People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Saad Dahlab University

Faculty of Science

Computer Science Department

Masters memoir

To obtain the master's degree in computer science

Option: Computer Systems and Network.

Theme

Study and Implementation of Hyper converged infrastructure architecture.

Realized by:

- Yasser DILMI
- Akram HAMIDA

Supervised by:

-Ms. Meriem ARKAM

-Mr. Abdelouahab GAHAR

Host Organization: OMES Algérie

Graduation Promotion: 2019/2020

Acknowledgment

*Thanks to **Allah** for his grace, the source of our strength and courage throughout our university studies.*

*We would like to express our heartfelt thanks and our deep gratitude to **Ms. Meriem ARKAM** for supervising us in our final thesis, encouragement and for leading us in our work.*

*We also thank **OMES Algérie** for the opportunity of this internship, and **Mr. Abdelouahab GAHAR** for his help in the implementation and thorough guidance.*

*We thank the **members of the jury** for agreeing to evaluate our work.*

*A big thank you also to **our families** for their moral and financial support and for their sacrifices.*

*We cannot end without thanking **everyone** who has participated in anyway in the development of this graduation project.*

Abstract

Conventional approaches to IT cannot accommodate business agility, which is challenged to accomplish. Consequently, to the emergence of the software-defined approaches, IT organizations have moved to the forefront of the enterprise charge toward digital transformation.

In this project, we propose a hyper-converged architecture to the OMES Algérie SARL digitalization process. The main design consideration of this infrastructure is to further maximize the overall system utilization and delivery, achieving High availability of data and functionalities, backup and restoration, bandwidth control, and storage scale-out. As a consequence, to accomplish the intended objective, the combination of multiple software-defined solutions SDC for computational functions, SDS for storage virtualization to get rid of the traditional storage limitation, and SDN for easy troubleshooting and flexible network management, are needed.

At the end of this project, after benchmarking the cluster capabilities of the network controller, storage solution and compute managers. we have managed to simulate the infrastructure in a local lab with the sought requirements.

Keywords: SDC, SDN, SDS, Hyper Converged, Virtualization.

Résumé

Les approches conventionnelles de l'informatique ne peuvent pas s'adapter à l'agilité des entreprises, qui est mise à l'épreuve. Par conséquent, avec l'émergence des approches logicielles, les organisations informatiques sont passées au premier plan de la charge de l'entreprise en matière de transformation numérique.

Dans ce projet, nous proposons une architecture hyper-convergente au processus de numérisation de OMES Algérie SARL. La principale considération de cette infrastructure est de maximiser l'utilisation et la livraison du système global, en atteignant une haute disponibilité de données et de fonctionnalités, la sauvegarde et la restauration, le contrôle de la bande passante et l'extensibilité du stockage. En conséquence, pour atteindre l'objectif visé, il faut combiner plusieurs solutions logicielles : SDC pour les fonctions de calcul, SDS pour la virtualisation du stockage afin de se débarrasser de la limitation traditionnelle du stockage, et SDN pour un dépannage facile et une gestion souple du réseau.

À la fin de ce projet, après avoir évalué les capacités du contrôleur de réseau, la solution de stockage et le gestionnaire de calcul, nous avons réussi à simuler l'infrastructure dans un laboratoire local avec les exigences recherchées.

Mots Clefs : SDC, SDN, SDS, Hyper Convergée, Virtualisation.

ملخص

ليس من السهل أن تواكب المناهج التقليدية لتكنولوجيا المعلومات سرعة الأعمال في الشركات، فظهرت تدريجيا مقاربات البرمجيات المعرفة، التي مكنت مؤسسات تكنولوجيا المعلومات من الانتقال إلى التحول الرقمي.

في هذا المشروع، نقترح بنية فائقة التقارب لعملية رقمنة شركة OMES Algérie SARL. يتمثل الاعتبار الرئيسي لتصميم هذه البنية التحتية في زيادة استخدام النظام الشامل و وصوله إلى أقصى حد، وتحقيق التوافر المستمر للبيانات والوظائف، النسخ الاحتياطي والاستعادة، التحكم في سرعة التدفق، وتوسيع حيز التخزين. نتيجة لذلك، ولتحقيق الهدف المنشود، هناك حاجة إلى الجمع بين الحلول المتعددة المعروفة بمناهج البرمجيات SDC للوظائف الحسابية، و SDS للمحاكاة الافتراضية للتخزين للتخلص من قيود التخزين التقليدية، و SDN لسهولة اكتشاف الأخطاء وإصلاحها والإدارة المرنة للشبكة.

في نهاية هذا المشروع، وبعد قياس قدرات وِحدة تحكم الشبكة، وحل التخزين المقترح ومسيّر الحوسبة، تمكنا من محاكاة البنية التحتية في حاسوب محلي بالمتطلبات المطلوبة.

كلمات مفتاحية : الحوسبة المعرفة بالبرمجيات، الشبكات المعرفة بالبرمجيات، التخزين المعرف بالبرمجيات، بنية التقارب الفائق، الافتراضية.

Table of Content

Figures Table

Tables List

Glossary

SMB: Server Message Block (SMB) Protocol is a network file sharing protocol, and as implemented in Microsoft Windows is known as Microsoft SMB Protocol. The set of message packets that defines a particular version of the protocol is called a dialect. The Common Internet File System (CIFS) Protocol is a dialect of SMB.

NFS: Sun Network Filesystem (NFS) protocol provides transparent remote access to shared files across networks. The NFS protocol is designed to be portable across different machines, operating systems, network architectures, and transport protocols [1].

Bit rot: is the slow deterioration in the performance and integrity of data stored on storage media. It is also known by the names bit decay, data rot, data decay and silent corruption [2].

Replication: Data replication is the process of copying data from one location to another, the technology helps an organization possess up-to-date copies of its data in the event of a disaster [3].

Snapshot: snapshot is a copy of a storage volume, file or database as they appeared at a given point in time and are used as method of data protection. In the event of a failure, users can restore their data from the most recent snapshot before the failure [4].

Block Storage: Block storage is an approach to data storage in which each storage volume acts as an individual hard drive that is configured by the storage administrator. In the block storage model, data is saved to the storage media in fixed-sized chunks called blocks. Each block is associated with a unique address, and the address is the only metadata assigned to each block [5].

Object storage: Object storage keeps the blocks of data that make up a file together and adds all of its associated metadata to that file. But object storage also adds extended metadata to the file and eliminates the hierarchical structure used in file storage, placing everything into a flat address space, called a storage pool [6].

Object storage daemon: the object storage daemon this is the software that is responsible for providing access to the data, and it optional for one per disk or one per path, OSDs are responsible also for serving stored objects to clients, and also for peering with other OSDs for replication and recovery tasks [7].

General Introduction

Data centers are at the center of modern software technology, serving a critical role in the expanding capabilities for enterprises. Whereas businesses just a few decades ago relied on error-prone paper-and-pencil documentation methods, data centers have vastly improved the usability of data. They have enabled businesses to do much more with much less, both in terms of physical space and the time required to create and maintain data. However, data centers are positioned to play an even more important role in the advancement of technology, with new concepts entering the landscape that represents a dramatic shift in the way data centers are conceived, configured, and utilized.

IT organizations have moved to the forefront of the enterprise charge toward digital transformation. The traditional data center, also known as a “siloed” data center, relies heavily on hardware and physical servers, that are restricted by the size of the physical space in which the hardware is stored. Nonetheless, this solution became less efficient with how much the data is growing it led ITs to quickly moving from merely managing infrastructure to deliver agile, flexible, and responsive systems that enable success. To make this change, IT organizations must leverage the new infrastructure that is more efficient and agile. During a server refresh, the underlying architecture can be upgraded to hyper-converged infrastructure (HCI) to meet digital business-changing demands for new products and services.

Hyper-Converged Infrastructure is a relatively new concept that was first introduced around 2013, HCI is a software-defined IT infrastructure that virtualizes all of the elements of conventional "hardware-defined" systems, it seeks to do the same, but adds more value by throwing in software-defined solutions and doesn't place much emphasis on networking. Today hyper-converged infrastructure can come as an appliance, a reference architecture, or as software that's flexible in terms of the platform it runs on, in converged infrastructure, the server, compute and networking components remain separate and are not integrated into nodes as they are with hyper-converged systems.

Problematic

OMES Algérie is aiming to digitalize its working space and create an optimized Data Center that could be centrally and easily managed, scalable and flexible because they are using a traditional way on their premises, as a consequence, they proposed a modern solution that is more advanced than the traditional approach, this solution is the hyper-convergence.

Objective of the project

In our project we are aiming to study and implement the approach of Hyper-Converged architecture in a LAN with most of its core components, software-defined storage, software-defined compute, and software-defined Network ensuring these requirements:

- High availability of functionalities and data.
- Live migration for maintenance.
- Backups and restorations.
- Access restriction.
- Bandwidth limitation QoS.

Organization of the memoir

To facilitate reading our work, this memoir will be organized as follows:

- **Chapter 1: Hyper-Converged Infrastructure-:** in this chapter we are going to explain the story behind the hyper convergence and how software defined solutions helped datacenters evolve.
- **Chapter 2: Software Defined Network-:** we are going to explain the software defined network and present its literature review.
- **Chapter 3: Implementation and tests:** in this chapter we will demonstrate our solution, and how it may help digitalize the OMES Algérie system.

1 Chapter 1 Hyper-converged Infrastructure

1.1 Preface

Virtualization fundamentally and permanently changed IT and the data center. Today, most services are running inside virtual environments, and IT often takes a “virtualized first” approach to new application and service deployment. That is, administrators consider the virtual environment for running new applications rather than just building a new physical environment [8].

In a converged infrastructure, the server, compute and networking components remain separate and are not integrated into nodes, this relatively led to a new concept called Hyper-converged Infrastructure, HCI is a software-defined IT infrastructure that virtualizes all of the elements of conventional "hardware-defined" systems into a single node, weather its Storage arrays, Network switches or compute power, it seeks to do the same, but adds more value and doesn't place much emphasis on networking. Today hyper-converged infrastructure can come as an appliance, a reference architecture, or as software that's flexible in terms of the platform it runs on [9].

To fully understand hyper-convergence it is important to understand the evolution of traditional IT industry that led to this point, and the importance of the virtualization in creating a scalable and reliable enterprise solution [8].

1.2 Virtualization

Virtualization as a term, is the process of running a virtual instance of a computer system in a layer abstracted from the actual physical resources. Typically, it refers to running multiple OS’s on a computer system at the same time. To the applications running on VM, it will seem as if they’re on their own dedicated hardware, where the operating system, libraries, and various programs are unique to the guest virtualized system and unconnected to the host OS which sits below it [10].

There are several reasons why individuals use virtualization:

- For desktop users, the main use is to be able to run applications meant for a different operating system without having to switch computers or reboot into a different system.

- For administrators of servers, virtualization also offers the ability to run different operating systems, but perhaps, more importantly, it offers a way to segment a large system into many smaller parts, allowing the server to be used more efficiently by a number of different users or applications with different needs. It also allows for isolation, keeping programs running inside of a virtual machine safe from the processes taking place in another virtual machine on the same host [10].

