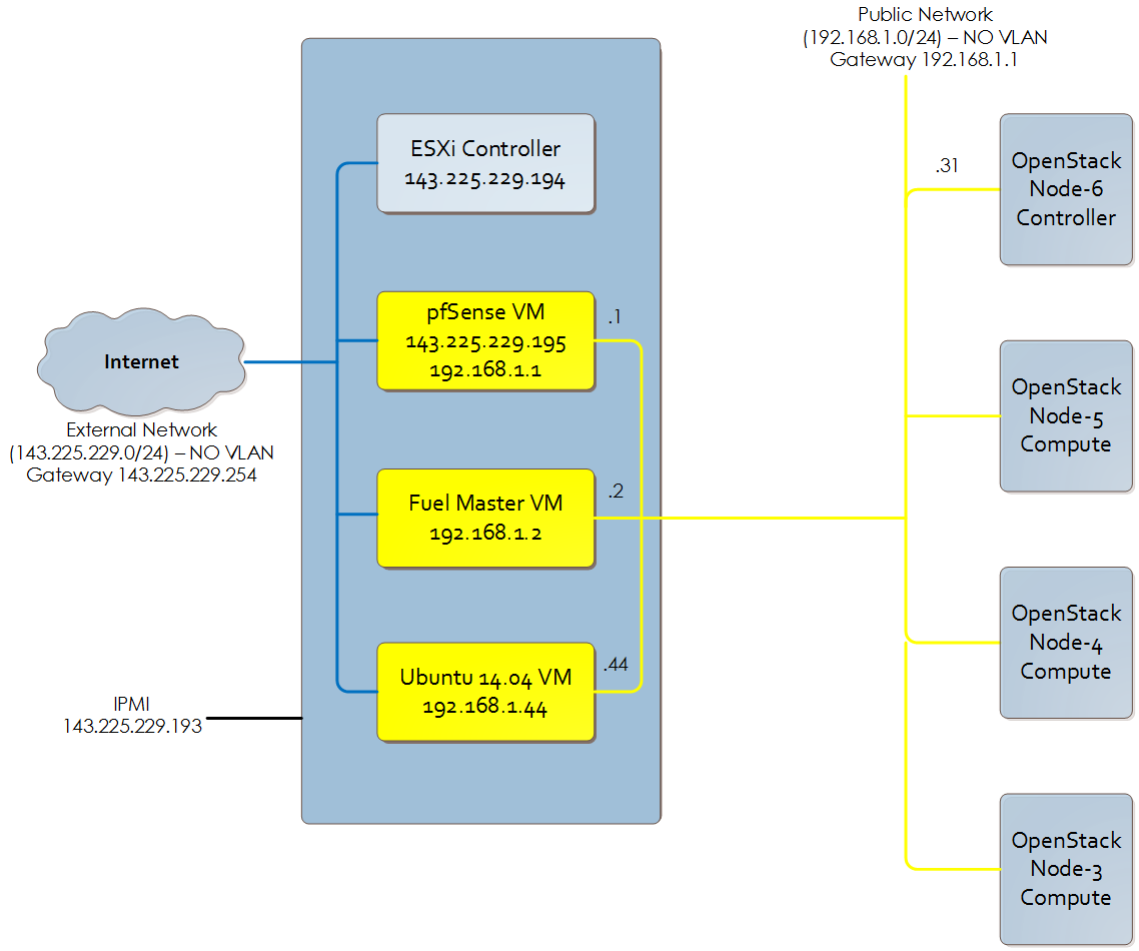


FIGURE 3.37: Logical view of testbed environment

3.5.4 Testbed Setup

The testbed consists of five physical machines (see Figure 3.37):

- **VMware ESXi**[8] dedicated server: it is a type 1 hypervisor that gets installed on top of the physical server. ESXi abstracts processor, memory, storage and networking resources into multiple virtual machines. To deploy virtual machines and perform administrative tasks, we use vSphere Client in order to manage the host. In particular, we create and deploy three VMs on ESXi host: pfSense, Fuel Master and Ubuntu. Each VM has a specific purpose:
  - **PfSense VM**: this VM is equipped with pfSense[9], a customized FreeBSD distribution. It is an open source implementation of a virtual firewall and a router. PfSense is installed on a dedicated virtual machine and requires

[8] https://www.vmware.com/products/esxi-and-esx/overview
[9] https://www.pfsense.org/

at least two network interfaces to operate as a firewall: it permits selected messages to flow into and out of the network. In particular, pfSense allows to setup port forwarding so that users can access services from outside of our private network. PfSense uses a file, called *config.xml*, which stores the configuration of all services available in the host.

  - **Fuel Master VM**: it hosts Mirantis Fuel[10], an open source tool that allows the automated deployment and management of an OpenStack environment, which typically is a time-consuming, complex and error-prone process. It provides both a command-line and a GUI, enabling hardware discovery, OS provisioning, service setup, cluster management, and other features. Fuel acts as a DHCP server for the node servers that are configured for booting from a network server by using the *Preboot Execution Environment* (PXE). In this way, Fuel deploys Ubuntu 14.04 and OpenStack Juno services to four servers. In detail, our OpenStack environment consists of a Controller node and three Compute nodes, which will be described below.
  - **Ubuntu VM**: this VM is equipped with the application that has been developed in order to create overlay networks between virtual machines. VMs are configured according to a specific topology.

- One server as **Controller node**: it is the control node for the OpenStack environment and it is responsible for running the management services. The Controller node runs all Nova services, except nova-compute.
- Three servers as **Compute nodes**: they represent the hosts on which we deploy virtual machines.

The testbed consists of five servers HP ProLiant MicroServer G7 N54L 704941-421 with the following configuration:

- 8 GB DDR3 RAM;
- HP NC360T PCI Express Dual Port Gigabit NIC;
- HP MicroServer Remote Access Card.

[10] https://www.mirantis.com/products/mirantis-openstack-software/

Each physical machine has three Ethernet ports, which accept cables with RJ45 connectors, and an *Intelligent Platform Management Interface* (IPMI) port, which allows to access, monitor, diagnose and manage the machine from a remote site. Five switches NETGEAR GS108T are utilized to support the following network configuration plan (see Figure 3.38):

- **External Network** (blue color): two ESXi server's ports (Ethernet and IPMI) are connected to a switch, so that we can remotely access to the host. In detail, the External Network allows access to pfSense (port 7777), ESXi (port 80) and Fuel Master (port 8000) interfaces.
- **Administrative (PXE) Network** (red color): it is used for PXE boot of Cloud servers and OpenStack installation.
- **Public Network** (yellow color): it provides virtual IPs for endpoints by which users can connect to the APIs of OpenStack services. This network is isolated from other networks in the OpenStack environment for security reasons. Public Network addresses can be divided into two groups:
  - **Public range**: these addresses can be used to communicate with the cluster and its VMs from outside of the cluster.
  - **Floating IP range**: these IPs are assigned to the VMs, allowing them to communicate with the outside world.
- **Private Network** (green color): it is used for the communication between each tenant's VMs. The VLAN range used for private networks is configured by using the Fuel GUI.
- **Management Network** (green color): it is responsible for the exchange of messages between Controller, Compute and Storage nodes.
- **Storage Network** (green color): it separates tenant storage traffic from other messages.

The Fuel node is connected only to External, Administrative and Public networks, while the other nodes are connected to all networks but one (External Network).

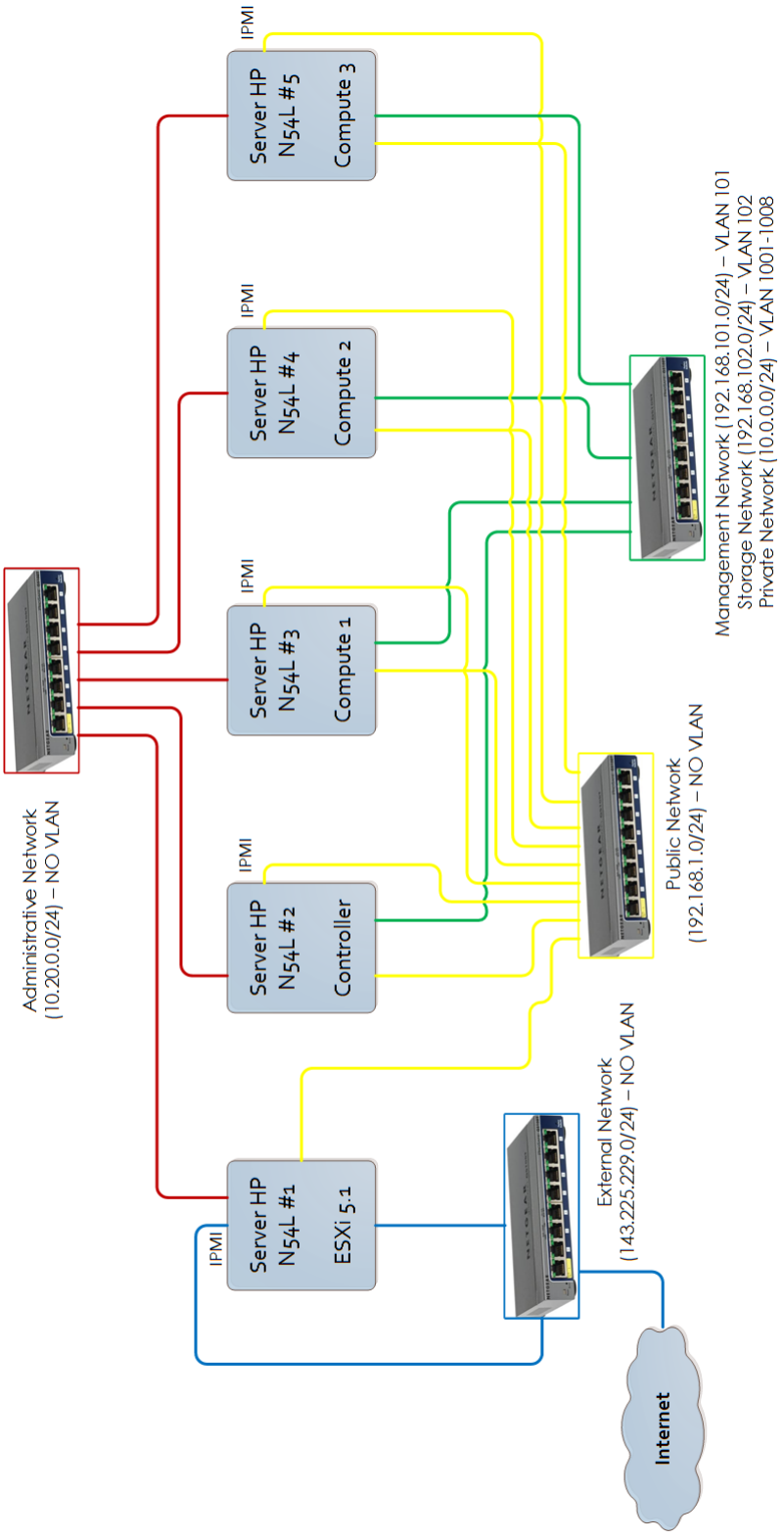


FIGURE 3.38: Physical view of the testbed environment

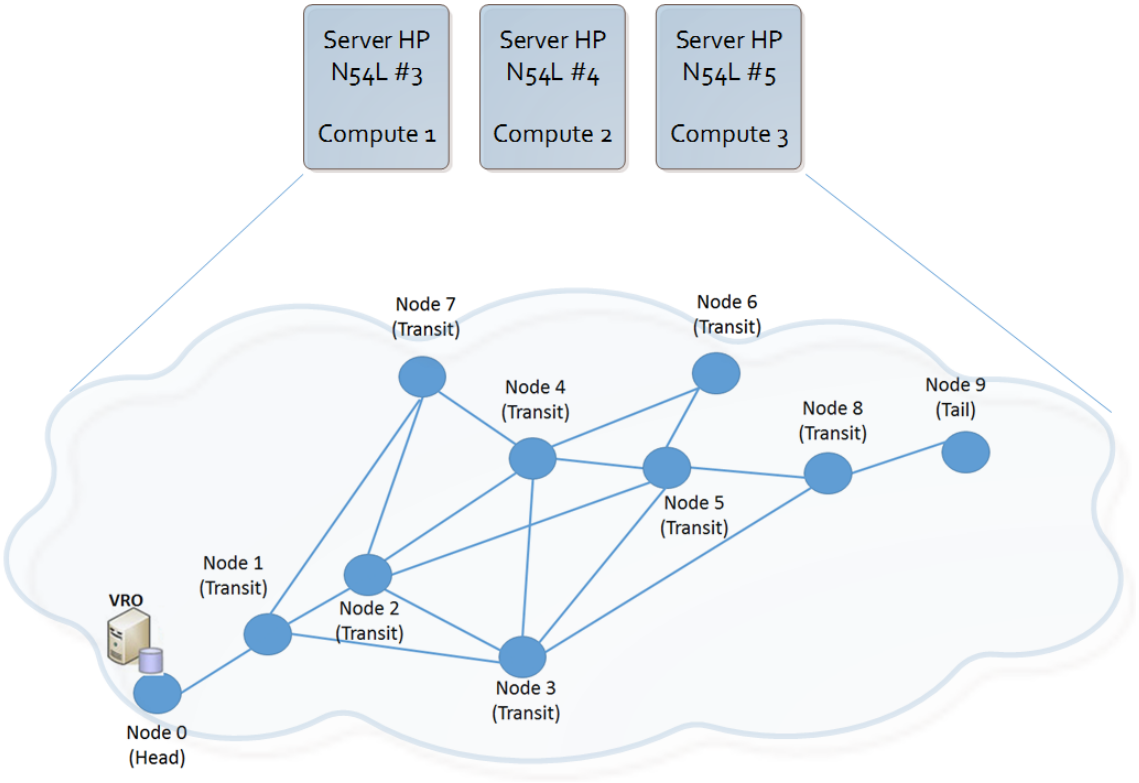


FIGURE 3.39: Topology of the virtualized infrastructure

3.5.5 Testbed Results

Figure 3.39 depicts the topology of a virtualized infrastructure where several VLSRs, implemented by means of virtual Ethernet switches and DRAGON software, provide traffic engineered paths. This infrastructure includes ten virtual machines, which are deployed on the Compute nodes of the OpenStack environment, and GRE tunnels that interconnect the VMs. Tunnelling provides a means for encapsulating packets inside a routable protocol via virtual interfaces.

The topology of the virtualized infrastructure consists of:

- head node (node 0), where a special version of the DRAGON RCE is configured to cover the VRO functionalities;
- transit nodes (1, 2, 3, 4, 5, 6, 7, 8), which act as intermediary nodes and implement OSPF-TE and RSVP-TE functionalities;
- tail node (node 9) representing the destination node of the topology.

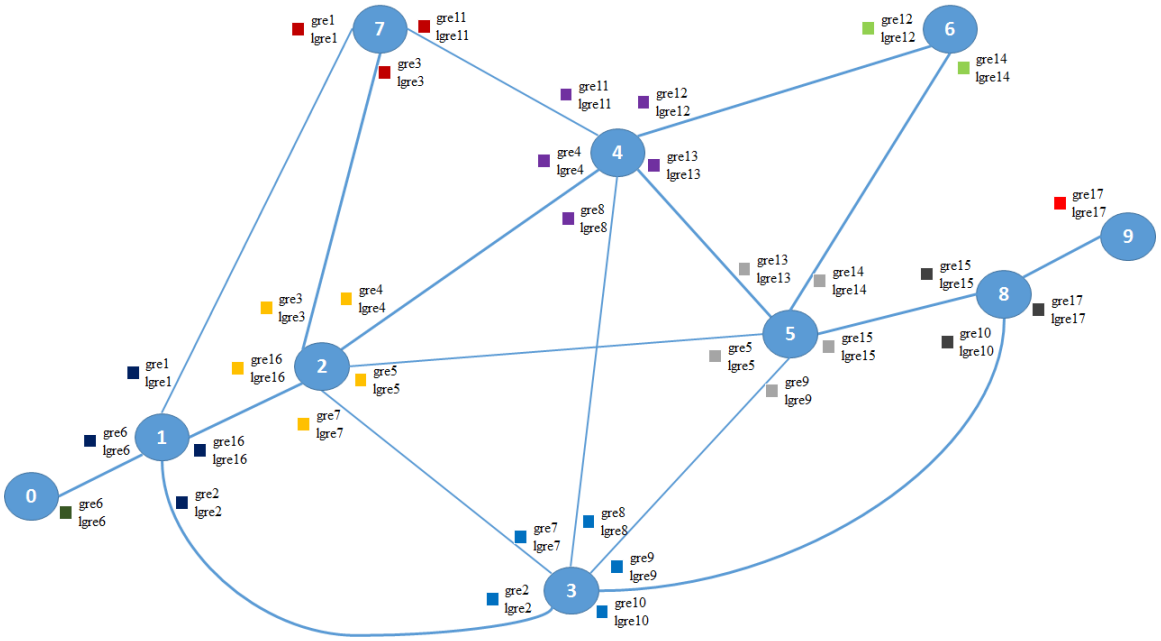


FIGURE 3.40: GRE tunnels between virtual switches

The VRO computes the best path connecting source-destination pairs of a VI, by considering the cost of the links and the energy consumption of the nodes. The VLSR and the (NARB)/RCE DRAGON components on each node of the transport network topology are configured. The VLSR allows to control Ethernet switches via the GMPLS control plane, making them capable of label switching. Connections between different nodes and interfaces of the topology are created by configuring several GRE tunnels, where routing and signalling messages pass through. Each node is configured with an interface exclusively dedicated to the exchange of control plane messages (GRE interface), and another one for exchanging routing messages (LGRE interface) (see Figure 3.40 and Table 3.8). On each node a local interface (local interface) is also configured in order to allow communications between nodes.

The testbed is designed to operate in the following way: ten virtual switches are first deployed and the DRAGON components are executed. Once the infrastructure has been created, a script is executed in order to build a LSP packet, with the aim of establishing a path from the head node 0 to the tail node 9. The link costs can have two different types of values, according to a *Green Path* (GP) metric or a *Shortest Path* (SP) metric, depending on the test that we want to perform. By default, the algorithm computes the best path by minimizing the distances (SP metric). If we want to calculate the path that guarantees the minimum power consumption, we should activate the GP metric.

TABLE 3.8: IP addresses of GRE/LGRE interfaces

Node 0	**Node 1**	**Node 2**	**Node 3**	**Node 4**
gre6 192.168.105.1	gre1 192.168.106.1	gre3 192.168.100.1	gre2 192.168.107.2	gre4 192.168.101.2
lgre6 192.168.254.1	lgre1 192.168.253.1	lgre3 192.168.0.1	lgre2 192.168.252.2	lgre4 192.168.251.2
	gre2 192.168.107.1	gre4 192.168.101.1	gre7 192.168.108.2	gre8 192.168.109.2
	lgre2 192.168.252.1	lgre4 192.168.251.1	lgre7 11.0.0.2	lgre8 12.0.0.2
	gre6 192.168.105.2	gre5 192.168.102.1	gre8 192.168.109.1	gre11 192.168.113.1
	lgre6 192.168.254.2	lgre5 10.0.0.1	lgre8 12.0.0.1	lgre11 15.0.0.1
	gre16 192.168.118.1	gre7 192.168.108.1	gre9 192.168.110.1	gre12 192.168.114.1
	lgre16 20.0.0.1	lgre7 11.0.0.1	lgre9 13.0.0.1	lgre12 16.0.0.1
		gre16 192.168.118.2	gre10 192.168.112.1	gre13 192.168.115.1
		lgre16 20.0.0.2	lgre10 14.0.0.1	lgre13 17.0.0.1

Node 5	**Node 6**	**Node 7**	**Node 8**	**Node 9**
gre5 192.168.102.2	gre12 192.168.114.2	gre1 192.168.106.2	gre10 192.168.112.2	gre17 192.168.119.2
lgre5 10.0.0.2	lgre12 16.0.0.2	lgre1 192.168.253.2	lgre10 14.0.0.2	lgre17 192.168.1.2
gre9 192.168.110.2	gre14 192.168.116.2	gre3 192.168.100.2	gre15 192.168.117.2	
lgre9 13.0.0.2	lgre14 18.0.0.2	lgre3 192.168.0.2	lgre15 19.0.0.2	
gre13 192.168.115.2		gre11 192.168.113.2	gre17 192.168.119.1	
lgre13 17.0.0.2		lgre11 15.0.0.2	lgre17 192.168.1.1	
gre14 192.168.116.1				
lgre14 18.0.0.1				
gre15 192.168.117.1				
lgre15 19.0.0.1				

By issuing a command at the DRAGON shell, we may switch from SP metric to GP metric. More in depth, the GP metric (M_{GP}) is computed by referring to Equation 3.36:

$$\begin{aligned} M_{GP} = & \left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}}\right]_i + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}}\right]_n \\ & + \left[\overline{x_{if}} \cdot P_{S_{if}} + \beta_{if} \cdot \frac{currRate_{if}}{maxRate_{if}}\right]_j + \left[\beta_{lc} \cdot \frac{currRate_{lc}}{maxRate_{lc}}\right]_m \\ & + \left[\frac{L_{link}}{L_{A_{max}}} + 2\right] \cdot \left(\overline{x_A} \cdot P_A^S\right) + \left[\frac{L_{link}}{L_{R_{max}}}\right] \cdot \left(x_R \cdot P_R^S + \delta_R \cdot currRate_\lambda\right) \end{aligned} \tag{3.36}$$

The power consumption of nodes and links are computed by referring to Table 3.5, Table 3.6, and Table 3.7 that have been previously discussed. Moreover, the lengths (km) of the fifteen bidirectional links of the illustrated network topology are specified in Figure 3.14. The SP metric is computed by considering the well-defined cost of an interface in OSPF ($cost = 100000000/bandwith\ in\ bps$). The bandwidth values are taken from Table 3.2. We first use SP metric (i.e. not minimizing power consumption) in order to establish a path from node 0 to node 9. In details, the LSP status summary displays the hops that are traversed by packets in order to reach the destination (see Figure 3.41a). The path followed by the LSP packets is the following: the head node creates the packet; through routing interface lgre, the head node forwards the LSP towards the interface of the node 3. The above-mentioned node receives the packet and places it on the output interface. Finally, the packet arrives at node 5. The resulting path 0-1-3-8-9 is shown in Figure 3.42a.

Now we analyze the results of the same test performed with GP metric. In Figure 3.41b we show the output of the *show lsp* command at the DRAGON shell. The path followed by the LSP packets is the following: the head node creates the packet and sends it to node 1; through routing interface lgre ($M_{GP} = 4.5$), the node 1 forwards the LSP towards the interface of the node 2 ($M_{GP} = 4.5$). This node places the packet in outgoing on the node 2 interface ($M_{GP} = 3.6$). Therefore, the packet is received by the node 4 interface ($M_{GP} = 3.6$), which places it on the output interface ($M_{GP} = 3.6$). The packet is received by the node 5 interface ($M_{GP} = 3.6$), which places it on the output interface ($M_{GP} = 4.5$). Finally, the packet arrives at the node 8 interface ($M_{GP} = 4.5$), which forwards the LSP to the tail node. The path 0-1-2-4-5-8-9 is shown in Figure 3.42b. This experiment shows that our implementation is able to setup a path through geographic network nodes, and this path minimizes the overall power consumption of the WAN infrastructure.

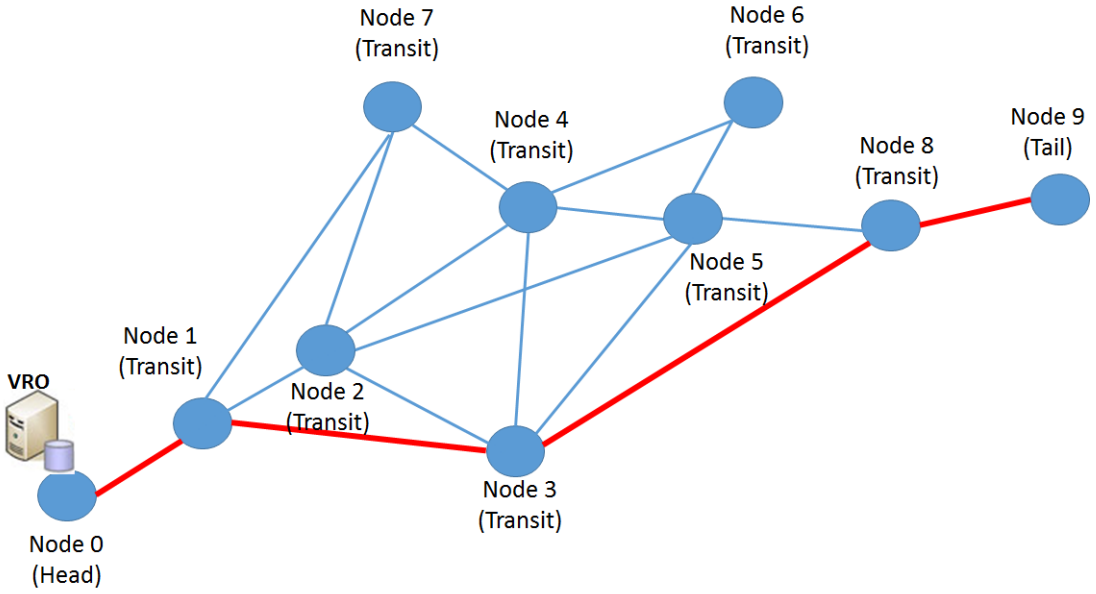
```
show LSP details.

show lsp lsp_NodeA_NodeF
Src 192.168.254.1/10, dest 192.168.1.2/100
Hop IP 192.168.254.1 strict
Hop IP 192.168.254.2 strict
Hop IP 192.168.252.1 strict
Hop IP 192.168.252.2 strict
Hop IP 14.0.0.1 strict
Hop IP 14.0.0.2 strict
Hop IP 192.168.1.1 strict
Hop IP 192.168.1.2 strict
Generic TSPEC R=eth100M, B=eth100M, P=eth100M, m=100, M=1500
Encoding ethernet, Switching l2sc, G-Pid ethernet
Ingress Local ID Type: none id, Value: 10
Egress Local ID Type: none id, Value: 100.
No E2E LSP VLAN Tag configured.
Status: In service
```