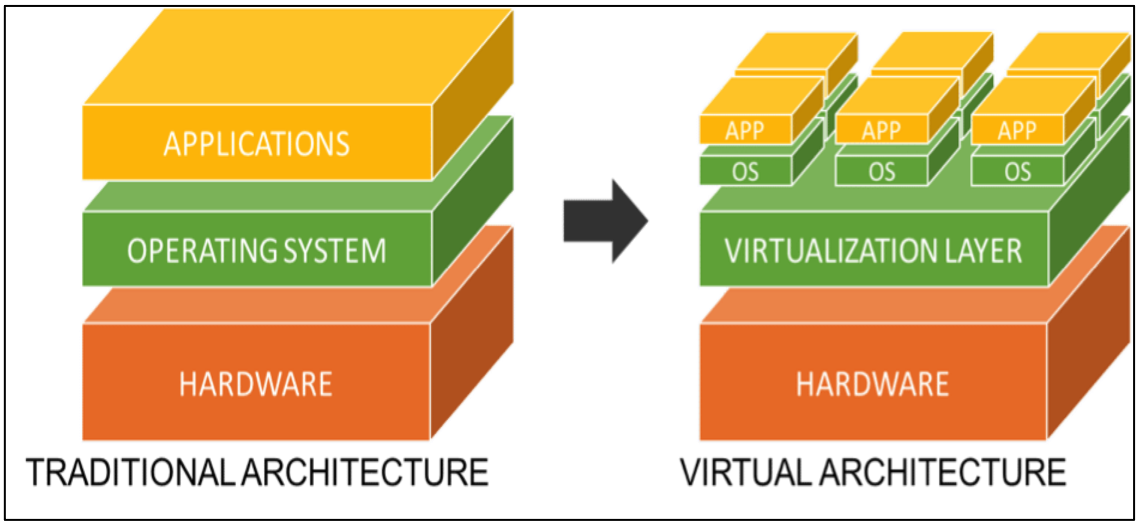


Figure 1.1 *traditional Architecture vs Virtual Architecture*

1.3 Software Defined Compute

The software defined compute made a huge impact on the information technology field, by introducing several new ways to the hardware resource management and utilization.

1.3.1 Definition

SDC is when a compute function is virtualized and abstracted from the hardware it is on, creating an operation that can be managed through a central interface. With software-defined compute, management can be moved to a central interface that sees all computing resources as one element. [11]

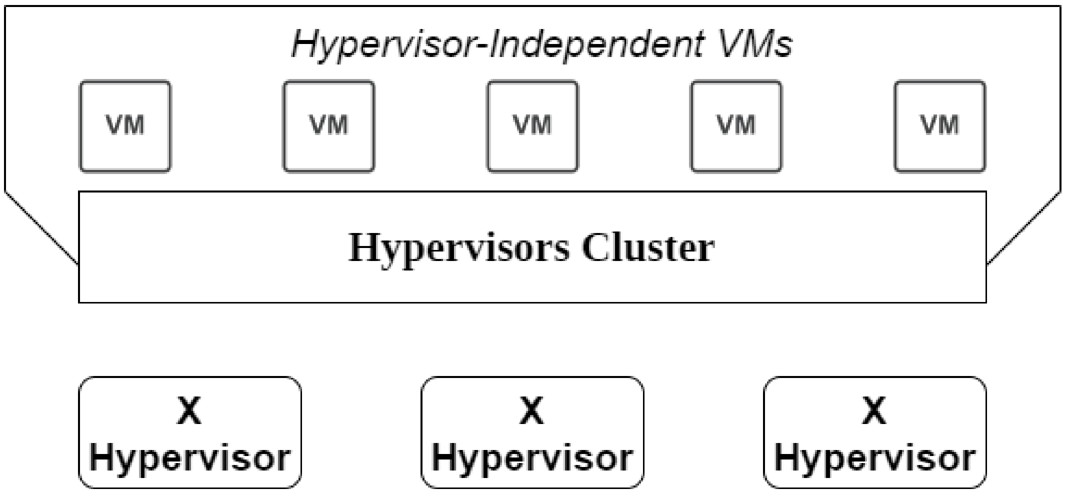


Figure 1.2 *Software Defined Compute*

1.3.2 Flexibility of SDC

Software-defined compute will pool the computational functions across any number of processing units and the workload could be spread among them instead of being assigned to one specific device. In addition, the compute functions can be moved around to different pieces of virtual infrastructure, depending on the availability of resources, without having to manually change the hardware, using virtualization technologies, the infrastructure can be broken up into resources that can be allocated on demand, thus will enable for a flexible infrastructure. [11]

By employing software-defined compute and virtualization, network managers can streamline many management processes by simplifying the data center so that it can be more easily scaled. Hardware components tend to be generic and industry-standard, so they can be easily added to satisfy demand [11].

1.3.3 Hypervisors

The hypervisor, also called a virtual machine manager, is a program that allows multiple operating systems to share a single hardware host. In appearance, each operating system has exclusivity of the processor, memory, and all other resources of the host. However, in reality, it is the hypervisor that controls the processor and host resources, alternately allocating to each operating system what it needs and ensuring that these guest systems (or virtual machines) do not interfere with each other [12].

Hypervisors have been traditionally split in Two types:

- Type one, or bare metal hypervisors that run guest virtual machines directly on a system's hardware, essentially behaving as an operating system.
- Type two, or hosted hypervisors behave more like traditional applications that can start or stop like normal programs. In modern systems, this split is less prevalent, particularly with systems like KVM.

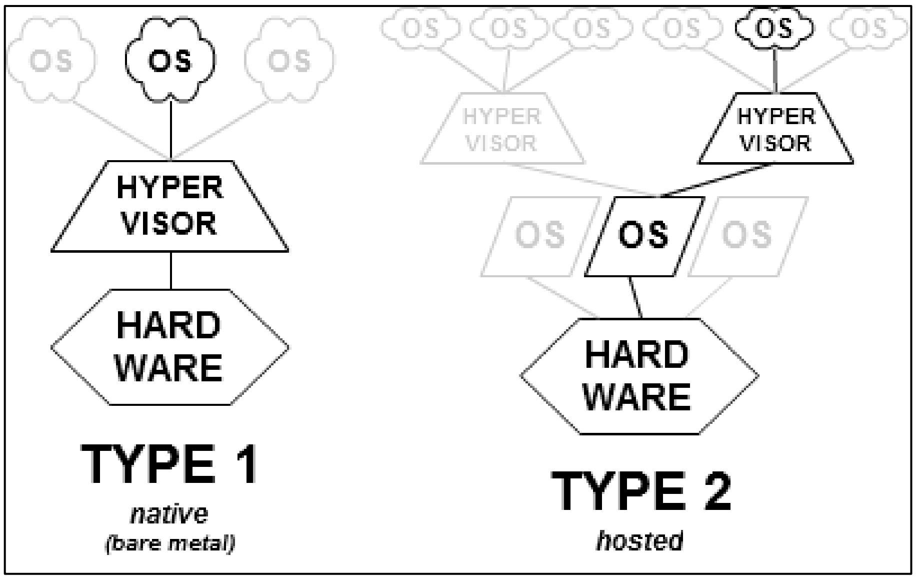


Figure 1.3 *Types of Hypervisors*

The main hypervisors on the market are those of VMware (ESX), Microsoft (Hyper-V), RedHat (KVM), Oracle (Oracle VM) and the open-source hypervisor Xen.

The main Type 2 solutions are VirtualBox (Oracle/Sun), VMware Workstation (on Windows and Linux) and Fusion (on Mac), and the open-source hypervisor QEMU [10].

1.4 Evolution of IT Infrastructure

IT Infrastructure went through different phases since the initial stages and each vendor adapted these phases to their own plans and terms. However, there is an agreement that the path was not exactly straight forward and different steps have been taken. passing by the 3-tiers “Siloed” architecture, managing it was not quite the optimal task, as it had the compute server attached to a separate independent storage and network devices, which lead to higher complexity level and different user interfaces to manage the many hardware components.

This motivated IT researchers to come up with solutions that eliminated as many hardware components, user management interfaces as possible from the 3-tiers “siloed” architecture, without sacrificing any of the benefits that the architecture environment had provided.

Eventually leading to the appearance of Converged Infrastructure, then Hyper-Convergence and the concept of a software defined data center.

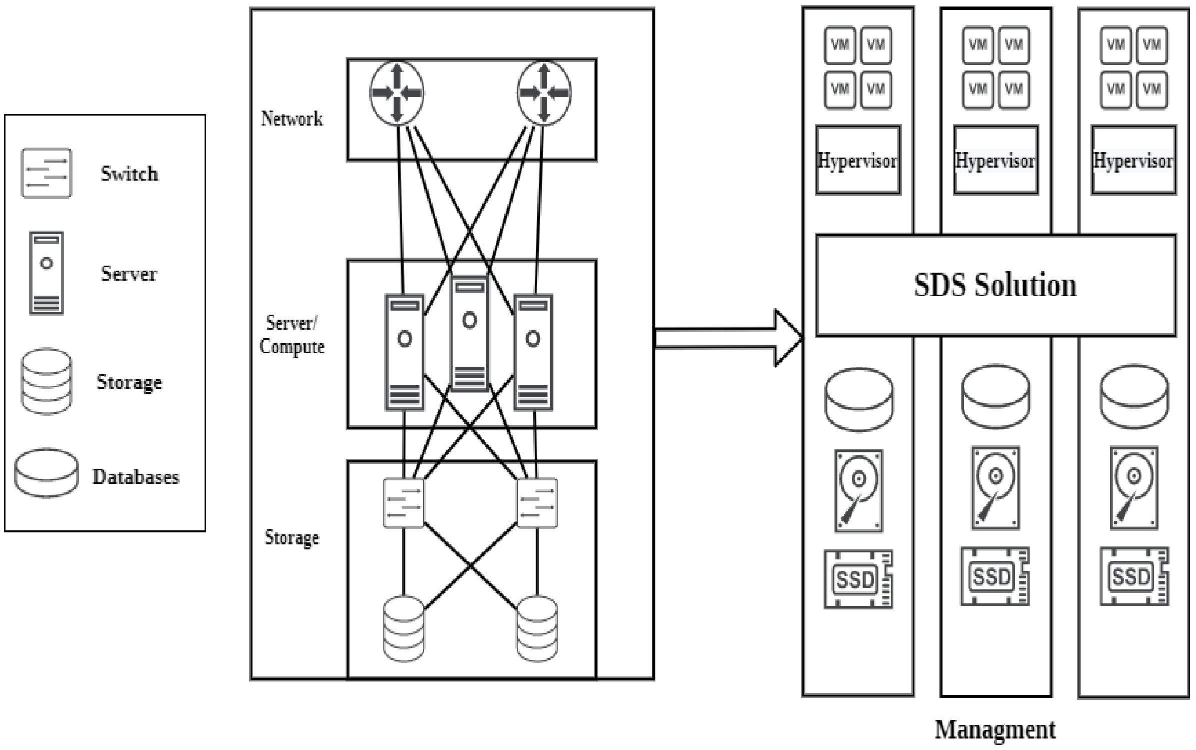


Figure 1.4 *Traditional infrastructure shifting to Hyper Converged Architecture*

1.4.1 Legacy infrastructure

The legacy Infrastructure is a preliminary stage of convergence in computing. Legacy system is an old method, technology, or computer system. Datacenters as all the digital solutions begun with a traditional infrastructure that were hardware-centric, in other words the compute power, storage arrays, and network components of the stack are physically detached and managed separately. However, the growing of digital businesses introduced different problems from a technological and interoperability point of view, such as the slow hard disks, unoptimized network devices, data loss, no deduplication, and separate management platforms, on top of these dilemmas there was a vendor compatibility problem, some of the hardware was not compatible with others which forced companies to stick to limited hardware vendors [8].