(A) Without power constraints

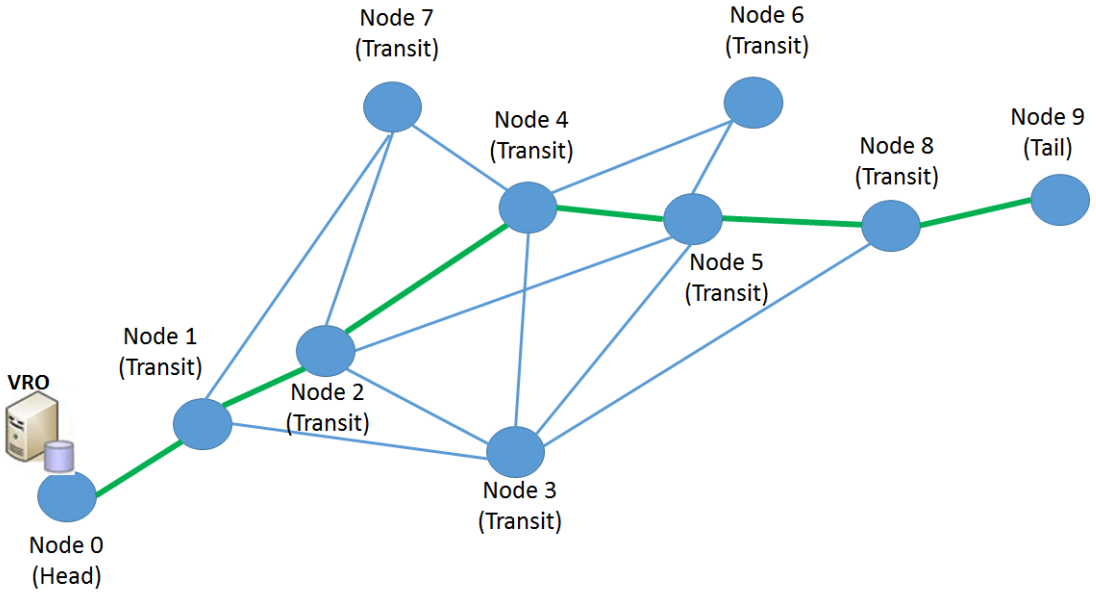
```
show LSP details.

show lsp lsp_NodeA_NodeF
Src 192.168.254.1/10, dest 192.168.1.2/100
Hop IP 192.168.254.1 strict
Hop IP 192.168.254.2 strict
Hop IP 20.0.0.1 strict
Hop IP 20.0.0.2 strict
Hop IP 10.0.0.1 strict
Hop IP 10.0.0.2 strict
Hop IP 19.0.0.1 strict
Hop IP 19.0.0.2 strict
Hop IP 192.168.1.1 strict
Hop IP 192.168.1.2 strict
Generic TSPEC R=eth100M, B=eth100M, P=eth100M, m=100, M=1500
Encoding ethernet, Switching l2sc, G-Pid ethernet
Ingress Local ID Type: none id, Value: 10
Egress Local ID Type: none id, Value: 100.
No E2E LSP VLAN Tag configured.
Status: In service
```

(B) With power constraints

FIGURE 3.41: Resulting path



(A) Without power constraints



(B) With power constraints

FIGURE 3.42: Network Topology calculation

Chapter 4

Security Management in a PCE-based Infrastructure

4.1 Context and Motivations

Over the latest years, the interest in Cloud computing has rapidly increased due to the flexibility and availability of the computing resources at a lower cost [27] [187]. The Cloud deployment model should be selected with the awareness that moving from private to public solutions, the control over data is gradually lost. It is possible that some information may require special security measures due to its intrinsic nature (e.g. health, genetic, financial data). In such cases, due to risks like unauthorized accesses, consumers should carefully decide whether it is recommendable to use the Cloud services or to preserve an in-house processing of such data types.

From small business to enterprise solutions, fully integrated security is often based on perimeter access control technologies, while Cloud services are outsourced to third-party Cloud providers and many companies share the same infrastructure within the public Cloud. In other words, because of fixed infrastructures gradually replaced by Cloud networks, security perimeters are no longer physical but virtual [188].

Moreover, customers should know whether data will remain in the physical availability of the provider, or if the service is designed on technologies provided by a third-party provider. The use of virtual resources makes Cloud exposed to attacks: eavesdroppers

can access secret information, violating confidentiality; hackers can directly attack the Cloud domain to delete messages, inject erroneous messages or impersonate a node, violating integrity of data. Compromised nodes, in turn, can launch attacks to other nodes within a network. Because any centralized entity is vulnerable to several types of security threats in the Cloud, a security solution must be based on the principle of distributed trust where one provider cooperates with some others [189] [190].

In this chapter, a focus on a *Virtual Intrusion Detection System* (VIDS) will be provided, which contributes to face security problems in the Cloud by implementing a real-time traffic analysis, providing re-actions and using ubiquitous control systems [191]. The traffic monitoring is a fundamental part of the network security: the idea is to define a distributed traffic analyzer that provides a real-time feedback, and shares the results between neighbouring nodes. A method based on the standard protocols of the GMPLS suite to react to malicious attacks will be introduced in this section as well.

4.2 Related Work

Cloud computing services require complicated management procedures in order to guarantee performance, security, robustness and reliability. Furthermore, there are not so many accepted standards or open source solutions for Cloud management and monitoring technologies. Different Cloud deployment models (public, private, community and hybrid) need different solutions about monitoring: public Clouds have geographically distributed resources, requiring large investments in monitoring and scalability, while in private Clouds resources are accessible within a private organization.

In this regard, in [192] authors give examples related to password management, backup policies and repair time. They propose a four-step process for defining Cloud-specific security service levels to make SLAs more auditable. There are open questions about the need to improve QoS through security SLA specifications and metrics, and to provide an architecture for both monitoring and management of security SLAs. They also analyze security in SLAs, focusing on measurable security metrics combined with a monitoring and controlling architecture.

In [193] authors propose the *Grid and Cloud Computing Intrusion Detection System* (GCCIDS), to cover the attacks that network and host cannot detect, by using an audit

system that monitors each node and alerts the others when the probability of an attack is high; therefore, each node contains IDS, which includes two components: Analyzer and Alert Systems. The Event Auditor monitors message exchange between nodes and captures system logs. Based on the data received from Event Auditor, the IDS service makes use of two intrusion detection techniques:

- Behaviour-based method, to control if user actions match usual behaviour profiles (this technique detects unknown attacks);
- knowledge-based method, which verifies security policy violations (this technique detects known attacks).

An *Artificial Neural Network* (ANN) to detect unknown attacks is used, and it requires lots of training samples as well as conspicuous time for detecting intrusions effectively. In order to reduce training time of ANNs, fuzzy logic can be used. Authors developed a prototype to evaluate the proposed architecture, using a middleware called Grid-M: a higher level of security and a lower rate of false positives and false negatives are achieved. Analysis is individually performed on each node, resulting in a lower data transmission between nodes and a decreased complexity of the system. GCCIDS has the following weaknesses: it can only detect specific intrusions, without preventing new types of attacks or creating an attack database. Furthermore, this approach works for intrusions at the middleware layer only (PaaS), therefore it is necessary to evaluate the possibility of extending the discussion to IaaS.

In [194] the authors propose a generic Cloud monitoring architecture for private Clouds; it is characterized by three levels and it is simple, modular and extensible. Infrastructure layer is composed of heterogeneous resources: services, hardware, network and available software; integration layer separates infrastructure details and monitoring information required by Cloud users; view layer provides interface through which monitoring information is analyzed. In order to validate the proposed architecture, authors developed a Private Cloud Monitoring System called PCMONS. It mainly works at the integration layer, by retrieving, gathering and preparing relevant information for the visualization layer. In previous works, we note the absence of a distributed infrastructure and the inability to detect new types of attack. In this context, we propose a novel architecture,

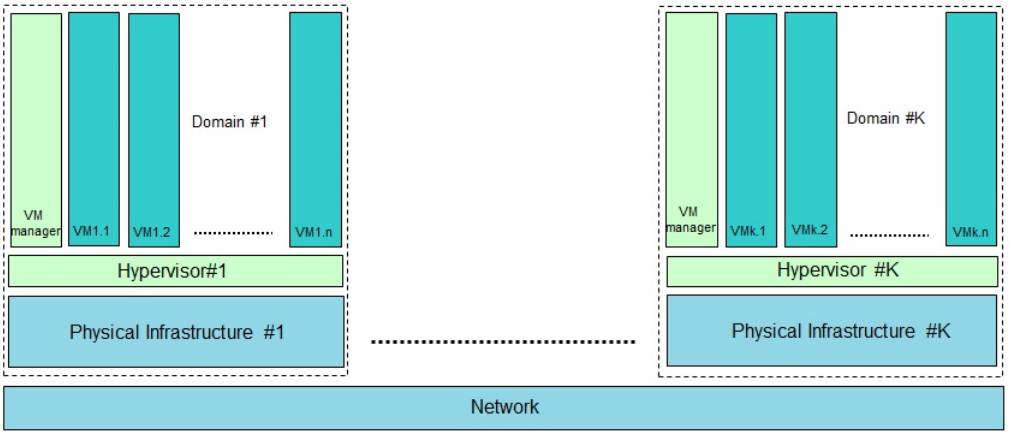


FIGURE 4.1: A simplified representation of an IaaS infrastructure

where a distributed traffic analyzer supervises, on top of a Cloud domain, computational resources and log messages in order to perform the related security corrections in real-time, using a closed-chain IDS.

4.3 Architecture of a Secure Cloud Infrastructure

In this chapter, we focus on IaaS service model because it guarantees the highest level of control over the infrastructure. Virtualization is a fundamental enabler to IaaS and it is commonly defined as a technology that introduces a software abstraction layer between the hardware and the operating system, and applications running on top of it [4]. This abstraction layer is called hypervisor, which provides a compatible, uniform view of underlying hardware, and hides the physical resources of the computing system from the operating system, making machines from different vendors with different I/O subsystems look the same. This means that it is possible to run multiple virtual machines with different operating systems in parallel on the same hardware.

IaaS Cloud delivery model provides an interface for instantiating VMs from system images. Figure 4.1 offers a simplified representation of an IaaS infrastructure that shows different Cloud domains, each of which abstracts completely heterogeneous physical resources (network and IT) from operating systems and applications using virtualization. Each Cloud domain contains a particular VM that provides a range of VM-related management tasks (activation, deletion, migration, monitoring, etc.). In this scenario, Cloud provider has total control over physical infrastructure, and has admin rights for

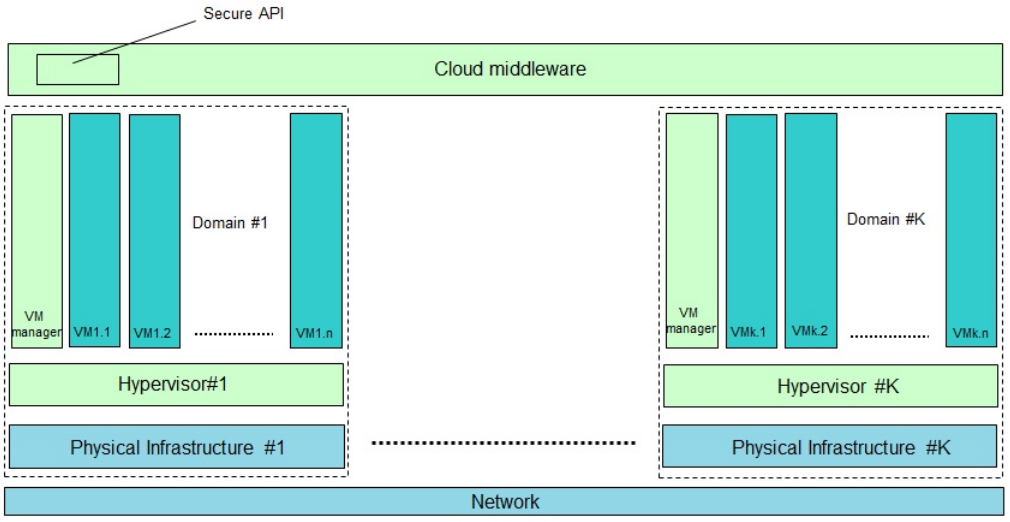


FIGURE 4.2: Example of a secure Cloud infrastructure

hypervisor and VM manager; moreover, it has no control over guest OS, middleware and application layer. Conversely, Cloud subscriber has admin rights for application deployed, middleware and guest OS, and it can make requests to hypervisor; moreover, it has not control over hardware. In this context, we need a distributed security system that works on a different layer than VM manager and hypervisor ones. Therefore, for each Cloud domain we define a trusted security perimeter that is controlled by a closed-chain IDS. We propose an architecture where the Cloud middleware orchestrates the local management system of each domain, exposing a secure API (see Figure 4.2); the API allows to exchange information to modify the status of each VM, relying on information collected about security state.

4.4 The Virtual Intrusion Detection System

The *Virtual Intrusion Detection System* (VIDS) is a relatively new and innovative idea in the Cloud security domain. It provides active alarms and performance reports about Cloud network, it analyzes the data in near real time and provides automatic re-action after a security issue (e.g. secure routing in Cloud networks defined at runtime). The proposed architecture enhances the IDS basic principles by introducing a set of functionalities, which allow the reconfiguration of the Cloud infrastructure in a smart and secure way. Below, component modules of the VIDS architecture are described (see Figure 4.3):

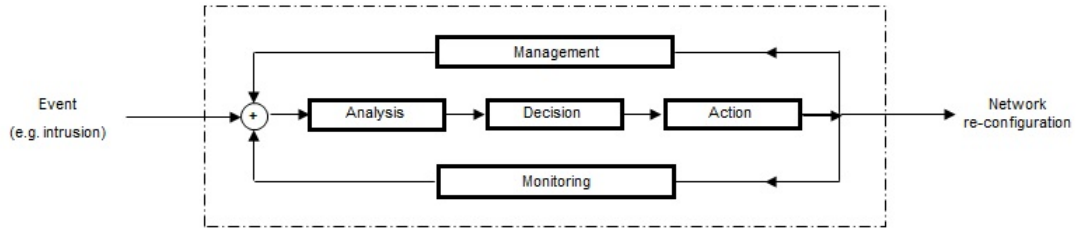


FIGURE 4.3: Conceptual model of the VIDS

- **Analysis module**: it reviews data from network and IT devices, providing an overall picture of the logical flow of incoming events, useful to the decision module to correct decisions in an automatic/adaptive way. The analysis module works with a preconfigured schema provided by the management module, and it conducts analysis considering already processed data related to previous events logged by the monitoring module. Several methods to find recursive patterns in the large amount of data collected by IDS in log files are available. In the VIDS context, the analysis module implements two different methods:
  - *Signature detection*: it represents the simplest way to analyze IDS data. The characteristics of previous attacks are identified, recorded into a database, and compared with information related to possible new attacks, in order to detect them.
  - *Anomaly detection*: it represents the most sophisticated technique. The normal behaviours of the host or the network that is under protection are learned and recorded into a database. After that, the analysis module tries to detect variations between the current behaviours and the information in the database.
- **Decision module**: it collects data from analysis module and takes decisions about real-time actions to be done, in order to solve anomalies related to network intrusions.
- **Action module**: it represents the actuator that solves security issues in a Cloud domain.
- **Monitoring module**: it captures intrusion and login attempts. Each event is time-stamped and stored in a database. A complete correlation map is created by using raw data captured by the monitoring module.

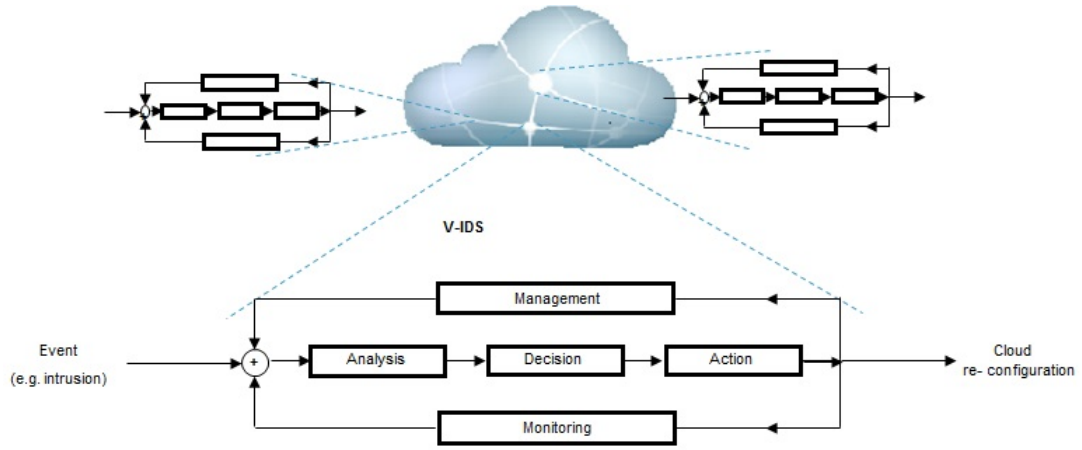


FIGURE 4.4: Conceptual model of distributed VIDS

- **Management module**: it properly configures and updates the VIDS connected to the Cloud infrastructure.

4.5 Distributed VIDS in the Cloud

The conceptual model of the Virtual IDS can be centralized or distributed. The centralized VIDS is a combination of individual sensors, which collect and forward network data to the central management system, where data of a well-defined Cloud network domain is stored and processed. Distributed VIDS, instead, includes more virtual devices that cooperate in order to perform both the data gathering and reporting functions. We propose a distributed VIDS based on a traffic analyzer that collects computational resources and log messages within a Cloud domain, in order to make the necessary security corrections in real-time.

In the proposed architecture in Figure 4.4, each node is responsible for locally detecting security issues, but neighbouring nodes can collaboratively monitor a broader area. VIDS instances run independently and monitor local activities, sharing hardware and software resources and implementing a closed-chain control system: they detect intrusions from local traces and react properly.

4.6 Prototype of a Cloud VIDS

Our objective is to design a flexible framework that ensures a high network protection from malicious attacks. Monitoring is mandatory to constantly check the performance of the system and guarantee that the required QoS levels are being met under the current security settings. The threat level of the network is based on the events reported by VIDSs or other entities (or both), such as firewalls and VIDSs of other collaborating networks. In other words, VIDS monitors network intrusions or malicious attacks to determine a number of threat levels ranging from low to severe. Based on the defined secure Cloud infrastructure and VIDS model, we have implemented a prototype of Cloud-based VIDS. Its main components are:

- **Security monitor**: it implements the monitoring module of the VIDS model. It was developed extending the OpenStack Telemetry service (Celiometer) [195].
- **Security manager**: it implements the management module of the VIDS model. It was developed extending the OpenStack Identity service (Keystone).
- **Security controller**: it implements the analysis-decision-action modules of the VIDS model. It was developed extending the DRAGON suite.

The implementation of the VIDS involves the control plane architecture, both in terms of protocol extensions and security constraints management. We introduce an architecture where logical modules involved are:

- PCC module, which performs the request of a path computation that has to be solved by the proper PCE. The information is sent by using the PCEP. The request includes information about the requested connection (source and destination addresses, security constraints). The Virtual IDS integrates the PCC component and sends requests to the PCE module.
- PCE module, which computes a path based on the network state information and security costs.
- Traffic Engineering Database is the repository where routing information retrieved by the PCEs is stored. It is related to available network resources such as network

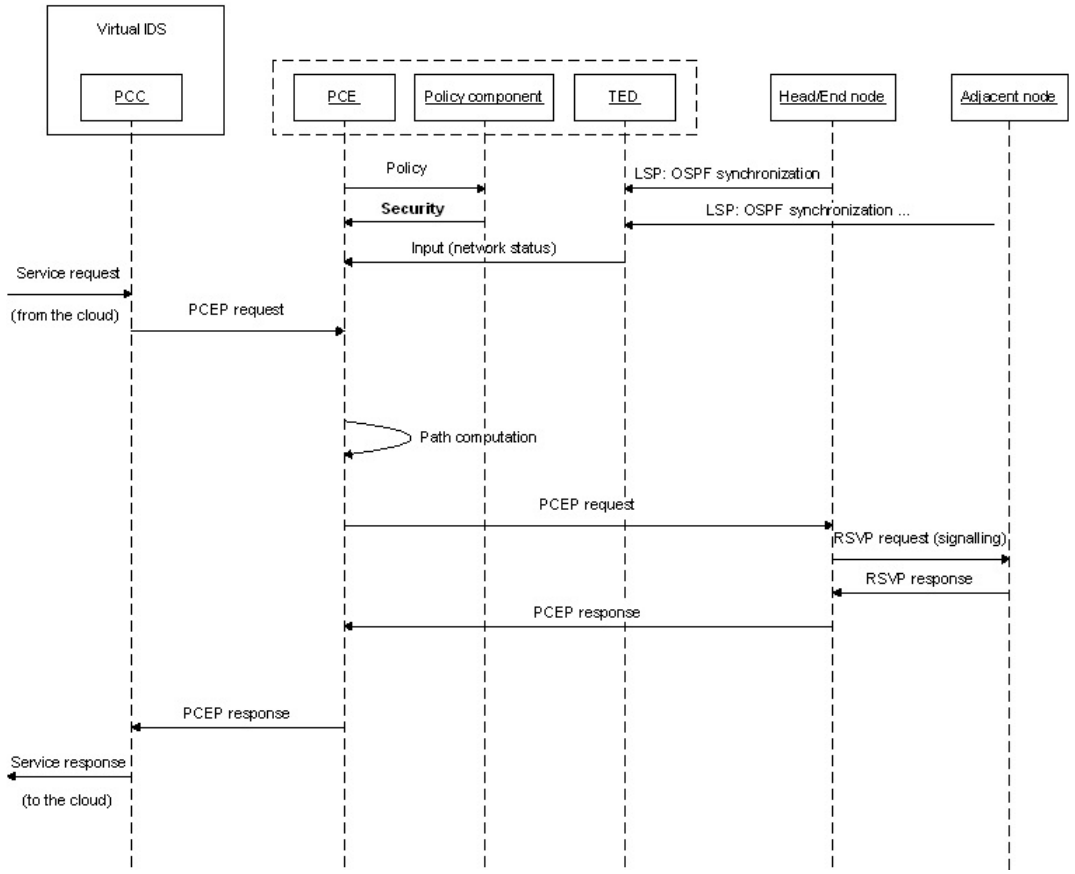


FIGURE 4.5: Virtual IDS secure path computation

topology, nodes and links status, connectivity, available bandwidth and security costs.

- Policy component provides PCE with the set of constraints that should be applied during the path calculation (security metric).

The network control plane might be embedded in the network element or not. In the last case, the signalling module and the routing module interact with the management interface of the network element, by turning RSVP and OSPF frames in secure management frames. The sequence diagram in Figure 4.5 shows the secure path calculation process, initiated by the Virtual IDS (PCC). The computation workflow follows these steps:

1. Off-line, with respect to the normal data exchange, a synchronization phase is started for populating the part of the database used for determining the correct route of the data over the network. This step is divided in two independent actions:

- Security Policy setup, used to configure the policy and security constraints that involve the area of the network under analysis.
- Security policy distribution, used to distribute network security information.

2. Once VIDS receives a service request, it chooses the PCE that is able to satisfy the request. The request is analyzed by the PCE by using the security information that is contained in the TED (acquired in step 1).

3. Based on the information received, the PCE calculates a new path.

4. The head node of the chain starts the RSVP reservation phase.

Figure 4.6 shows the computation of the transmission path with low level of security, or with the security controller disabled, while Figure 4.7 shows the alternative transmission path, calculated for high-security purposes. Results are achieved by considering the prototype implementation of the VRO architecture discussed in Chapter 3.

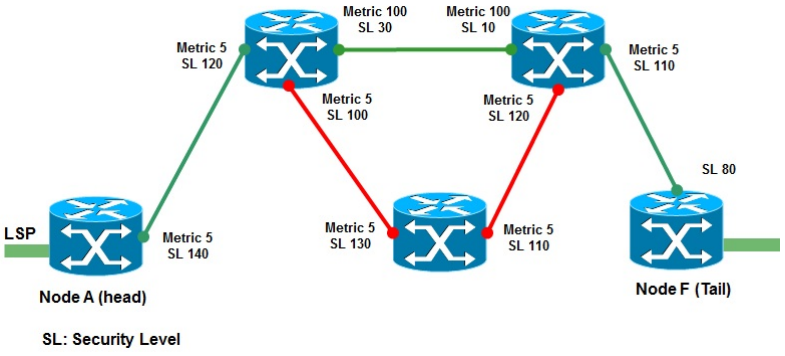


FIGURE 4.6: Low-security path

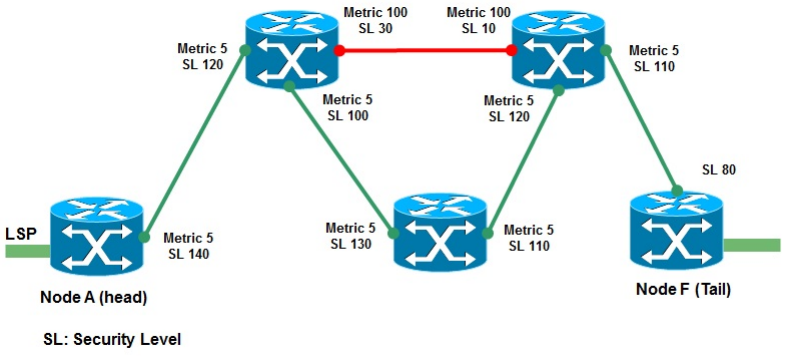


FIGURE 4.7: High-security path

Chapter 5

Conclusions

Computing has evolved over time according to different paradigms, along with an increasing need for computational power. Modern computing paradigms basically share the same underlying concept of *Utility Computing*, that is a service provisioning model through which a shared pool of computing resources is used by a customer when needed. The objective of *Utility Computing* is to maximize the resource utilization and bring down the relative costs. Afterwards, the *Cloud Computing* paradigm emerged as a result of the growth and confluence of several approaches and technologies, such as *Utility Computing*, virtualization, service orientation, and distributed computing. As the popularity of the *Cloud Computing* has grown over the past few years, several different service and deployment models have emerged with the aim of meeting the needs of distinct user categories. In particular, the *Infrastructure as a Service* (IaaS) model has become the mainstream in the commercial world; the IaaS includes the key components for Cloud IT and its aim is to provide users with the highest level of management control and flexibility over IT resources.

IaaS systems enable the joint deployment of infrastructures and applications, and their realization requires a control platform for orchestrating the provisioning, configuration, and management of the virtual resources over physical hardware. Yet, since the orchestration process is complex and potentially error-prone if performed manually, the need for a tool that minimizes human intervention in order to automatize resource configurations and to remain scalable is necessary. Indeed, an easy configurable system allows to reduce costs and scale data center energy consumption. From the Cloud provider's

perspective, however, building a system to orchestrate resources is challenging, due to the growth of data centers, QoS constraints and SLAs. Therefore, the management problem of network and IT resources has generated a lot of interest among the research community and ICT industry.

Due to the success of the *Cloud Computing* paradigm, in the last few years there has been a remarkable growth in the number of data centers, which represent one of the leading sources of increased business data traffic on the Internet. They store increasing amounts of data while offering also a wide variety of services. Furthermore, in order to increase performance and reliability, as well as to support the growing service demands from customers, Cloud providers are now building additional geographically distributed data centers with high-speed backbone networks interconnecting them. An effect of the growing scale and the wide use of data centers is the dramatical increase of power consumption, with significant consequences both in terms of environmental and operational costs. Moreover, the network communication between interconnected VMs might involve additional network elements (switches and links), which means a higher consume of energy. In addition to energy consumption, also carbon footprint of the Cloud infrastructures is becoming a serious concern, since a lot of power is generated from non-renewable sources. Hence, energy awareness has become one of the major design constraints for Cloud infrastructures. In order to face these challenges, a new generation of energy-efficient and eco-sustainable network infrastructures is needed. In this regard, several approaches have been promoted in order to reduce CO_2 emissions resulting from data centers. One of these approaches, for instance, is the use of green nodes that are powered by renewable energy sources.