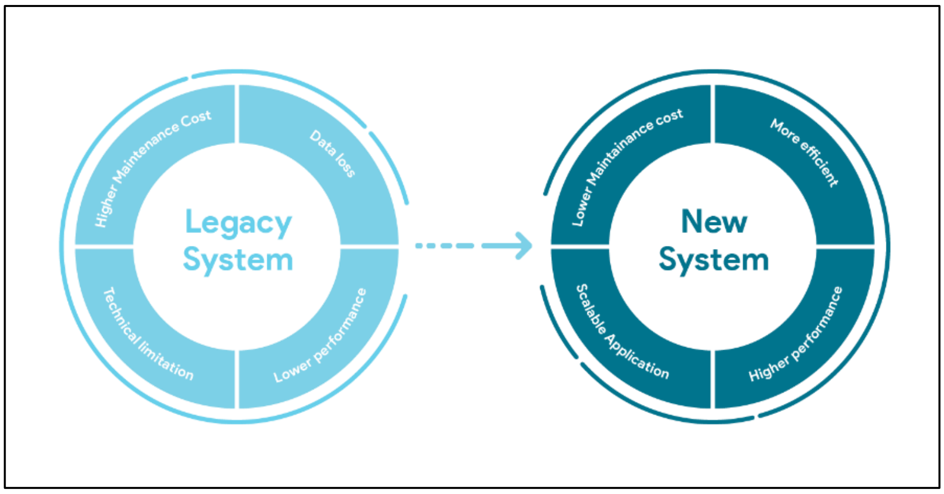


Figure 1.5 *Legacy system Limitation vs newer systems [13]*

1.4.2 Convergence

Convergence in infrastructure, also known as converged architecture, is data center management approach that packages compute, networking, servers, storage, and virtualization tools on a prequalified turnkey appliance. Converged systems also include a toolkit of management software.

Converged infrastructure is gaining momentum as IT companies shift away from owning and managing hardware to a flexible self-service model in which resources are used on demand. Rather than multiple IT assets existing in independent silos, converged infrastructure bundles hardware components with management software to orchestrate and provision the resources as a single integrated system.

The motivation of a converged infrastructure is the reduction of complexity of data center management. The main design factor is to remove hardware incompatibility issues. Ease of deployment of Converged Infrastructure is certainly a big appeal point [8].

1.4.3 Hyper convergence

One of the trending topics in the IT industry right now, is Hyper Convergence. In this model, infrastructure is strongly software-centric which means instead of delivering server functions via hardware, HCI does it through software, as a consequence it tightly integrates compute, storage, networking and virtualization resources in a single system that usually consists of x86 hardware, as opposed to the traditional infrastructure, where most of these resources were separate physical elements.

This kind of infrastructure does not only focus on centralizing and compacting the local datacenter, but also the integration and consolidation of management and resources from remote datacenters, extending the management and administration beyond the boundaries of the local facilities. Most hyper-converged systems require a minimum of three hardware nodes for high availability and can be expanded through the addition of nodes to the base unit. A grouping of nodes is known as a cluster [8].

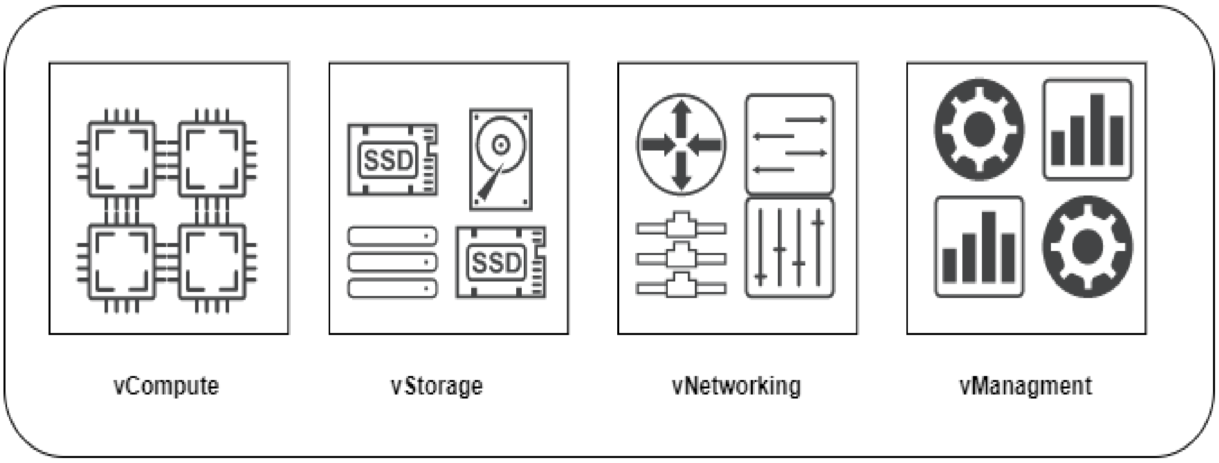


Figure 1.6 *Hyper Convergence*

Basically, hyper-convergence reduces the complexities and incompatibility issues associated with traditional datacenters. One of the most important features of HCI is the single point of management that allows administrators to use a console that unifies setup, management, configure, and monitor all resources and how easy it is to scale up once you reached the capacity of your infrastructure by just adding more nodes [14].

1.5 Software Defined Storage

Software defined storage is a term for computer program that manages storage and all its functionalities.

1.5.1 Introduction

In the most traditional IT architectures, computer programs or applications transform raw data into business value by processing data in various ways. However, the data does not normally occupy the application it is stored elsewhere, in a storage device. But imagine placing software in the data path between apps and storage devices, the applications can see no change, on the hand of the storage perspective everything has changed and this is the basis of SDS [15].

The SDS is a type of storage system to cope with the phenomenon of exponential growth in data volume, particularly related to the massive adoption of the Internet and mobile technologies. Something that traditional storage solutions do not allow. Their design is rigid, and does not offer the scalability and flexibility needed to cope with the explosion of Big Data. The explosion in the volume of unstructured data is indeed contributing to the rise of SDS, the only type of system that allows the scale of a storage architecture to change as needed [16].

Unlike SAN (Storage Area Network) and NAS (Network Area Storage) systems, which inseparably couple hardware and software, Software Defined Storage products allow users to update their software independently of the hardware. Common features of these devices include the ability to aggregate storage resources. The scale of a system can be scaled on a cluster of servers. The shared storage pool is managed through a single administration interface. Finally, rules are defined to control storage functionality [17].

1.5.2 Software Defined Storage motivations

People started to figure out how to scale data up, in other words they took the computer in between the client and the data to replace it with a giant spendy computer, this is called scaling up. However, they realized that this approach is broken. As a consequence, they went instead to scaling out, so alternatively to what they have today and make it bigger, they are taking what they have today and making it broader, which means you have more of it and you are parallelizing work, which is a whole new approach [18].

As people begin to figure this out, they started to build appliances that do this so all these computers and all these disks in this scale out storage architecture are put in a one box and this is what we call storage appliance [18].

1.5.3 Storage solutions

Before the software defined storage that tend to virtualize the storage by putting a software layer between the applications and storage devices, it existed before other storage solutions that are still efficient and used today in businesses infrastructure to store their data electronically and making them accessible easily and reliably [17].

1.5.3.1 Direct Attached Storage

Direct-attached storage (DAS) is a digital storage array directly attached to the server and been accessed directly through host bus adapter and not accessible to other servers. The hard disk or SSD is usually the form of DAS. In the companies and enterprises, the individual disk drives of a server are referred to as direct attachment storage, as are groups of drives that are external to the server but directly connected by a small computer system interface (SCSI), serial advanced technology attachment (SATA), serial attachment SCSI (SAS), or iSCSI [19].

Despite that DAS is less expensive than NAS and SAN, it does not have the performance to stand tall above SAN and DAS, moreover it doesn't offer advanced features like remote replication and snapshots [19].

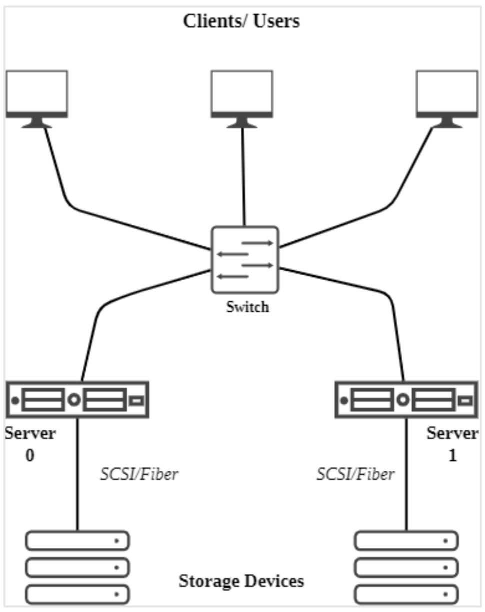


Figure 1.7 *Direct Attached Storage (DAS)*

1.5.3.2 Network Area Storage

Network-attached storage (NAS) is a dedicated file storage connected to a local area network, usually through an ethernet connection that enables multiple users and heterogeneous client devices to retrieve data from centralized disk capacity, NAS is easy to manage, reliable and provides a consolidated storage space however the average data transfer speed is not sufficient with the growth of the data and the human errors are not tolerated which may cause you to lose your NAS [20].

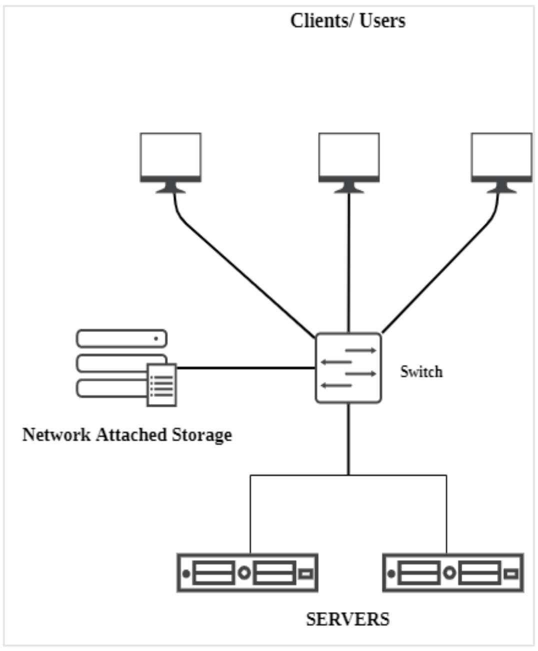


Figure 1.8 *Network Attached Storage (NAS)*

1.5.3.3 Storage Area Network

Storage area network (SAN) is a block-based storage and a dedicated high-speed network or sub-network that interconnects and presents shared pools of storage devices to multiple data path, in other words SAN breaks up the data into blocks and then stores those blocks as a separate pieces, each with a unique identifier then placing them wherever it is most efficient, whenever a user query a data it will be re-assembled through the network and retrieved quickly. SANs Allow high fault tolerance, good RAID storage configuration and high-speed data transfer rate all of these will offer scalability [21].