In this thesis, a novel energy-aware resource orchestration framework for distributed Cloud infrastructures is discussed. The aim is to explain how both network and IT resources can be managed while, at the same time, the overall power consumption and carbon footprint are being minimized. A high-level overview of the system architecture is proposed and is made up of three layers: the *Physical Layer*, the *Virtual Layer*, and the *Management Layer*. At the *Physical Layer*, network nodes and physical machines (hosting VMs) are organized in clusters. Each device is controlled by a system service that is referred to as *Local Controller*, which monitors network devices and detects overload/underload anomaly situations. The *Virtual Layer* provides scalability to the

system and it is composed of a *Domain Leader* that manages a subset of *Domain Managers*. Finally, the *Management Layer* provides the user interface to customers. VMs and network devices are periodically monitored, and resource utilization information is transferred to the corresponding *Domain Manager* by the *Local Controllers*. Based on the information received, the *Domain Manager* takes decisions about VM allocation and consolidation (in order to solve host overload/underload problems), and computes the appropriate network path in order to satisfy energy-efficient objectives. VM allocation and consolidation decisions contribute to the creation of a *Green Migration Plan*, which specifies the new mapping between network/IT resources and *Local Controllers*. Virtual machines are periodically consolidated into a lower number of physical hosts, in order to turn off the idle devices, thus saving energy consumption. The migration plan is achieved by applying several resource relocation algorithms, which are evaluated according to their performance.

Afterwards, the aforementioned resource management framework is used to offer resources to service providers in the form of virtual infrastructures, while ensuring that energy consumption and CO_2 emissions are minimized. The problem we addressed is twofold: from the network-side, a green management of the geographical networking infrastructure must be pursued, in order to lower the energy consumption and the CO_2 emissions; from the IT-side, an energy efficient consolidation of virtual machines in data centers needs to be reached. The optical network infrastructure is assumed GMPLS-enabled, because the *Generalized Multi-Protocol Label Switching* (GMPLS) is a good solution for providing enhanced traffic engineering facilities within backbone networks. Then, an extension to the standard *Path Computation Element* (PCE) architecture has been proposed, with the aim of assigning to a centralized entity, named *Virtual Resource Orchestrator* (VRO), the responsibility of optimally allocating the physical resources needed to instantiate a requested Virtual Infrastructure. The PCE provides functions of path computation in support of traffic engineering in networks controlled by GMPLS. The OSPF-TE routing protocol within the GMPLS framework has been properly extended to include energy-related information, such as the carbon footprint and the energy consumption associated with nodes and links. Furthermore, based on energy cost and carbon footprint metrics, a constrained shortest-path Dijkstra's algorithm that minimizes energy consumption and CO_2 emissions of networks connecting multiple distant data centers has been implemented. We have presented an extensive evaluation

conducted by simulations. These simulations show how the VRO can be configured to pursue green management objectives and to obtain appreciable energy savings while, at the same time, preserving contractual SLAs related to minimum guaranteed bandwidth and maximum activation time of a network path. This is a multi-objective optimization problem in which two distinct objective functions are considered, namely the power consumption and the *Greenhouse Gas* (GHG) emissions, which are combined into a single total cost function.

The simulations have been conducted in a test scenario composed of eight geographically distributed data centers interconnected through an optical backbone. We have supposed that the eight data centers are spread all over the U.S. and that the data centers are co-located at eight different generation sources, so that they will use power produced locally. At the beginning, one optimization objective is considered at a time, in order to evaluate separately the contribution of the individual objective functions to the multi-objective optimization problem, without taking into account trade-off situations. We have compared the overall power consumption and carbon footprint resulting from the *Green*, *Min CO2*, *Min Hops*, and *Worst-Case* approaches, under the same load conditions. As expected, the lowest power consumption is achieved by the *Green* strategy, in which the cost function depends solely on the power consumption increases of network devices. The energy savings achieved by using the *Green* approach are 35% compared to *Min CO2*, 47% compared to *Min Hops*, and 68% with respect to the *Worst-Case*. Instead, the lowest GHG emissions are achieved by the *Min CO2* strategy, in which the cost function depends entirely on carbon footprint increases of the network devices. The CO_2 savings achieved by using the *Min CO2* approach are 9% compared to *Green*, 14% compared to *Min Hops*, and 33% with respect to the *Worst-Case*. Moreover, the proposed energy-aware algorithms have shown only a marginal increase for the number of routing hops compared to the *Min Hops* approach. Finally, the possible trade-offs have been discussed: when the objectives are equally weighted, the best compromise has been achieved.

A performance evaluation of several energy efficient VM consolidation algorithms has also been conducted by using CloudSim tool. Dynamic resource provisioning policies on a large heterogeneous Cloud infrastructure have proven to be able to minimize data center energy consumption. In more detail, THR-MMT policy achieves the best energy savings by using a single utilization threshold in order to determine if a host is overloaded,

and by selecting the VM that requires the minimum time to complete a migration, compared to other VMs allocated to the host. Then, a prototype of our framework has been implemented, which is able to configure GMPLS-enabled network nodes and Cloud-enabled data centers with the aim of providing Virtual Infrastructures.

Finally, we propose a solution for Cloud security based on a *Virtual Intrusion Detection System* (VIDS). We have presented a novel architecture that considers the basic principles of Cloud computing, virtualization and GMPLS, and applies them to the intrusion detection systems. The Cloud infrastructure supports a distributed architecture of ubiquitous VIDSs administrated by different stakeholders, where each VIDS follows a set of cooperation rules to form complex security services. Based on the defined architecture, we have implemented a prototype of the Cloud-based IDS. In conclusion, the VRO-based architecture has been proven to be flexible enough to adapt to varying requirements and objectives in the management of network infrastructures.

Acknowledgements

I would like to thank Professor Roberto Canonico, Engineer Pasquale Donadio, and Professor Giorgio Ventre for their professional and personal support. A special thanks goes to Carlo Bussemi, Valentina Geri, Saverio De Blasio, Daniel Burbano, Raffaele Sommese, Antonio Marotta, Professor Nicola Pasquino, and Maria De Blasio for their invaluable help. A final thanks goes to Massimiliano Minei and my former colleagues of Poste Italiane, as well as my colleagues of Media Motive. Lastly, I thank my family for the incredible and everlasting intellectual encouragement that they have always provided me with.

Ringraziamenti

Questa tesi nasce dal voler affrontare obiettivi sempre più difficili, perché si sa, la vita non è mai abbastanza complicata. Quando qualche anno fa decisi di iniziare a lavorare e, in parallelo, cominciare un dottorato su argomenti completamente differenti, provarono a farmi desistere. In effetti avevano ragione: in questi anni è stato sacrificato tutto, qualsiasi cosa possa annoverarsi tra gli abituali passatempi di un ragazzo della mia età. Eppure, oggi, io sono un po' più felice di qualche anno fa. Non sono solito dare "lezioni di vita", non sono in grado di farlo. Mi piacerebbe però che il raggiungimento di questo obiettivo possa essere di aiuto a tante persone che si sono un po' smarrite negli anni. Sin da piccolo, ho sempre creduto che tutti coloro che mi stavano attorno, compagni di scuola, colleghi, amici fossero più fortunati. In effetti, per un simpatico scherzo del destino, non ho la possibilità di riposare come tutti. Grazie al dottorato, però, ho scoperto che sono la passione e la volontà a fare la differenza. Questa esperienza dovrà servire non soltanto a me, ma soprattutto agli altri per convincersi che nessuno può stabilire quali sono i nostri limiti, nemmeno noi stessi possiamo farlo. I limiti non esistono, sono soltanto un modo per giustificare il mancato raggiungimento degli obiettivi. Tutti possono ottenere il massimo dalla propria vita, perché la felicità non è soltanto il raggiungere o meno un obiettivo. La felicità è sapere che tutti ce la possono fare, basta solo sognare un po' di più.

I miei primi ringraziamenti vanno al Prof. Roberto Canonico, tecnicamente e umanamente eccellente. Un esempio di quanto possa essere determinante il non considerarsi mai "arrivati". Lo ringrazio per i confronti costruttivi che mi ha concesso e per i tanti spunti di riflessione ricevuti. Ringrazio Pasquale Donadio, soprattutto per il supporto ricevuto quando sono stato sul punto di mollare. Ringrazio il prof. Ventre per la bellissima opportunità che mi ha concesso di lavorare con Lui e il suo gruppo, e per la fiducia che ho sentito in numerosi momenti. Ringrazio Carlo Bussemi, Valentina Geri, Saverio De Blasio, Daniel Burbano, Raffaele Sommese, Antonio Marotta, il Prof. Nicola Pasquino e Maria De Blasio. Il loro contributo è stato fondamentale per la stesura di questa tesi. Un ringraziamento va anche agli indimenticati colleghi di Poste Italiane, con i quali ho vissuto gran parte di questo percorso e che mi hanno supportato in modo

totale. Porterò sempre con me i loro insegnamenti professionali e comportamentali. Un ringraziamento particolare va a Massimiliano Minei. Non dimenticherò mai tutto ciò che ha fatto per me; lo ringrazio per avermi dato la possibilità di portare avanti un dottorato pur dovendo affrontare stressanti giornate lavorative. I suoi insegnamenti nelle vesti di ingegnere e di "fratello maggiore" li conserverò per sempre. Un ringraziamento anche ai miei nuovi colleghi di Netcom, che hanno dovuto sopportarmi nei mesi più delicati. Un grazie a Raffaella per il sostegno ricevuto in questi anni. Infine, ringrazio la mia famiglia per il supporto totale e per il considerarmi un punto di riferimento, pretendendo da me sempre il massimo (e più) di quanto una persona possa dare.

Bibliography

[1] A.A. Semnanian, J. Pham, B. Englert, and Xiaolong Wu. Virtualization Technology and its Impact on Computer Hardware Architecture. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, pages 719–724, April 2011. doi: 10.1109/ITNG.2011.127.

[2] S. Nanda and T. Chiueh. A survey of Virtualization Technologies. 2005. doi: 10.1.1.74.371. URL http://dx.doi.org/10.1.1.74.371.

[3] G. J. Popek and R. P. Goldberg. Formal Requirements for Virtualizable Third Generation Architectures. *Commun. ACM*, 17(7):412–421, July 1974. ISSN 0001-0782. doi: 10.1145/361011.361073. URL http://doi.acm.org/10.1145/361011.361073.

[4] J. Sahoo, S. Mohapatra, and R. Lath. Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pages 222–226, April 2010. doi: 10.1109/ICCNT.2010.49.

[5] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: High Availability via Asynchronous Virtual Machine Replication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 161–174, Berkeley, CA, USA, 2008. USENIX Association. ISBN 111-999-5555-22-1. URL http://dl.acm.org/citation.cfm?id=1387589.1387601.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945462. URL http://doi.acm.org/10.1145/945445.945462.

[7] VMware ESXi web site. URL http://www.vmware.com/products/esxi-and-esx/overview/.

[8] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: the Linux Virtual Machine Monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, Ottawa, Ontario, Canada, June 2007. URL http://linux-security.cn/ebooks/ols2007/OLS2007-Proceedings-V1.pdf.

[9] VirtualBox web site. URL http://www.virtualbox.org/.

[10] VMware Workstation web site. URL http://www.vmware.com/products/workstation/.

[11] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 275–287, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-636-3. doi: 10.1145/1272996.1273025. URL http://doi.acm.org/10.1145/1272996.1273025.

[12] M. N. Rao. *Cloud Computing*. PHI Learning Pvt. Ltd., 1st edition, 2015. ISBN 9788120350731.

[13] N. Poulton. *Data Storage Networking*. Sybex, 1st edition, 2014. ISBN 9781118915943.

[14] W. Chen, J. Chan, O. Mueller, M. Singh, and T. Vaattanen. *DB2 Virtualization*. IBM Redbooks, 1st edition, 2009. ISBN 9780738433431.

[15] Jorge Carapinha and J.. Jiménez. Network Virtualization: A View from the Bottom. In *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '09, pages 73–80, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-595-6. doi: 10.1145/1592648.1592660. URL http://doi.acm.org/10.1145/1592648.1592660.

[16] M. Rosenblum and T. Garfinkel. Virtual Machine Monitors: Current Technology and Future Trends. *Computer*, 38(5):39–47, May 2005. ISSN 0018-9162. doi: 10.1109/MC.2005.176. URL http://dx.doi.org/10.1109/MC.2005.176.

[17] M. Revett, I. Boyd, and C. Stephens. Network Computing: a Tutorial Review. *Electronics Communication Engineering Journal*, 13(1):5–15, Feb 2001. ISSN 0954-0695. doi: 10.1049/ecej:20010101.

[18] I. Ahmad. Network Computers: The Changing Face of Computing. *IEEE Concurrency*, 8(4):9–11, October 2000. ISSN 1092-3063. doi: 10.1109/4434.895097. URL http://dx.doi.org/10.1109/4434.895097.

[19] H.L. Yu, W.M. Zhen, and M.M. Shen. Extending the Power of Server Based Computing. In M. Bubak, G. D. van Albada, P. Sloot, and J. Dongarra, editors, *Computational Science - ICCS 2004*, volume 3038 of *Lecture Notes in Computer Science*, pages 242–249. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22116-6. doi: 10.1007/978-3-540-24688-6_34. URL http://dx.doi.org/10.1007/978-3-540-24688-6_34.

[20] R. Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0130137847.

[21] J. D. Sloan. *High Performance Linux Clusters: With OSCAR, Rocks, openMosix, and MPI*. O'Reilly Media, Inc., 2004. ISBN 0596005709.

[22] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11:115–128, 1996.

[23] Daniel Minoli. *A Networking Approach to Grid Computing*. Wiley-Interscience, 2004. ISBN 0471687561.

[24] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, August 2001. ISSN 1094-3420. doi: 10.1177/109434200101500302. URL http://dx.doi.org/10.1177/109434200101500302.

[25] Ian T. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *J. Comput. Sci. Technol.*, 21(4):513–520, 2006.

[26] The Globus Security Team 1. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective, 2004.

[27] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing As the 5th Utility. *Future Gener. Comput. Syst.*, 25(6):599–616, June 2009. ISSN 0167-739X. doi: 10.1016/j.future.2008.12.001. URL http://dx.doi.org/10.1016/j.future.2008.12.001.

[28] B. Hayes. Cloud Computing. *Commun. ACM*, 51(7):9–11, July 2008. ISSN 0001-0782. doi: 10.1145/1364782.1364786. URL http://doi.acm.org/10.1145/1364782.1364786.