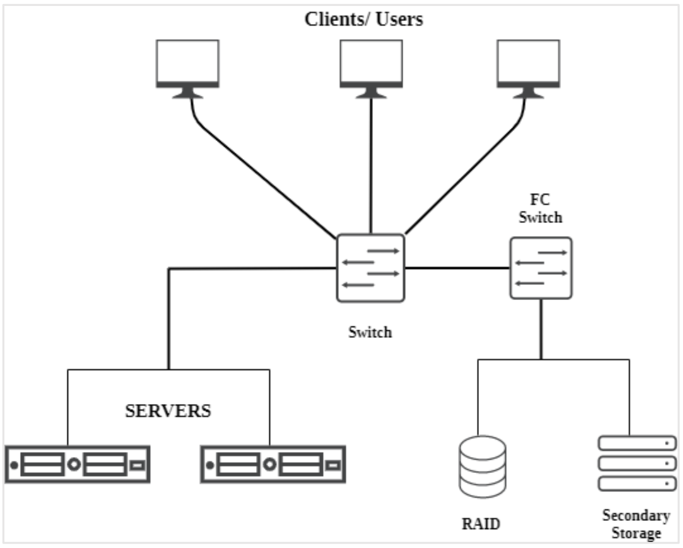


Figure 1.9 *Storage Area Network (SAN)*

Usual infrastructure goes to one server for each application with different storage compute needs, and they expand as they go. However, this is only valuable on the short-term but over time the growing complexity will lead to a cluttered environment with more silos, less visibility and less flexibility, but with SDS abstraction of the of application layer from storage resources it made the architecture more flexible and the data more portable fixing the common issues.

1.5.4 Definition of SDS

Software-defined storage is an approach to data management in which data storage resources are abstracted from the underlying physical storage hardware. Therefore, a solution with a service management interface, including automation, standard interfaces, virtualized data path, scalability and transparency. The resource flexibility is paired with programmability to enable storage that rapidly and automatically adapts to new demands. In the SDS model, the disks, enclosures, and networking components are all interchangeable, but the software intelligence managing the hardware does not need to be replaced [22].

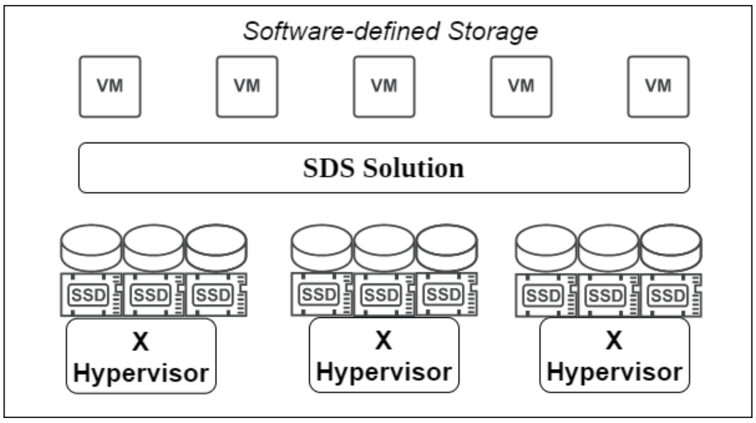


Figure 1.10 *Software-defined Storage an inside overview*

1.5.5 Software Defined Storage Benefits

Software-defined storage has rapidly become a key technology for enterprises looking to modernize or create their data centers, as more IT enterprises moves to the on premises infrastructure, optimizing the management of their data centers becomes critical to the success of the digital business, despite the efficiency of the existing storage solutions the need for a better solution that coop with the evolving technologies is becoming a must for businesses.

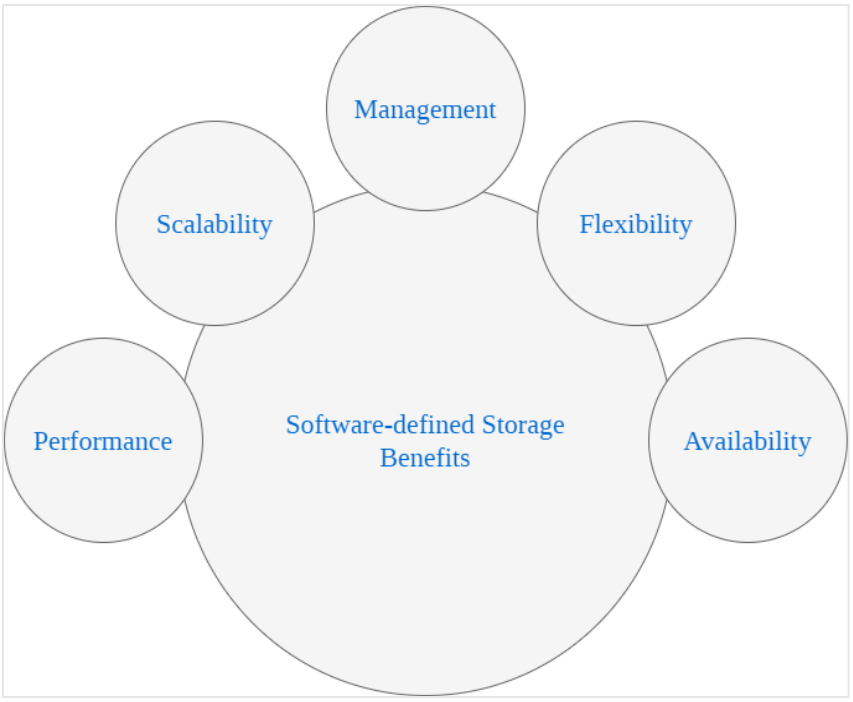


Figure 1.11 *Software-defined Storage benefits*

Management: management of storage use while knowing the available resources is simplified due automation and transparency, in addition You can execute every enterprise level function (snapshots, full clone, or create DR copy) supported by your SDS node [15]

Scalability: As your application data and I/O performance demands continue to grow, your ability to scale on the fly will be crucial to stay ahead. This can only be achieved if you have an adaptive infrastructure in place, SDS has the ability to scale out the storage infrastructure without impeding performance [15].

Flexibility: Software defined storage combines storage virtualization with advanced storage management capabilities, when you scale out your business storage arrays it doesn’t matter which vendor because SDS provide this leverage of flexibility [15].

Performance: refers to the amount of work performed by SDS components in relation to the time and resources used, SDS auto-tiering provides massive benefits both from performance and costs perspectives. When it comes to the average hot-data changes per day, studies have shown it is less than 5% of your total used capacity. This means that you really don’t need flash for all of your data [15].

High Availability: Availability is usually expressed as a percentage of the time available in a given year. Unavailability of services can generally occur as a result of system failures, SDS provide the availability by replications and backups across the pool [15].

Note: Having 2 active software-defined nodes will reduce hardware-related downtime by 100% and will provide a fully-automated failover infrastructure for your hypervisors, databases, and applications [23]

1.5.6 SDS existing solution

There are multiple existing solutions some are paid others are opensource such as Ceph storage, and GlusterFS.

1.5.6.1 GlusterFS

GlusterFS is free open source software and scale-out storage network file system suitable for data-intensive tasks such as cloud storage and media streaming solution that provides flexible and agile unstructured data storage and can utilize common off-the-shelf hardware [24].

GlusterFS provides new opportunities to unify data storage, increase performance, and make availability and manageability more improved in order to meet a broader set of storage challenges and needs, it can be installed and managed on-premise, or in a public cloud [25].

GlusterFS aggregates disk storage resources from multiple servers through network interconnects into one large single global parallel network file system. with stackable user space design, it delivers exceptional performance for diverse workloads and is a key development block of GlusterFS [24]:

- Accessible using standard protocols like NFS and SMB.
- Provides replication, geo-replication, snapshots and Bit rot detection.
- Allows optimization for different workloads.
- Scales to multiple petabytes.

1.5.6.2 Ceph Storage

Ceph is an open source platform for distributed storage, it is part of the Software-defined Storage (SDS) family of solutions, its use cases vary from cloud infrastructure and hyperconverged infrastructure to big data analytics and rich media. This SDS approach separates the physical storage hardware from the intelligence of data storage management. This has several advantages [26].

- As a distributed platform it provides very large scalability, extending storage to several petabytes; while offering strong resilience, as data is replicated to different locations in a cluster. In the event of a disk failure, the platform "rebuilds" itself.
- Administration is also simplified through automated policy-based management.
- It provides interfaces for multiple storage types within a single cluster.
- Because of its RADOS (Reliable Autonomic Distributed Object Store) abstraction layer, Ceph allows storage in block, object or Posix-compatible file system mode, the standard that defines common interfaces for Unix-like systems.
- Ceph provides scaling and fault management capabilities, which makes it ideal for microservice and container-based workloads such as cloud, OpenStack or others mainly for how effective Ceph addresses large data volume storage needs [7] [27].