[29] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl. Scientific Cloud Computing: Early Definition and Experience. In *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, HPCC '08, pages 825–830, Washington, DC, USA, 2008. IEEE

Computer Society. ISBN 978-0-7695-3352-0. doi: 10.1109/HPCC.2008.38. URL http://dx.doi.org/10.1109/HPCC.2008.38.

[30] A. Shawish and M. Salama. Cloud Computing: Paradigms and Technologies. In Fatos Xhafa and Nik Bessis, editors, *Inter-cooperative Collective Intelligence: Techniques and Applications*, volume 495 of *Studies in Computational Intelligence*, pages 39–67. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-35015-3. doi: 10.1007/978-3-642-35016-0_2. URL http://dx.doi.org/10.1007/978-3-642-35016-0_2.

[31] M. Klems, J. Nimis, and S. Tai. Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. In Christof Weinhardt, Stefan Luckner, and Jochen Stößer, editors, *Designing E-Business Systems. Markets, Services, and Networks*, volume 22 of *Lecture Notes in Business Information Processing*, pages 110–123. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01255-6. doi: 10.1007/978-3-642-01256-3_10. URL http://dx.doi.org/10.1007/978-3-642-01256-3_10.

[32] C. Hewitt. ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing. *IEEE Internet Computing*, 12(5):96–99, September 2008. ISSN 1089-7801. doi: 10.1109/MIC.2008.107. URL http://dx.doi.org/10.1109/MIC.2008.107.

[33] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[34] A. Tikotekar, G. Vallée, T. Naughton, H. Ong, C. Engelmann, S. L. Scott, and A. M. Filippi. Effects of Virtualization on a Scientific Application Running a Hyperspectral Radiative Transfer Code on Virtual Machines. In *Proceedings of the 2Nd Workshop on System-level Virtualization for High Performance Computing*, HPCVirt '08, pages 16–23, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-120-0. doi: 10.1145/1435452.1435455. URL http://doi.acm.org/10.1145/1435452.1435455.

[35] R. Buyya, C. S. Yeo, and S. Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services As Computing Utilities. In *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, HPCC '08, pages 5–13, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3352-0. doi: 10.1109/HPCC.2008.172. URL http://dx.doi.org/10.1109/HPCC.2008.172.

[36] P. M. Mell and T. Grance. SP 800-145. The NIST Definition of Cloud Computing. Technical report, Gaithersburg, MD, United States, 2011.

[37] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A Break in the Clouds: Towards a Cloud Definition. *SIGCOMM Comput. Commun. Rev.*, 39(1): 50–55, December 2008. ISSN 0146-4833. doi: 10.1145/1496091.1496100. URL http://doi.acm.org/10.1145/1496091.1496100.

[38] R. Buyya, R. Ranjan, and R. N. Calheiros. InterCloud: Utility-oriented Federation of Cloud Computing Environments for Scaling of Application Services. In *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, pages 13–31, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13118-2, 978-3-642-13118-9. doi: 10.1007/978-3-642-13119-6_2. URL http://dx.doi.org/10.1007/978-3-642-13119-6_2.

[39] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1519-7. doi: 10.1145/2342509.2342513. URL http://doi.acm.org/10.1145/2342509.2342513.

[40] I. Stojmenovic. Machine-to-Machine Communications With In-Network Data Aggregation, Processing, and Actuation for Large-Scale Cyber-Physical Systems. *Internet of Things Journal, IEEE*, 1(2):122–128, April 2014. ISSN 2327-4662. doi: 10.1109/JIOT.2014.2311693.

[41] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran. MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, pages 183–194, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1758-0. doi: 10.1145/2488222.2488265. URL http://doi.acm.org/10.1145/2488222.2488265.

[42] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, pages 15–20, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2180-8. doi: 10.1145/2491266.2491270. URL http://doi.acm.org/10.1145/2491266.2491270.

[43] H. Madsen, G. Albeanu, B. Burtschy, and F.L. Popentiu-Vladicescu. Reliability in the utility computing era: Towards reliable Fog computing. In *Systems, Signals*

and Image Processing (IWSSIP), 2013 20th International Conference on, pages 43–46, July 2013. doi: 10.1109/IWSSIP.2013.6623445.

[44] N. Antonopoulos and L. Gillam. *Cloud Computing: Principles, Systems and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 1849962405, 9781849962407.

[45] T. Erl, R. Puttini, and Z. Mahmood. *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2013. ISBN 0133387526, 9780133387520.

[46] F. M. Aymerich, G. Fenu, and S. Surcis. A real time financial system based on grid and cloud computing. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pages 1219–1220, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-166-8. doi: 10.1145/1529282.1529555. URL http://doi.acm.org/10.1145/1529282.1529555.

[47] Amazon EC2 web site. URL http://aws.amazon.com/ec2/.

[48] Rackspace web site. URL http://www.rackspace.com/.

[49] K. Keahey and T. Freeman. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In *Cloud Computing and Its Applications 2008 (CCA-08)*, Chicago, IL, October 2008.

[50] Microsoft Azure web site. URL http://azure.microsoft.com/it-it/.

[51] Google App Engine web site. URL http://code.google.com/appengine/.

[52] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292)*. CreateSpace Independent Publishing Platform, USA, 2012. ISBN 1478168021, 9781478168027.

[53] M. Carroll, A. van der Merwe, and P. Kotzé. Secure cloud computing: Benefits, risks and controls. In *Information Security South Africa (ISSA), 2011*, pages 1–9, August 2011. doi: 10.1109/ISSA.2011.6027519.

[54] P. Vitti, D. Dos Santos, C. B. Westphall, C. M. Westphall, and K. Vieira. Current Issues in Cloud Computing Security and Management. *SECURWARE 2014*, page 47, 2014.

[55] Qilin Li and M. Zhou. The Survey and Future Evolution of Green Computing. In *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, pages 230–233, Aug 2011. doi: 10.1109/GreenCom.2011.47.

[56] The Climate Group. SMART 2020: Enabling the low carbon economy in the information age. Technical report, 2008. URL http://www.smart2020.org/_assets/files/02_Smart2020Report.pdf.

[57] EPA. Report to Congress on Server and Data Center Energy Efficiency. Technical report, U.S. Environmental Protection Agency, 2007. URL https://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.

[58] Y. Courtois, D. McPhee, and J. Rerolle. Carbon footprint stomps on firm value. Technical report, 2013. URL https://www.kpmg.com/Global/en/IssuesAndInsights/ArticlesPublications/Documents/carbon-footprint-stomps-value-v5.pdf.

[59] NRDC. Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers. Technical report, 2014. URL https://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf.

[60] The Green Grid. The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE. Technical report, 2008. URL https://www.eni.com/green-data-center/it_IT/static/pdf/Green_Grid_DC.pdf.

[61] E. Jaureguialzo. PUE: The Green Grid metric for evaluating the energy efficiency in DC (Data Center). Measurement method using the power demand. In *Telecommunications Energy Conference (INTELEC), 2011 IEEE 33rd International*, pages 1–8, Oct 2011. doi: 10.1109/INTLEC.2011.6099718.

[62] N. Rasmussen. Electrical Efficiency Measurement for Data Centers. Technical report, Schneider Electric, 2011. URL http://www.apcmedia.com/salestools/NRAN-72754V/NRAN-72754V_R2_EN.pdf?sdirect=true.

[63] The Green Grid. A Framework for Data Center Energy Productivity. Technical report, 2008. URL http://www.thegreengrid.org/~/media/WhitePapers/WhitePaper13FrameworkforDataCenterEnergyProductivity5908.ashx?lang=en.

[64] The Green Grid. Carbon Usage Effectiveness (CUE): A Green Grid Data Center Sustainability Metric. Technical report, 2010. URL http://www.thegreengrid.org/~/media/WhitePapers/CarbonUsageEffectivenessWhitePaper20101202.ashx.

[65] G. L. Tsafack Chetsa, L. Lefèvre, J. M. Pierson, P. Stolf, and G. Da Costa. Exploiting performance counters to predict and improve energy performance of

HPC systems. *Future Generation Computer Systems*, 36:87 – 298, 2014. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2013.07.010. URL http://www.sciencedirect.com/science/article/pii/S0167739X13001556.

[66] G. Da Costa, M. D. de Assunção, J. Gelas, Y. Georgiou, L. Lefèvre, A. Orgerie, J. Pierson, O. Richard, and A. Sayah. Multi-facet Approach to Reduce Energy Consumption in Clouds and Grids: The GREEN-NET Framework. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 95–104, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0042-1. doi: 10.1145/1791314.1791329. URL http://doi.acm.org/10.1145/1791314.1791329.

[67] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein. Dynamic Energy-aware Capacity Provisioning for Cloud Computing Environments. In *Proceedings of the 9th International Conference on Autonomic Computing*, ICAC '12, pages 145–154, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1520-3. doi: 10.1145/2371536.2371562. URL http://doi.acm.org/10.1145/2371536.2371562.

[68] H. Qian and D. Medhi. Server Operational Cost Optimization for Cloud Computing Service Providers over a Time Horizon. In *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE'11, pages 4–4, Berkeley, CA, USA, 2011. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1972422.1972427.

[69] H. Qian and D. Medhi. Data Center Resource Management with Temporal Dynamic Workload. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 948–954, May 2013.

[70] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and Performance Management of Virtualized Computing Environments Via Lookahead Control. In *Proceedings of the 2008 International Conference on Autonomic Computing*, ICAC '08, pages 3–12, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3175-5. doi: 10.1109/ICAC.2008.31. URL http://dx.doi.org/10.1109/ICAC.2008.31.

[71] B. Guenter, N. Jain, and C. Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *INFOCOM, 2011 Proceedings IEEE*, pages 1332–1340, April 2011. doi: 10.1109/INFCOM.2011.5934917.

[72] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware Server Provisioning and Load Dispatching for Connection-intensive Internet Services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design*

and Implementation, NSDI'08, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association. ISBN 111-999-5555-22-1. URL http://dl.acm.org/citation.cfm?id=1387589.1387613.

[73] L. Lefèvre and A. Orgerie. Designing and Evaluating an Energy Efficient Cloud. *J. Supercomput.*, 51(3):352–373, March 2010. ISSN 0920-8542. doi: 10.1007/s11227-010-0414-2. URL http://dx.doi.org/10.1007/s11227-010-0414-2.

[74] M.F. Zhani, Qi Zhang, G. Simon, and R. Boutaba. VDC Planner: Dynamic migration-aware Virtual Data Center embedding for clouds. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 18–25, May 2013.

[75] N. Kumari, A. Shah, C. Bash, Yuan Chen, Zhenhua Liu, Zhikui Wang, T. Cader, M. Slaby, D. Cepulis, C. Felix, A. Aviles, and M. Figueroa. Optimizing data center energy efficiency via ambient-aware IT workload scheduling. In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2012 13th IEEE Intersociety Conference on*, pages 539–544, May 2012. doi: 10.1109/ITHERM.2012.6231477.

[76] E. Pakbaznia and M. Pedram. Minimizing Data Center Cooling and Server Power Costs. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '09, pages 145–150, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-684-7. doi: 10.1145/1594233.1594268. URL http://doi.acm.org/10.1145/1594233.1594268.

[77] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures. *Communications Surveys Tutorials, IEEE*, 13(2):223–244, Second 2011. ISSN 1553-877X. doi: 10.1109/SURV.2011.071410.00073.

[78] X. Wang, A. V. Vasilakos, M. Chen, Y. Liu, and T. T. Kwon. A Survey of Green Mobile Networks: Opportunities and Challenges. *Mob. Netw. Appl.*, 17(1):4–20, February 2012. ISSN 1383-469X. doi: 10.1007/s11036-011-0316-4. URL http://dx.doi.org/10.1007/s11036-011-0316-4.

[79] G. Cook, T. Dowdall, D. Pomerantz, and Y. Wang. Clicking Clean: How Companies are Creating the Green Internet. Technical report, Greenpeace Inc., 2014.

[80] P. Donadio, G. B. Fioccola, R. Canonico, and G. Ventre. Combining IT and Network Orchestration for Green Clouds. In *Proceedings of the 2014 International Conference on Intelligent Networking and Collaborative Systems*, INCOS '14, pages 539–544, Washington, DC, USA, 2014. IEEE Computer Society. ISBN

978-1-4799-6387-4. doi: 10.1109/INCoS.2014.54. URL http://dx.doi.org/10.1109/INCoS.2014.54.

[81] A. Farrel, J.-P. Vasseur, and J. Ash. A Path Computation Element (PCE)-based architecture. RFC 4655 (Proposed Standard), August 2006. URL http://www.ietf.org/rfc/rfc4655.txt.

[82] M. Pickavet, W. Vereecken, S. Demeyer, P. Audenaert, B. Vermeulen, C. Develder, D. Colle, B. Dhoedt, and P. Demeester. Worldwide energy needs for ICT: the rise of power-aware networking. In *Advanced Networks and Telecommunication Systems, 2008. ANTS '08. 2nd International Symposium on*, pages 1 – 3, December 2008. doi: 10.1109/ANTS.2008.4937762.

[83] S. Avallone and G. Ventre. Energy efficient online routing of flows with additive constraints. *Computer Networks*, 56(10):2368 – 2382, 2012. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2012.03.011. URL http://www.sciencedirect.com/science/article/pii/S1389128612001041. Green communication networks.

[84] L. T. Kou and G. Markowsky. Multidimensional Bin Packing Algorithms. *IBM J. Res. Dev.*, 21(5):443–448, September 1977. ISSN 0018-8646. doi: 10.1147/rd.215.0443. URL http://dx.doi.org/10.1147/rd.215.0443.

[85] A. Beloglazov and R. Buyya. Energy Efficient Resource Management in Virtualized Cloud Data Centers. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 826–831, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4039-9. doi: 10.1109/CCGRID.2010.46. URL http://dx.doi.org/10.1109/CCGRID.2010.46.

[86] S. Srikantaiah, A. Kansal, and F. Zhao. Energy Aware Consolidation for Cloud Computing. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 10–10, Berkeley, CA, USA, 2008. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855610.1855620.

[87] Wayne D. Grover. *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003. ISBN 013494576X.

[88] B. Mukherjee. WDM Optical Communication Networks: Progress and Challenges. *Selected Areas in Communications, IEEE Journal on*, 18(10):1810–1824, Oct 2000. ISSN 0733-8716. doi: 10.1109/49.887904.

[89] W. Van Heddeghem, M. De Groote, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester. Energy-efficiency in Telecommunications Networks: Link-by-link Versus End-to-end Grooming. In *Proceedings of the 14th Conference on Optical Network Design and Modeling*, ONDM'10, pages 48–53, Piscataway, NJ, USA, 2010. IEEE Press. ISBN 978-1-4244-5973-5. URL http://dl.acm.org/citation.cfm?id=1834075.1834086.

[90] ITU-t recommendation g.872: Architecture of optical transport networks. Technical report, International Telecommunication Union, November 2001. URL http://www.itu.int/rec/T-REC-G.872/en.

[91] ITU-t recommendation g.8080/y.1304: Architecture for the automatic switched optical network (ason). Technical report, International Telecommunication Union, November 2001. URL http://www.itu.int/rec/T-REC-G.8080/en.

[92] E. Mannie. Generalized Multi-Protocol Switching (GMPLS) Architecture. RFC 3945 (Proposed Standard), October 2004. URL https://tools.ietf.org/html/rfc3945.

[93] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001. URL http://www.ietf.org/rfc/rfc3031.txt. Updated by RFCs 6178, 6790.

[94] E. Oki, S. Kohei, D. Shimazaki, N. Yamanaka, W. Imajuku, and Y. Takigawa. Dynamic Multilayer Routing Schemes in GMPLS-based IP + Optical Networks. *Communications Magazine, IEEE*, 43(1):108–114, January 2005. ISSN 0163-6804. doi: 10.1109/MCOM.2005.1381883.

[95] K. Ishiguro, V. Manral, A. Davey, and A. Lindem. Traffic Engineering Extensions to OSPF Version 3. RFC 5329 (Proposed Standard), September 2008. URL http://www.ietf.org/rfc/rfc5329.txt.

[96] H. Smit and T. Li. Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE). RFC 3784 (Informational), June 2004. URL http://www.ietf.org/rfc/rfc3784.txt. Obsoleted by RFC 5305, updated by RFC 4205.

[97] L. Berger. Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions. RFC 3473 (Proposed Standard), January 2003. URL http://www.ietf.org/rfc/rfc3473.txt. Updated by RFCs 4003, 4201, 4420, 4783, 4874, 4873, 4974, 5063, 5151, 5420, 6002, 6003, 6780.

[98] P. Ashwood-Smith and L. Berger. Generalized Multi-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions. RFC 3472 (Proposed Standard), January 2003. URL http://www.ietf.org/rfc/rfc3472.txt. Updated by RFCs 3468, 4201.

[99] J. Lang. Link Management Protocol (LMP). RFC 4204 (Proposed Standard), October 2005. URL http://www.ietf.org/rfc/rfc4204.txt. Updated by RFC 6898.

[100] M. Yannuzzi, X. Masip-Bruin, S. Sanchez, J. Domingo-Pascual, A. Oreda, and A. Sprintson. On the Challenges of Establishing Disjoint QoS IP/MPLS Paths Across Multiple Domains. *Communications Magazine, IEEE*, 44(12):60 – 66, Dec. 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.273101.

[101] P. Castoldi, F. Cugini, L. Valcarenghi, N. Sambo, E. Le Rouzic, M. J. Poirrier, N. Andriolli, F. Paolucci, and A. Giorgetti. Centralized vs. Distributed Approaches for Encompassing Physical Impairments in Transparent Optical Networks. In *Proceedings of the 11th International IFIP TC6 Conference on Optical Network Design and Modeling*, ONDM'07, pages 68 – 77. Springer-Verlag, 2007.

[102] M. Chamania and A. Jukan. A Survey of Inter-Domain Peering and Provisioning Solutions for the Next Generation Optical Networks. *Communications Surveys Tutorials, IEEE*, 11(1):33 – 51, First 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090104.

[103] F. Aslam, Z.A. Uzmi, and A. Farrel. Interdomain Path Computation: Challenges and Solutions for Label Switched Networks. *Communications Magazine, IEEE*, 45 (10):94 – 101, October 2007. ISSN 0163-6804. doi: 10.1109/MCOM.2007.4342830.

[104] I.T. Okumus, Junseok Hwang, H.A. Mantar, and S.J. Chaplin. Inter-domain LSP setup using bandwidth management points. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 1, pages 7 – 11, 2001. doi: 10.1109/GLOCOM.2001.965070.

[105] J.P. Vasseur and J.L. Le Roux. Path Computation Element (PCE) Communication Protocol (PCEP). RFC 5440 (Proposed Standard), March 2009. URL https://tools.ietf.org/html/rfc5440.

[106] E. Feller, C. Rohr, D. Margery, and C. Morin. Energy Management in IaaS Clouds: A Holistic Approach. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, pages 204–212, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4755-8. doi: 10.1109/CLOUD.2012.50. URL http://dx.doi.org/10.1109/CLOUD.2012.50.

[107] B. Claise and J. Parello. EMAN: Energy-Management Activities at the IETF. *IEEE Internet Computing*, 17(3):80 – 82, 2013. ISSN 1089-7801. doi: 10.1109/MIC.2013.49.

[108] C. Ghribi, M. Hadji, and D. Zeghlache. Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 671–678, May 2013. doi: 10.1109/CCGrid.2013.89.

[109] A. Murtazaev and S. Oh. Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing. *IETE Technical Review*, 28(3): 212–231, 2011. doi: 10.4103/0256-4602.81230. URL http://www.tandfonline.com/doi/abs/10.4103/0256-4602.81230.

[110] N. Bobroff, A. Kochut, and K. Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pages 119–128, May 2007. doi: 10.1109/INM.2007.374776.

[111] F. Hermenier, X. Lorca, J. Menaud, G. Muller, and J. Lawall. Entropy: A Consolidation Manager for Clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 41–50, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-375-4. doi: 10.1145/1508293.1508300. URL http://doi.acm.org/10.1145/1508293.1508300.

[112] M. Marzolla, O. Babaoglu, and F. Panzieri. Server consolidation in Clouds through gossiping. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–6, June 2011. doi: 10.1109/WoWMoM.2011.5986483.

[113] A. Beloglazov and R. Buyya. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints. *IEEE Trans. Parallel Distrib. Syst.*, 24(7):1366–1379, July 2013. ISSN 1045-9219. doi: 10.1109/TPDS.2012.240. URL http://dx.doi.org/10.1109/TPDS.2012.240.

[114] G. Dhiman, G. Marchetti, and T. Rosing. vGreen: A System for Energy Efficient Computing in Virtualized Environments. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '09, pages 243–248, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-684-7. doi: 10.1145/1594233.1594292. URL http://doi.acm.org/10.1145/1594233.1594292.

[115] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. *Future Gener. Comput. Syst.*, 28(5):755–768, May 2012. ISSN 0167-739X. doi: 10.1016/j.future.2011.04.017. URL http://dx.doi.org/10.1016/j.future.2011.04.017.

[116] A. Beloglazov and R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation: Practice & Experience*, 24(13):1397 – 1420, September 2012. ISSN 1532-0626. doi: 10.1002/cpe.1867.

[117] A. Verma, G. Dasgupta, T. K.r Nayak, P. De, and R. Kothari. Server Workload Analysis for Power Minimization Using Consolidation. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, USENIX'09, pages 28–28, Berkeley, CA, USA, 2009. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855807.1855835.

[118] Network functions virtualization: an introduction, benefits, enablers, challenges and call for action. Issue 3, October 2014. URL https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf.

[119] N.M. Chowdhury and R. Boutaba. A Survey of Network Virtualization. *Comput. Netw.*, 54(5):862–876, April 2010. ISSN 1389-1286. doi: 10.1016/j.comnet.2009.10.017. URL http://dx.doi.org/10.1016/j.comnet.2009.10.017.

[120] J. F. Riera et al. Virtual infrastructures as a Service Enabling Converged Optical Networks and Data Centres. *Optical Switching and Networking*, 14, Part 3(0):197 – 208, 2014. ISSN 1573-4277. doi: 10.1016/j.osn.2014.05.01.

[121] M. Fiorani, S. A Aleksic, and M. Casoni. Hybrid Optical Switching for Data Center Networks. *Electrical and Computer Engineering*, 2014, February 2014. doi: 10.1155/2014/139213.

[122] R. Bifulco, R. Canonico, G. Ventre, and V. Manetti. Transparent migration of virtual infrastructures in large datacenters for Cloud computing. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 179 – 184, June 2011. doi: 10.1109/ISCC.2011.5984013.

[123] G. Koslovski, Wai-Leong Yeow, C. Westphal, T.T. Huu, J. Montagnat, and P. Vicat-Blanc. Reliability Support in Virtual Infrastructures. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 49 – 58, November 2010. doi: 10.1109/CloudCom.2010.23.

[124] J. M. Pierson and H. Casanova. On the Utility of DVFS for Power-aware Job Placement in Clusters. In *Proceedings of the 17th International Conference on Parallel Processing - Volume Part I*, Euro-Par'11, pages 255–266, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23399-9. URL http://dl.acm.org/citation.cfm?id=2033345.2033372.

[125] L. Jiong, N. K. Jha, and L. S. Peh. Simultaneous Dynamic Voltage Scaling of Processors and Communication Links in Real-Time Distributed Embedded Systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15 (4):427–437, April 2007. ISSN 1063-8210. doi: 10.1109/TVLSI.2007.893660.

[126] C. Xian, Y. H. Lu, and Z. Li. Dynamic Voltage Scaling for Multitasking Real-Time Systems With Uncertain Execution Time. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8):1467–1478, August 2008. ISSN 0278-0070. doi: 10.1109/TCAD.2008.925778.

[127] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra. Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration. In *Parallel Processing (ICPP), 2010 39th International Conference on*, pages 228–237, September 2010. doi: 10.1109/ICPP.2010.30.

[128] H. D. Saunders. The Khazzoom-Brookes Postulate and Neoclassical Growth. *The Energy Journal*, 13(4):131–148, 1992. ISSN 01956574, 19449089. URL http://www.jstor.org/stable/41322471.

[129] B. Aksanli, T. S. Rosing, and I. Monga. Benefits of green energy and proportionality in high speed wide area networks connecting data centers. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 175–180, March 2012. doi: 10.1109/DATE.2012.6176458.

[130] M. Lemay, K. Nguyen, B. St. Arnaud, and M. Cheriet. Toward a Zero-Carbon Network: Converging Cloud Computing and Network Virtualization. *IEEE Internet Computing*, 16(6):51–59, November 2012. ISSN 1089-7801. doi: 10.1109/MIC.2011.128.

[131] J. Wang, A. M. Fagertun, S. Ruepp, and L. Dittmann. Analysis of energy efficiency in dynamic optical networks employing solar energy sources. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 590–593, January 2013. doi: 10.1109/ICCNC.2013.6504152.

[132] P. Donadio, G.B. Fioccola, R. Canonico, and G. Ventre. A PCE-based Architecture for the Management of. Virtualized Infrastructures. In *Cloud Networking*

(CloudNet), 2014 IEEE 3rd International Conference on, pages 223 – 228, October 2014. doi: 10.1109/CloudNet.2014.6968996.

[133] A. Moreira, J. Moreira, D. Sadok, A. Callado, M. Rodrigues, M. Neves, V. Souza, and P.P. Karlsson. A case for virtualization of Content Delivery Networks. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 3178 – 3189, December 2011. doi: 10.1109/WSC.2011.6148016.

[134] C. Papagianni, A. Leivadeas, and S. Papavassiliou. A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment. *Dependable and Secure Computing, IEEE Transactions on*, 10(5):287 – 300, September 2013. ISSN 1545-5971. doi: 10.1109/TDSC.2013.12.

[135] Network Functions Virtualization: use cases, October 2013. URL http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf.

[136] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs. Network function placement for NFV chaining in packet/optical data centers. In *Optical Communication (ECOC), 2014 European Conference on*, pages 1–3, September 2014. doi: 10.1109/ECOC.2014.6963925.

[137] P. Shuping, R. Nejabati, and D. Simeonidou. Role of Optical Network Virtualization in. Cloud Computing. *Optical Communications and Networking, IEEE/OSA Journal of*, 5(10):A162–A170, October 2013. ISSN 1943-0620. doi: 10.1364/JOCN.5.00A162.

[138] A. S. Reaz, V. Ramamurthi, M. Tornatore, and B. Mukherjee. Cloud-Integrated WOBAN: an Offloading-Enabled Architecture for Service-Oriented Access Networks. *Computer Networks*, 68(0):5 – 19, 2014. ISSN 1389-1286. doi: 10.1016/j.comnet.2013.12.003.

[139] Z.A. Mann. Allocation of Virtual Machines in Cloud Data Centers - A Survey of Problem Models and Optimization Algorithms. http://www.cs.bme.hu/~mann/publications/Preprints/Mann_VM_Allocation_Survey.pdf, 2015.

[140] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, and R. Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice & Experience*, 41(1):23 – 50, January 2011. ISSN 0038-0644. doi: 10.1002/spe.995.

[141] D. Kliazovich, P. Bouvry, and S. Khan. GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers. *The Journal of Supercomputing*, 62(3):1263 – 1283, 2012. doi: 10.1007/s11227-010-0504-1.

[142] N.M.M.K. Chowdhury and R. Boutaba. Network virtualization: state of the art and research challenges. *Communications Magazine, IEEE*, 47(7):20 – 26, July 2009. ISSN 0163-6804. doi: 10.1109/MCOM.2009.5183468.