Table 1-1 *Ceph Storage functions comparison to GlusterFS*

	Ceph Storage	**GlusterFS**
Pros	• Object storage strengths. • Better performance on simpler hardware. • Easy integration into all systems, no matter the operating system being used. • Block device for Linux. • RADOS compatibility. • Easier possibilities to create Client-specific modifications. • Block device for Linux. • CephFS file system for Linux.	• No central metadata server necessary. • Quicker storage algorithm. • File system strengths. • Lower complexity. • Better suitability for saving larger files (starting at around 4 MB per file). • Better suitability for data with sequential access. • Easy integration into Linux systems. • POSIX-compatibility.
Cons	• Weaker file system functions • Higher integration effort needed due to completely new storage structures	• Integration into Windows systems can only be done indirectly

1.5.7 Ceph Architecture

The Ceph architecture is really an impressive technology approach we will be representing it with this diagram and explain each component

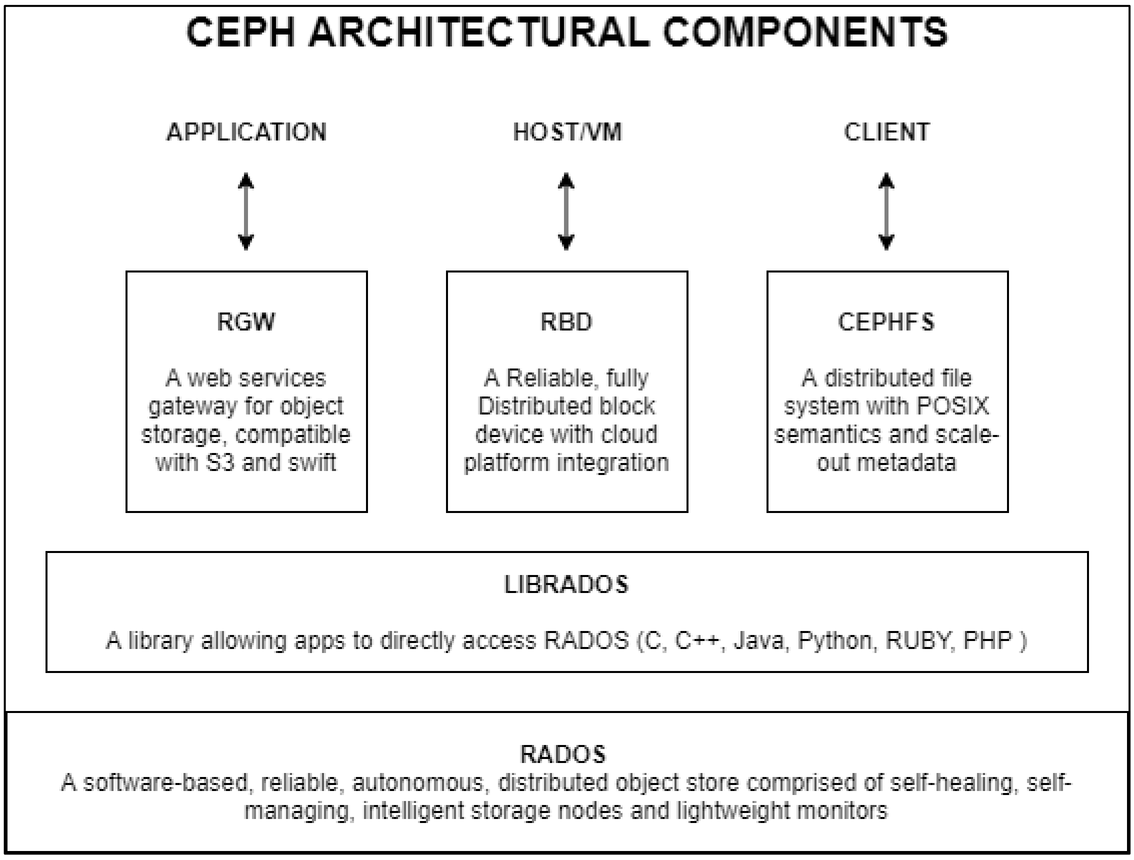

CEPH ARCHITECTURAL COMPONENTS
APPLICATION
HOST/VM
CLIENT
RGW
A web services gateway for object storage, compatible with S3 and swift
RBD
A Reliable, fully Distributed block device with cloud platform integration
CEPHFS
A distributed file system with POSIX semantics and scale-out metadata
LIBRADOS
A library allowing apps to directly access RADOS (C, C++, Java, Python, RUBY, PHP)
RADOS
A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

Figure 1.12 *Ceph Architectural Components Overview*

RADOS: Reliable Autonomic Distributed Object Store (RADOS) is an open source object storage service that is an integral part of the Ceph distributed storage system, and this is basically how it works, when having multiple disks either SSDs, HDDs, NVMEs or even a raid groups and on top of each disk there is a file system i.e. BTRFS, XFS or EXT4 on top of those file system there are OSDs “Object Storage Daemon” this is the software layer and what it does is it takes each of those disks and makes it part of the Ceph storage cluster, so it is a simple piece of software when you configure it and point it to a path and it turns that path into a storage location for the storage cluster, also when you interact with the storage cluster you interact with the entire cluster as a logical unit not as individual series of posts [18].

LIBRADOS: it’s a native protocol its job is to give an application when linked to librados the intelligence to speak to the storage cluster, so LIBRADOS provides direct access to RADOS for applications [18].

RADOSGW: is an HTTP REST gateway for the RADOS object store, a part of the Ceph distributed storage system [18].

RBD: Ceph block devices are thin-provisioned, resizable and store data striped over multiple OSDs in a Ceph cluster. Ceph block devices leverage RADOS capabilities such as snapshotting, replication and consistency. Ceph's RADOS Block Devices (RBD) interact with OSDs using kernel modules or the librbd library [18].

CephFS: is a POSIX-compliant file system built on top of Ceph's distributed object store, RADOS. CephFS endeavors to provide a state-of-the-art, multi-use, highly available, and performant file store for a variety of applications, including traditional use-cases like shared home directories, HPC scratch space, and distributed workflow shared storage [7].

1.6 The importance of software defined solutions

As with all previous technological revolutions in the infrastructural & enterprise world, the Software-Defined approaches will force organizations to adapt. Software-defined environments will require rethinking many IT processes including automation, metering, and billing and executing service delivery, service activation, and service assurance [28].

While a widespread transition to the SDDC could take years, this process will certainly become a necessity for the industry very, very soon. Those who will refuse to change their approach to IT, will eventually become obsolete, just like the critics who opposed virtualization as a whole in the past. [28]

1.7 Conclusion

In this chapter, we talked about the evolution of the IT infrastructure, we have discussed the core components of SDC and SDS solutions in order to apply its concepts in our project.

In the next chapter we are going to view and discuss the network side of the software defined approach that will be a crucial part in the implementation.

2 Chapter 2 Software-defined Network

2.1 Introduction

Nowadays, the internet has known a huge success. It has become a universal and indispensable tool for businesses and the daily lives of billions of people. With the explosion in the number of mobile devices, the massive use of social networks and the emergence of new IT trends such as Cloud Computing, virtualization, BIG-DATA and the Internet of Things, that generate large volumes of data, the Internet is facing the challenge of handling these increasingly gigantic volumes of traffic [29].

However, the Internet has become a critical infrastructure because of the lack of changes in the core network layer and the rigidity of the deployed equipment, which makes the establishment and deployment of new network services difficult and costly, one of the remaining problems that everyone wants to solve is the establishment of strong, safe and low-cost solutions that can be relied on and are easy to process while still being powerful enough to solve any future problems that may arise [29].

This gave researchers a reason to move to a centralized controlled architecture, dynamic, easy to manage and highly scalable, hence the Software-Defined Network (SDN), has revolutionized the traditional approach to networks. In this chapter, we present the SDN technology while introducing the need to have a programmable network [30].

2.2 The motivation to SDN

Over the past decade, we have witnessed many changes in these traditional networks, which have led the technologists to developing these devices, and networks to try and evolve this model. The main drivers for change are the size of the networks which have grown massively due to the number of devices which need connectivity from virtual machines sprawl to tablets and smart phones.

The rate of change in configuration of the devices is also Affected by the explosive rollout of applications being developed using new methods and in order to make changes to these devices a network admin has to connect to each device and make the configuration change. To add to the complexity each device vendor uses their own configuration language, user interface and syntax making it a highly skilled time-consuming task [31].

2.3 Traditional network

The structure of the Internet and computer networks generally consists of different network devices such as routers, switches and different types of middleboxes that are vertically integrated and designed by chips with high throughput and specific function [32]. traditional network equipment currently in use consists of two fundamentals parts: control plan and data plan (the brain and body). The function of the first part is to make decisions and initiate processes, such as routing or redirection of traffic, whereas the second part deals with the implementation of all the decisions made by the first part, i.e. when a packet arrives on a port of a switch or router, they apply the routing or switching rules that are written into its operating system [33].

It is difficult to manage networks and to apply new policies effectively, because each device operates with a protocol of a given level and must be appropriately configured to communicate with each other. So, it is not easy to add a rule because there is no standardized way to do so, but also because one device may conflict with one of the many other devices on the network. In addition, rules were originally intended to be static and therefore the network is static and poorly adapted to the needs and current technologies.

Network administrators when managing and configuring one of these network devices, a set of specific and predefined command lines based on an integrated operating system is used. As a result, it can be argued that managing a large number of network devices is a great challenge that is prone to many errors.

Networking has not experienced the same revolution of the Personal Computer such as commoditizing hardware, offering a choice between multiple operating systems and creating a market of competing applications. All manufacturers of network machines have developed very complicated systems. Thus, traditional networks suffer from significant gaps in research and innovation, reliability, scalability, flexibility and management. As a consequence, for businesses it's an expensive market [33].

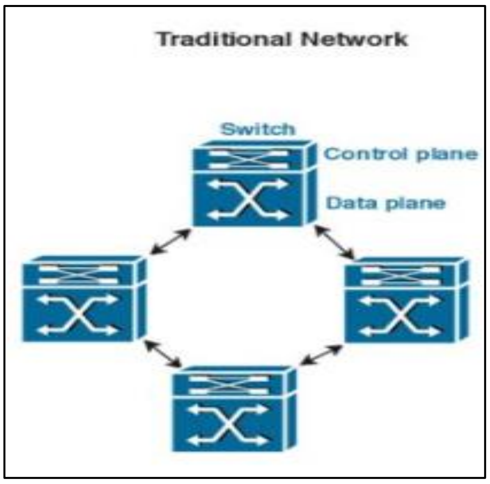


Figure 2.1 *Traditional Network [34]*

Since the birth of the Internet, networks are expanding and new technologies such as the cloud, social networks and virtualization have emerged, the need for networks with higher bandwidth, greater accessibility and higher dynamic management has become a critical issue [35].

To solve the problems and limitations of traditional networks, a structure has been proposed, known as SDN, where the control of the network is separated from the transmission mechanism and can be programmed and controlled directly.

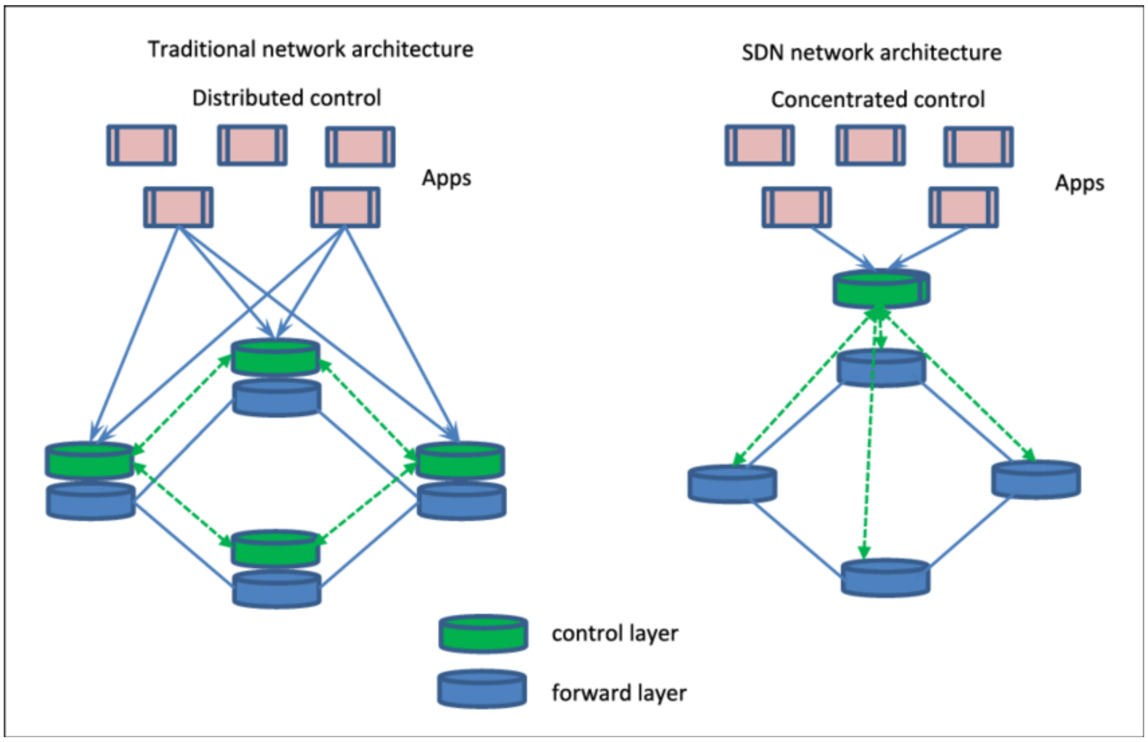


Figure 2.2 *Traditional network architecture and SDN network architecture. [36]*

2.4 Definition of SDN

Software-defined network (SDN) is a technology that makes the network programmable. SDN has a network architecture where the control plane is totally decoupled from the data plane. The control plane does the management of network devices, while the data plane is the Hardware layer responsible for transferring network packets according to the policies defined in the control plane. This decoupling transform network switches/routers into simple control devices, while the logic command is implemented in the controller which functions as a centralized network operating system. Thus, the SDN is composed of two entities: the controller and Forwarding Devices (switch or router).

The SDN as a set of solutions/architectures to remove boundaries existing between the worlds of applications and the network. This makes it more globally recognized today as an architecture to open the network to applications. This integrates the two parts:

- Allowing APIs to program the network to accelerate deployment.
- Allowing the network to better identify the applications transported in order to better manage them. (quality of service, safety, traffic engineering...)

2.5 Software Defined Network Benefits

The goal of these innovations is to simplify network administration and, following the example of what virtualization has achieved in the world of servers, to make the consumption of network resources by applications more flexible.

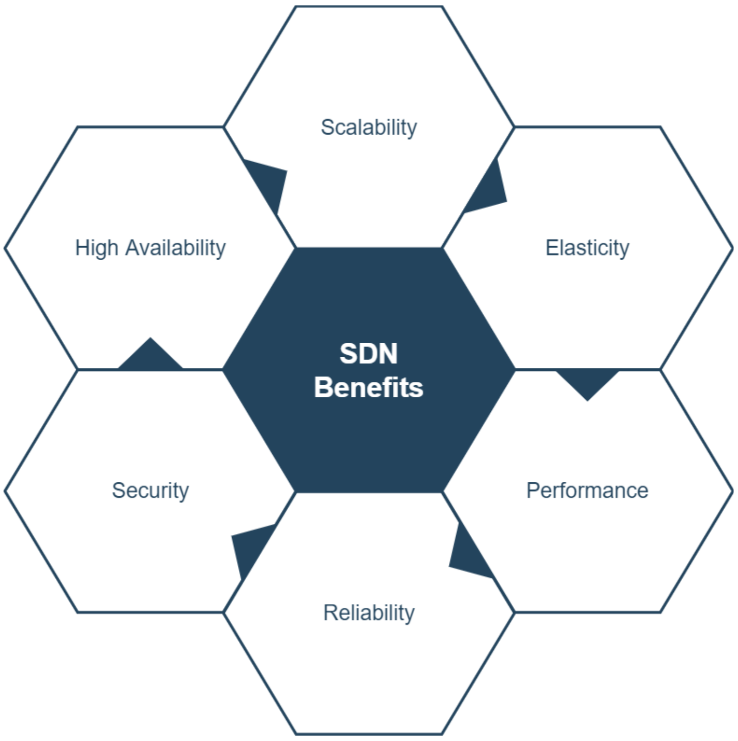


Figure 2.3 *Software-Defined Network Benefits*

Scalability: This defines the ability of the SDN, more specifically in the control plan, to Manage and process an increasing workload. Scalability aims to expand the capacity of the SDN by implementing mechanisms such as devolving [37], clustering [38] and "high processing" [39] to cope with the increasing load.

High Availability: HA is an important aspect of today's services that should be available whenever a customer requests a particular service or resource. Availability is usually expressed as a percentage of the time available in a given year. Unavailability of services can generally occur as a result of network outages or system failures [40], [41]. Network providers typically deploy backup to provide HD by implementing redundant server hardware, redundant server components, and server and network OS.

Security: The security of the SDN consists in protecting the information against theft or damage to hardware and software as well as against interruption of services [42], [43]. Securing the

SDN encompasses the physical security of the hardware, as well as the prevention of logical threats that may come from the network or data. The vulnerabilities of the SDN are the gateway to intentional or accidental security attacks.

Reliability: The SDN is considered reliable when reporting data failures by real time. In such a system, there should be a specified minimum reliability for the routing of critical data. In current implementations, the SDN controllers [44] must be capable of meeting the real-time reliability and punctuality requirements of routing.

Performance: Performance refers to the amount of work performed by SDN components in relation to the time and resources (e.g., CPU and RAM) used [45]. It There are many different ways to measure the performance of a network [46], [47], because every network is different in nature and design. With regard to the SDN, the measures important are bandwidth, throughput, latency and jitter.

Elasticity: Elasticity in the SDN is the ability to provide and maintain a level of service acceptable even in the event of a service, network or node failure. When an SDN is faulty, the network must provide continuous operational service with the same performance. In order to increase the elasticity of the SDN, challenges and potential risks need to be identified and to protect services [48]

2.6 SDN Architecture

The structure of the SDN consists of three main parts. The lowest level, includes the data plane, at the highest level there is the application plane, and the control plane is between them. Communication between the controllers/control plane and the data plane is managed via the Southbound Interface (SBI) at the SDN switch, and communication between the applications (network API's) and the controllers is handled by the Northbound Interface (NBI) in the control plane. [49].

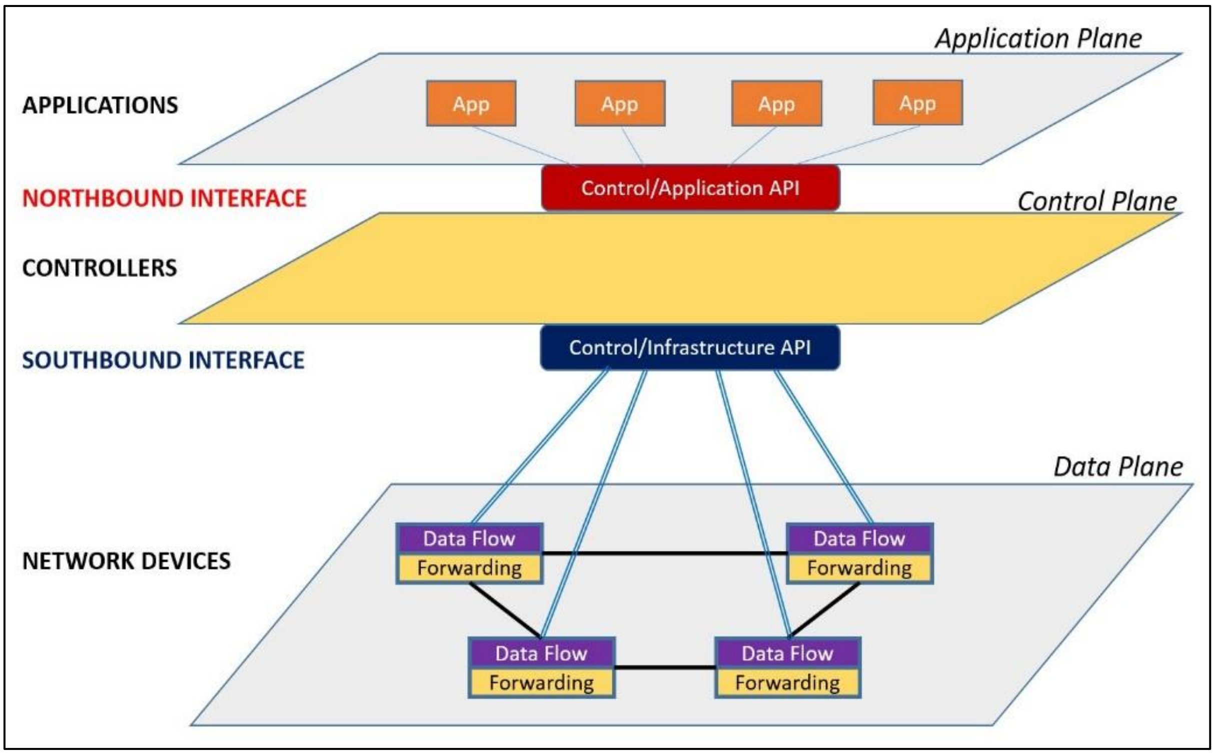


Figure 2.4 *Software Defined Network Architecture [50]*

Using the repartition between control plane and data plane, applications follow their own particular purpose such as security method, QoS, traffic, engineering and network measurement and monitoring solutions. In addition, the controller helps to achieve their goal by controlling the SDN switches via flow tables. In others, the network adapts to user needs, and using the controller and the APIs, network managers can easily control the network automatically by adding new functions to the control plane, without making any changes to the data plane [50].

2.6.1 Data plane

An SDN infrastructure, like a traditional network, is composed of a set of network equipment, the main difference lies in the fact that these traditional physical devices are now simple transmission elements without integrated control or software to make autonomous decisions. Network intelligence is removed from the devices in the data plane to a logically centralized control system, i.e. the Network Operating System (NOS) and applications, as shown in figure 2.5. Most importantly, these new networks are built on open and standard interfaces (e.g. OpenFlow), a crucial approach to ensure compatibility of configuration and communication and interoperability between different data devices and control plans. In other words, these open interfaces allow controller entities to dynamically program heterogeneous transmission devices, which is difficult in traditional networks because of the wide variety of proprietary interfaces and the closed distributed nature of the control plane interface.