[143] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual Network Embedding: A Survey. *Communications Surveys Tutorials, IEEE*, 15 (4):1888 – 1906, Fourth 2013. ISSN 1553-877X. doi: 10.1109/SURV.2013.013013.00155.

[144] M.G. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba. On Tackling Virtual Data Center Embedding Problem. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 177 – 184, May 2013.

[145] Xiong Fu and C. Zhou. Virtual machine selection and placement for dynamic consolidation in cloud computing environment. *Frontiers of Computer Science*, 9 (2):322 – 330, February 2015. ISSN 2095-2228. doi: 10.1007/s11704-015-4286-8.

[146] M.F. Bari, M.F. Zhani, Qi Zhang, R. Ahmed, and R. Boutaba. CQNCR: Optimal VM Migration Planning in Cloud. Data Centers. In *Networking Conference, 2014 IFIP*, pages 1 – 9, June 2014. doi: 10.1109/IFIPNetworking.2014.6857120.

[147] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.M. Pierson, and A.V. Vasilakos. Cloud Computing: Survey on Energy Efficiency. *ACM Computing Surveys*, 47(2):33:1 – 33:36, Dec. 2014. doi: 10.1145/2656204.

[148] S. Ricciardi, J. Wang, F. Palmieri, D. Careglio, A. Manolova, and G. Santos-Boada. Eco-sustainable routing in optical networks. *Photonic Network Communications*, 26(2-3):140 – 149, 2013. doi: 10.1007/s11107-013-0416-0.

[149] P. Donadio, S. Russo, R. Canonico, and G. Ventre. O-gene: towards an open green network control plane. In *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pages 173 – 178, July 2013. doi: 10.1109/ISCC.2013.6754941.

[150] D. Tafani, B. Kantarci, H. T. Mouftah, C. McArdle, and L. P. Barry. Towards Energy Efficiency for Cloud Computing Services. In *Communication Infrastructures for Cloud Computing*, pages 306 – 328, Hershey, PA, 2014. IGI Global. doi: 10.4018/978-1-4666-4522-6.ch014.

[151] K. Bilal, S. U. Khan, S. A. Madani, K. Hayat, M. I. Khan, N. Min-Allah, J. Kolodziej, L. Wang, S. Zeadally, and D. Chen. A Survey on Green Communications Using Adaptive Link Rate. *Cluster Computing*, 16(3):575–589, September 2013. ISSN 1386-7857. doi: 10.1007/s10586-012-0225-8. URL http://dx.doi.org/10.1007/s10586-012-0225-8.

[152] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy Proportional Datacenter Networks. *SIGARCH Comput. Archit. News*, 38(3):338–347, June 2010. ISSN 0163-5964. doi: 10.1145/1816038.1816004. URL http://doi.acm.org/10.1145/1816038.1816004.

[153] Y. Shang, D. Li, J. Zhu, and M. Xu. On the Network Power Effectiveness of Data Center Architectures. *Computers, IEEE Transactions on*, 64(11):3237–3248, November 2015. ISSN 0018-9340. doi: 10.1109/TC.2015.2389808.

[154] P. Mahadevan, S. Banerjee, P. Sharma, A. Shah, and P. Ranganathan. On energy efficiency for enterprise and data center networks. *Communications Magazine, IEEE*, 49(8):94–100, August 2011. ISSN 0163-6804. doi: 10.1109/MCOM.2011.5978421.

[155] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J.A. Maestro. IEEE 802.3az: the road to energy efficient ethernet. *Communications Magazine, IEEE*, 48(11):50–56, November 2010. ISSN 0163-6804. doi: 10.1109/MCOM.2010.5621967.

[156] A. Carrega, S. Singh, R. Bruschi, and R. Bolla. Traffic merging for energy-efficient datacenter networks. In *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2012 International Symposium on*, pages 1–5, July 2012.

[157] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding Data Center Traffic Characteristics. *SIGCOMM Comput. Commun. Rev.*, 40(1):92–99, January 2010. ISSN 0146-4833. doi: 10.1145/1672308.1672325. URL http://doi.acm.org/10.1145/1672308.1672325.

[158] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman. VMFlow: Leveraging VM Mobility to Reduce Network Power Costs in Data Centers. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 198–211, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-20756-3. URL http://dl.acm.org/citation.cfm?id=2008780.2008800.

[159] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 17–17, Berkeley, CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855711.1855728.

[160] Maruti Gupta and Suresh Singh. Greening of the Internet. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 19–26, New York, NY, USA, 2003. ACM. ISBN 1-58113-735-4. doi: 10.1145/863955.863959. URL http://doi.acm.org/10.1145/863955.863959.

[161] L. Chiaraviglio, M. Mellia, and F. Neri. Reducing Power Consumption in Backbone Networks. In *Proceedings of the 2009 IEEE International Conference on Communications*, ICC'09, pages 2298–2303, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3434-3. URL http://dl.acm.org/citation.cfm?id=1817271.1817698.

[162] L. Gyarmati and T. A. Trinh. How Can Architecture Help to Reduce Energy Consumption in Data Center Networking? In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 183–186, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0042-1. doi: 10.1145/1791314.1791343. URL http://doi.acm.org/10.1145/1791314.1791343.

[163] R.N. Calheiros, R. Buyya, and C.A.F. De Rose. A Heuristic for Mapping Virtual Machines and Links in Emulation Testbeds. In *Parallel Processing, 2009. ICPP '09. International Conference on*, pages 518–525, September 2009. doi: 10.1109/ICPP.2009.7.

[164] R. Basmadjian, H. Meer, R. Lent, and G. Giuliani. Cloud computing and its interest in saving energy: the use case of a private cloud. *Journal of Cloud Computing*, 1(1):1–25, 2012. ISSN 2192-113X. doi: 10.1186/2192-113X-1-5. URL http://dx.doi.org/10.1186/2192-113X-1-5.

[165] D. Quan, R. Basmadjian, H. Meer, R. Lent, T. Mahmoodi, D. Sannelli, F. Mezza, L. Telesca, and C. Dupont. *Computer and Information Sciences II: 26th International Symposium on Computer and Information Sciences*. Springer London, London, 2012. ISBN 978-1-4471-2155-8. doi: 10.1007/978-1-4471-2155-8_16. URL http://dx.doi.org/10.1007/978-1-4471-2155-8_16.

[166] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Dang, and K. Pentikousis. Energy-Efficient Cloud Computing. *Comput. J.*, 53(7):1045–1051, September 2010. ISSN 0010-4620. doi: 10.1093/comjnl/bxp080. URL http://dx.doi.org/10.1093/comjnl/bxp080.

[167] L. Nonde, T. E. H. El-Gorashi, and J. M. H. Elmirghani. Energy Efficient Virtual Network Embedding for Cloud Networks. *Journal of Lightwave Technology*, 33(9):1828–1849, May 2015. ISSN 0733-8724. doi: 10.1109/JLT.2014.2380777.

[168] B. Kantarci and H. T. Mouftah. Energy-Efficiency in a Cloud Computing Backbone. In *Communication Infrastructures for Cloud Computing*, pages 283 – 305, Hershey, PA, 2014. IGI Global. doi: 10.4018/978-1-4666-4522-6.ch013.

[169] S. Ricciardi, F. Palmieri, U. Fiore, A. Castiglione, and G. Santos-Boada. Modeling energy consumption in next-generation wireless access-over-wdm networks with hybrid power sources. *Mathematical and Computer Modelling*, 58(5–6):1389 – 1404, 2013. ISSN 0895-7177. doi: http://dx.doi.org/10.1016/j.mcm.2012.12.004. URL http://www.sciencedirect.com/science/article/pii/S0895717712003482.

[170] DRAGON: Dynamic Resource Allocation via GMPLS Optical Networks. URL http://dragon.maxgigapop.net.

[171] F. Farahnakian, A. Ashraf, P. Liljeberg, T. Pahikkala, J. Plosila, I. Porres, and H. Tenhunen. Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pages 104–111, June 2014. doi: 10.1109/CLOUD.2014.24.

[172] MATLAB and Statistics Toolbox. *Release 2013b)*. The MathWorks Inc., Natick, Massachusetts, United States, 2013.

[173] G. P. Agrawal. *Optical Fibers*, pages 24–78. John Wiley and Sons, Inc., 2011. ISBN 9780470918524. doi: 10.1002/9780470918524.ch2. URL http://dx.doi.org/10.1002/9780470918524.ch2.

[174] D. Kliazovich, P. Bouvry, and S. U. Khan. DENS: Data Center Energy-efficient Network-aware Scheduling. *Cluster Computing*, 16(1):65–75, March 2013. ISSN 1386-7857. doi: 10.1007/s10586-011-0177-4. URL http://dx.doi.org/10.1007/s10586-011-0177-4.

[175] A. Leivadeas, C. Papagianni, and S. Papavassiliou. *Going Green with the Networked Cloud: Methodologies and Assessment*, pages 351–374. John Wiley and Sons, Inc., 2015. ISBN 9781119131151. doi: 10.1002/9781119131151.ch13. URL http://dx.doi.org/10.1002/9781119131151.ch13.

[176] J. Ahn and Park H. Measurement and modeling the power consumption of router interface. In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pages 860–863, February 2014. doi: 10.1109/ICACT.2014.6779082.

[177] S. Ricciardi, J. Wang, F. Palmieri, D. Careglio, and L. Dittman. An Energy-aware Engineered Control Plane for Wavelength-routed Networks. *Trans. Emerg. Telecommun. Technol.*, 26(2):231–249, February 2015. ISSN 2161-3915. doi: 10.1002/ett.2815. URL http://dx.doi.org/10.1002/ett.2815.

[178] M. M. Hassan and E. N. Huh. *Dynamic Cloud Collaboration Platform: A Market-Oriented Approach*. Springer Publishing Company, Incorporated, 2012. ISBN 1461451450, 9781461451457.

[179] Z. Zhang, S. Su, J. Zhang, K. Shuang, and P. Xu. Energy Aware Virtual Network Embedding with Dynamic Demands. *Comput. Netw.*, 93(P3):448–459, December 2015. ISSN 1389-1286. doi: 10.1016/j.comnet.2015.09.036. URL http://dx.doi.org/10.1016/j.comnet.2015.09.036.

[180] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy. Optimal Power Allocation in Server Farms. *SIGMETRICS Perform. Eval. Rev.*, 37(1):157–168, June 2009. ISSN 0163-5999. doi: 10.1145/2492101.1555368. URL http://doi.acm.org/10.1145/2492101.1555368.

[181] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Y. Zomaya. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. *Advances in Computers*, 82:47–111, 2011. URL http://dblp.uni-trier.de/db/journals/ac/ac82.html#BeloglazovBLZ11.

[182] X. Fan, W. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-sized Computer. *SIGARCH Comput. Archit. News*, 35(2):13–23, June 2007. ISSN 0163-5964. doi: 10.1145/1273440.1250665. URL http://doi.acm.org/10.1145/1273440.1250665.

[183] S. Aiken, D. Grunwald, A.R. Pleszkun, and J. Willeke. A performance analysis of the iscsi protocol. In *Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, pages 123–134, April 2003. doi: 10.1109/MASS.2003.1194849.

[184] J. Van der Ham, P. Grosso, R. van der Pol, A. Toonk, and C. de Laat. Using the Network Description Language in Optical Networks. In *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pages 199–205, May 2007. doi: 10.1109/INM.2007.374784.

[185] T. Lehman, J. Sobieski, and B. Jabbari. DRAGON: a framework for service provisioning in heterogeneous grid networks. *Communications Magazine, IEEE*, 44(3):84–90, March 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.1607870.

[186] C. Develder, M. De Leenheer, B. Dhoedt, M. Pickavet, D. Colle, F. De Turck, and P. Demeester. Optical Networks for Grid and Cloud Computing Applications. *Proceedings of the IEEE*, 100(5):1149–1167, May 2012. ISSN 0018-9219. doi: 10.1109/JPROC.2011.2179629.

[187] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672. URL http://doi.acm.org/10.1145/1721654.1721672.

[188] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards Trusted Cloud Computing. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855533.1855536.

[189] Trusted Computing Group. TPM Main Specification Version 1.2 rev. 103. http://www.trustedcomputinggroup.org/resources/tpm_main_specification, July 2007. URL http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

[190] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou. Security and Privacy in Cloud Computing: A Survey. In *Proceedings of the 2010 Sixth International Conference on Semantics, Knowledge and Grids*, SKG '10, pages 105–112, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4189-1. doi: 10.1109/SKG.2010.19. URL http://dx.doi.org/10.1109/SKG.2010.19.

[191] P. Donadio, G. B. Fioccola, R. Canonico, and G. Ventre. Network security for Hybrid Cloud. In *Euro Med Telco Conference (EMTC), 2014*, pages 1–6, November 2014. doi: 10.1109/EMTC.2014.6996640.

[192] S. A. De Chaves, C. B. Westphall, and F. R. Lamin. SLA Perspective in Security Management for Cloud Computing. In *Proceedings of the 2010 Sixth International Conference on Networking and Services*, ICNS '10, pages 212–217, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-3969-0. doi: 10.1109/ICNS.2010.36. URL http://dx.doi.org/10.1109/ICNS.2010.36.

[193] K. Vieira, A. Schulter, C.B. Westphall, and C.M. Westphall. Intrusion Detection for Grid and Cloud Computing. *IT Professional*, 12(4):38–43, July 2010. ISSN 1520-9202. doi: 10.1109/MITP.2009.89.

[194] S.A. De Chaves, R.B. Uriarte, and C.B. Westphall. Toward an architecture for monitoring private clouds. *Communications Magazine, IEEE*, 49(12):130–137, December 2011. ISSN 0163-6804. doi: 10.1109/MCOM.2011.6094017.

[195] K. Jackson. *OpenStack Cloud Computing Cookbook*. Packt Publishing, 2012. ISBN 1849517320, 9781849517324.