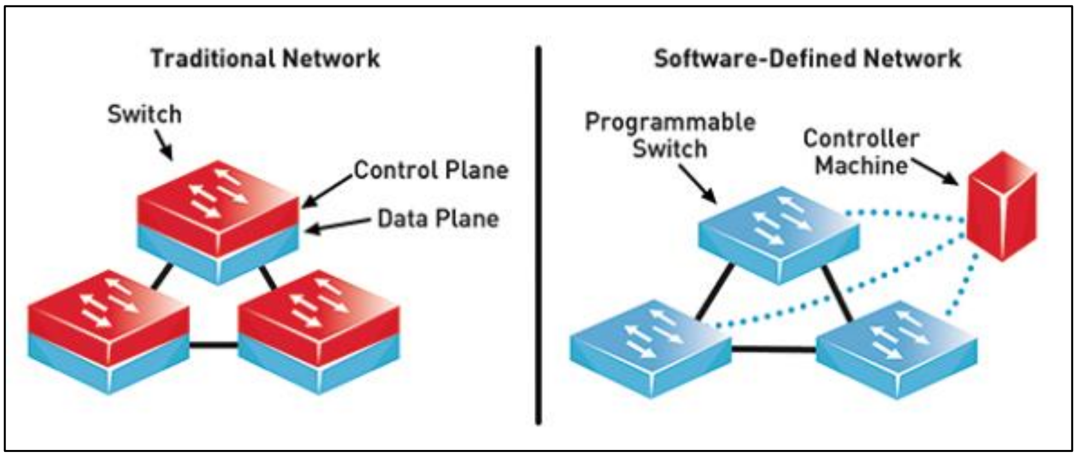


***Figure 2.5** Traditional Network versus SDN [51]*

In an SDN/OpenFlow architecture, there are two main elements, the controllers and the transmission devices. A data plane device is a hardware or software element specialized in packet transmission, while a controller is the network brain running on a conventional hardware platform. An OpenFlow-compatible transmission device is based on a flow table pipeline where each flow table input has three parts: (1) a matching rule; (2) actions to be performed on the matching packets and (3) counters that keep statistics of the matching packets. This high-level, simplified model derived from OpenFlow is currently the most widely used design of SDN data plan devices.

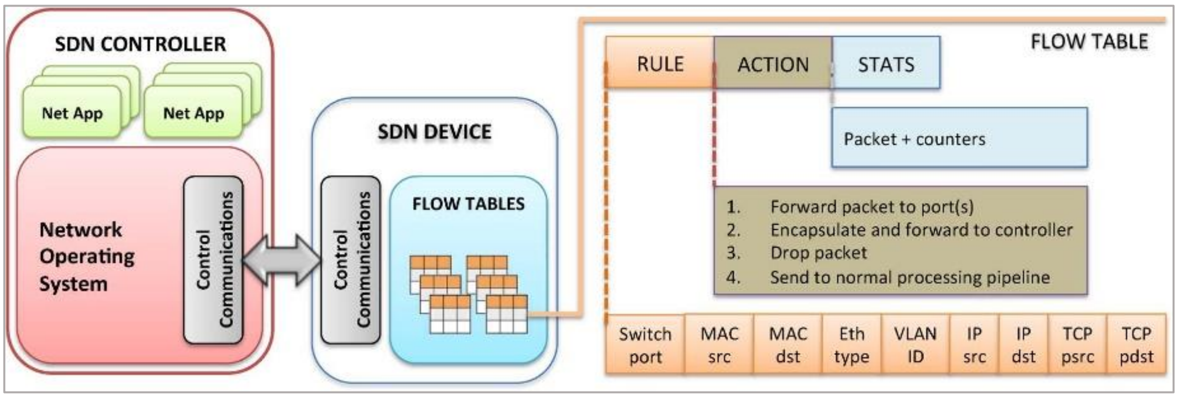


***Figure 2.6** OpenFlow Activated on SDN devices [52]*

2.6.2 Southbound Interface

The southbound interface of the SDN controller needs a way to communicate with network forwarding devices, some types of information that needs to be communicated includes packet handling, instructions alerts of packet arrivals on network, nodes notifications of status changes like links going up or down and providing statistics information like flow counters all this happens over the southbound interface, that’s why the southbound API are a crucial component and the most critical in the SDN system. The are many SDN protocols on the southbound interfaces such as OpenFlow, OpFlex, POF and OpenState. However, the most used protocol for packet handling instructions is OpenFlow.

2.6.2.1 OpenFlow Protocol

OpenFlow is a protocol that describes the interaction of one or more control servers with OpenFlow-enabled switches. OpenFlow is standardized by the Open Networking Foundation (ONF). An OpenFlow controller installs flow table entries in the switches, so that the switches can transfer traffic based on these entries. Thus, the OpenFlow switches depend on the configuration of the controllers. [53].

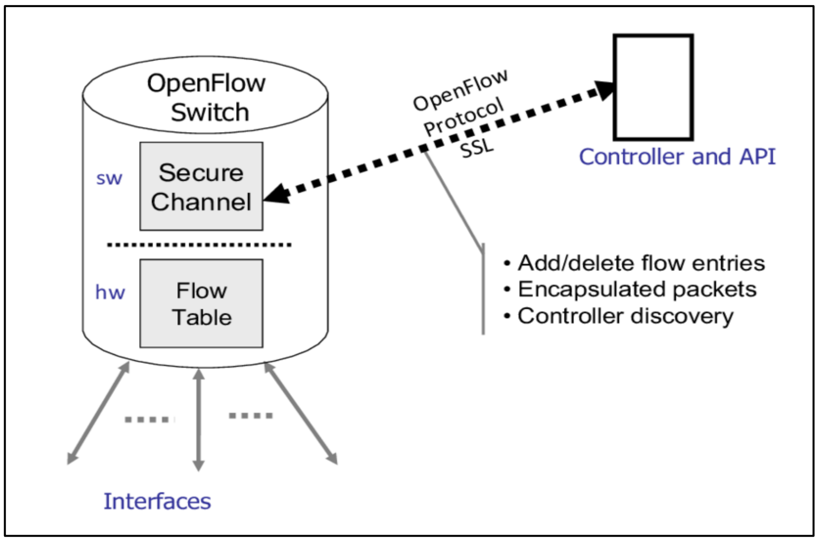


Figure 2.7 *OpenFlow Protocol [52]*

2.6.2.2 OVSDB Protocol

It’s true that OpenFlow is the most widely used Southbound protocol however it doesn’t work alone, complementary to OpenFlow is the OVSDB protocol which is a network management protocol used to manage network device configurations an OpenvSwitch and is also being adopted by some hardware-based switches, OF-Config is a management protocol developed by the ONF and is supposed to work with any OpenFlow compatible device, while the OVSDB protocol is specifically developed for OpenvSwitch. OVSDB works with the software and hardware implementation of OVS.

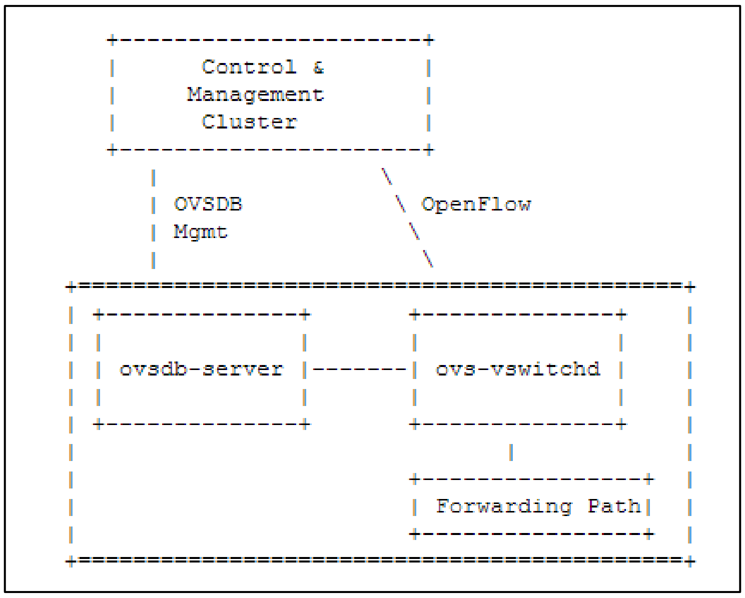


Figure 2.8 *OVSDB Architecture [54]*

OVSDB manages switching operations such as creating interfaces, defining QoS policies or shutting down a physical port. The figure 2.8 illustrates the architecture of the OVSDB. An OVS instance consists of a database server (ovsdb-server) and a vswitch daemon (ovs-vswitchd). The management and control cluster consists of the OVSDB managers and the OpenFlow controller, which can be located in the same or different devices. While the controller communicates with the switches via the OpenFlow channel, the OVSDB server talks to its manager via the OVSDB management protocol. The OVSDB switch daemon, located in a switch, monitors the database for additions, deletions, and changes to this information. Any changes in the database are applied to the switch. The OVSDB server stores switch information in a database. The configuration in OVSDB is permanent; the switch will not lose its configuration if the switch is restarted.

The OVSDB is formalized in RFC7047 [54]. Many OpenFlow controllers, such as OpenDayLight and Ryu have an API to communicate with OVSDB. This support allows the integration of switch management into the OpenFlow application.

The following diagram shows the relationship among tables in the database. Each node represents a table. Tables that are part of the ‘‘root set’’ are shown with double borders. Each edge leads from the table that contains it and points to the table that its value represents. Edges are labeled with their column names, followed by a constraint on the number of allowed values: ? for zero or one, * for zero or more, + for one or more. Thick lines represent strong references; thin lines represent weak references [54].

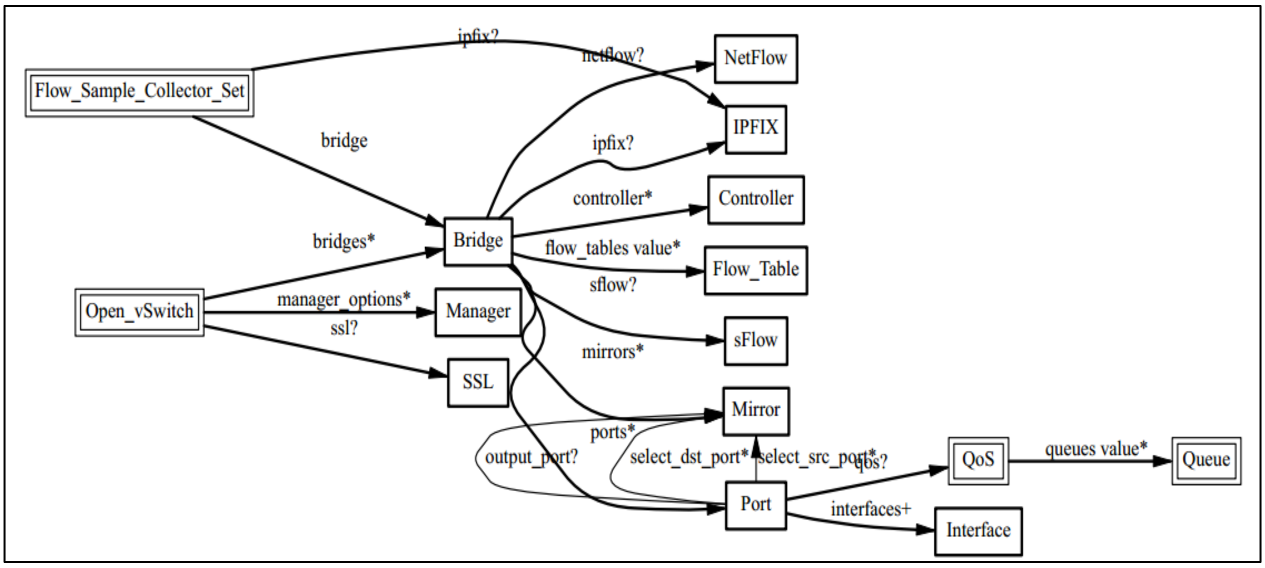


Figure 2.9 *OpenvSwitch Data base Schema [54]*

2.6.3 Control Plane

Traditional operating systems provide abstractions (e.g. high-level programming APIs) to access lower-level devices, manage concurrent access to underlying resources (e.g. hard disk, network adapter, CPU, memory) and provide protection mechanisms. These features and resources are key tools to increase productivity and make life easier for system and application developers. Their Widespread use has greatly contributed to the evolution of various ecosystems (e.g., the programming languages) and the development of a variety of applications. However, networks have been managed and configured using sets device-specific lower-level instructions and network operating systems mostly closed source such as Cisco IOS or Juniper JunOS. In addition, the idea of operating systems abstracting the specific characteristics of the devices and providing Transparently, common functionalities are still largely absent in the networks. For example, today's designers of routing protocols are faced with complex distributed algorithms to solve network problems. Network practitioners are therefore always solving the same problems over and over again.

The SDN is promised to facilitate network management and ease the burden of solving network problems through the logically centralized control offered by a network operating system. Second, the SDN controller acts as the "brain" of the network. It provides SDN users with a central view of the entire network and allows network administrators to tell switches and routers how the transfer plan should direct network traffic [53].

2.6.4 Northbound API

The Southbound API already has a widely accepted protocol (OpenFlow), but a common Northbound API remains an open issue. For the time being, it may still be a bit too early to define a standard Northbound API, as the use cases are still under development. Existing controllers such as Floodlight, Trema, NOX and OpenDayLight offer and define their own Northbound APIs. However, each one of them has its own specific definitions.

Programming languages can provide a wide range of abstractions and powerful mechanisms such as application composition, fault tolerance in the data plane, and a variety of building blocks to facilitate the development of software modules and applications.

It is unlikely that a single Northbound API will be a winner, as the requirements of different network applications are very different. APIs for security applications are likely to be different from those for routing applications. ONF's architectural work [55] includes the possibility of providing northbound APIs to allow dynamic and granular control of network resources from client applications, possibly across different business and organizational boundaries.

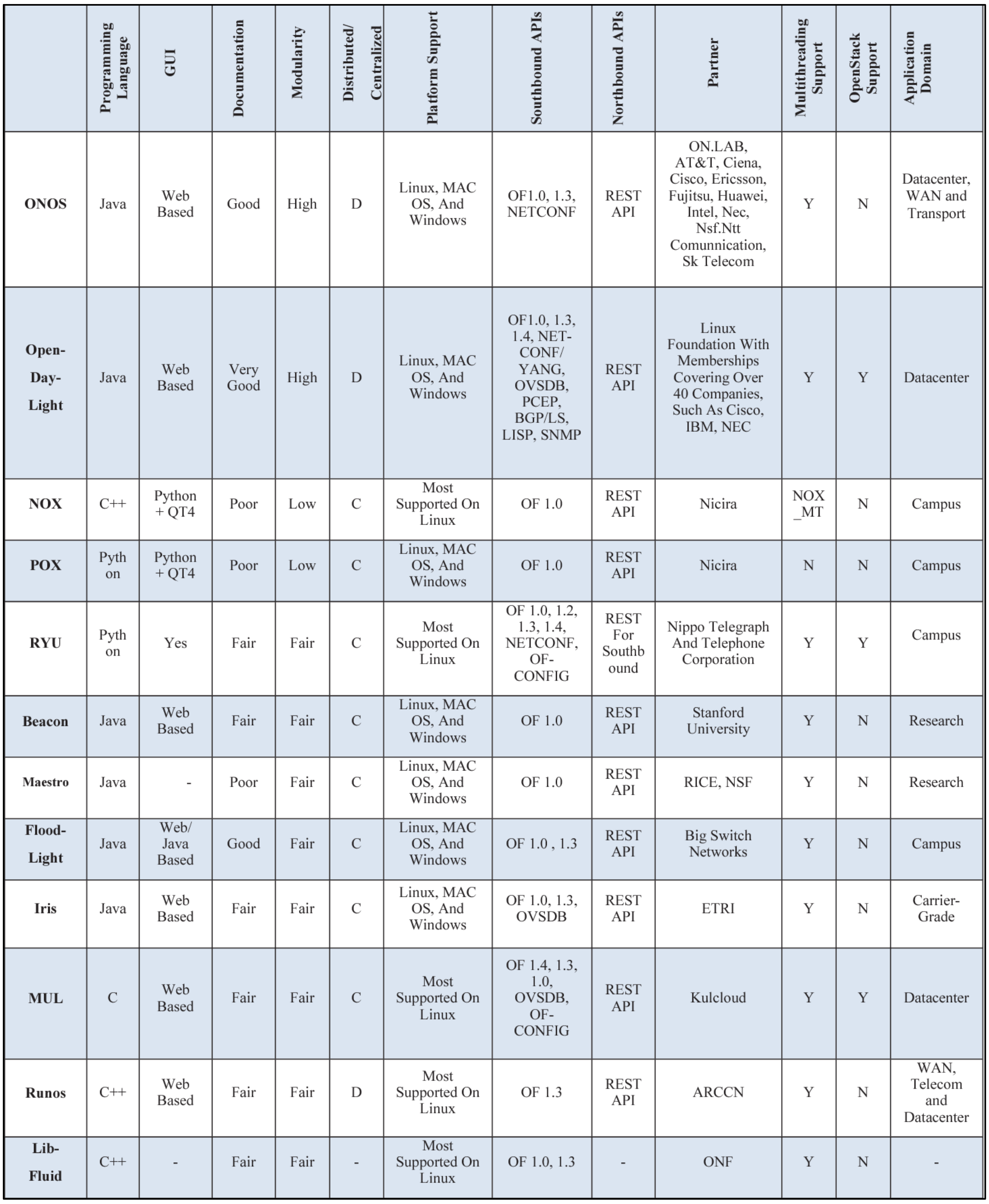
2.6.5 Application Plane

As shown in figure 2.4, the application layer is above the control layer. Through the control layer, SDN applications can easily access a global view of the network with instantaneous status via Northbound, for example, the ALTO Protocol (Application Layer Traffic Optimization) [56]. With this information, SDN applications can programmatically implement strategies to manipulate the underlying physical networks using a high-level language provided by the control layer.

2.7 Types of SDN controllers

- **Beacon**: Beacon is an SDN controller that was introduced in 2010. It has been used in several research projects. It is a Java-based controller. It can run on many platforms, including high-end multi-core Linux and Android phones.
- **OpenDayLight**: OpenDayLight is inspired by Beacon. It is a Java-based controller derived from Beacon. It supports OpenFlow and other Southbound APIs. The OpenDayLight is present in its own Java Virtual Machine (JVM).

- **Maestro**: Maestro is the first OpenFlow control system that exploits parallelism. In Maestro, programmers can modify the functionality of the data plane by writing simple single threaded programs. Maestro has its own set of concepts and techniques that support the OpenFlow protocol. It is a highly portable Java-based controller for different operating systems and architectures.
- **IRIS**: IRIS is an SDN controller platform that is created to control the OpenFlow base. It can manage a large network. IRIS supports architectures that are horizontally scalable. Thus, servers can be added dynamically to the controller cluster. This increases the performance factor of the flat control.
- **NOX**: NOX is the first SDN OpenFlow Controller platform for building network control applications. It was initially developed by Nicira Networks, alongside of OpenFlow. Later, NOX was given to the SDN community. Applications can be in Python or C++ and can be loaded dynamically.
- **POX**: Pox is similar to the NOX Controller. POX is an SDN controller that enables rapid network development and prototyping. It follows the OpenFlow protocol, and serves as a framework between OpenFlow switches.
- **Floodlight**: Floodlight is an open source SDN controller. It is an enterprise-class OpenFlow controller, based on Java. It works with both physical and virtual switches that use the OpenFlow protocol.
- **DISCO**: Disco is a distributed controller. It is mainly used for WAN networks and overlay networks. Each controller is responsible for one network domain. The regulators communicate with each other through an inter-controller channel. DISCO can adapt dynamically to different heterogeneous network topologies.
- **Ryu**: Ryu means "flow" in Japanese. Ryu is a component-based SDN controller, provides many extended software components and APIs to create and manage network applications. Ryu supports protocols such as OpenFlow, Netconf, OF-config, etc. Ryu is fully implemented in Python language. It is licensed under the Apache 2.0 license.



	Programming Language	GUI	Documentation	Modularity	Distributed/ Centralized	Platform Support	Southbound APIs	Northbound APIs	Partner	Multithreading Support	OpenStack Support	Application Domain
ONOS	Java	Web Based	Good	High	D	Linux, MAC OS, And Windows	OF1.0, 1.3, NETCONF	REST API	ON.LAB, AT&T, Ciena, Cisco, Ericsson, Fujitsu, Huawei, Intel, Nec, Nsf.Ntt Comunnication, Sk Telecom	Y	N	Datacenter, WAN and Transport
Open-Day-Light	Java	Web Based	Very Good	High	D	Linux, MAC OS, And Windows	OF1.0, 1.3, 1.4, NET-CONF/ YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP	REST API	Linux Foundation With Memberships Covering Over 40 Companies, Such As Cisco, IBM, NEC	Y	Y	Datacenter
NOX	C++	Python + QT4	Poor	Low	C	Most Supported On Linux	OF 1.0	REST API	Nicira	NOX _MT	N	Campus
POX	Pyth on	Python + QT4	Poor	Low	C	Linux, MAC OS, And Windows	OF 1.0	REST API	Nicira	N	N	Campus
RYU	Pyth on	Yes	Fair	Fair	C	Most Supported On Linux	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OF-CONFIG	REST For Southb ound	Nippo Telegraph And Telephone Corporation	Y	Y	Campus
Beacon	Java	Web Based	Fair	Fair	C	Linux, MAC OS, And Windows	OF 1.0	REST API	Stanford University	Y	N	Research
Maestro	Java	-	Poor	Fair	C	Linux, MAC OS, And Windows	OF 1.0	REST API	RICE, NSF	Y	N	Research
Flood-Light	Java	Web/ Java Based	Good	Fair	C	Linux, MAC OS, And Windows	OF 1.0 , 1.3	REST API	Big Switch Networks	Y	N	Campus
Iris	Java	Web Based	Fair	Fair	C	Linux, MAC OS, And Windows	OF 1.0, 1.3, OVSDB	REST API	ETRI	Y	N	Carrier-Grade
MUL	C	Web Based	Fair	Fair	C	Most Supported On Linux	OF 1.4, 1.3, 1.0, OVSDB, OF-CONFIG	REST API	Kulcloud	Y	Y	Datacenter
Runos	C++	Web Based	Fair	Fair	D	Most Supported On Linux	OF 1.3	REST API	ARCCN	Y	N	WAN, Telecom and Datacenter
Lib-Fluid	C++	-	Fair	Fair	-	Most Supported On Linux	OF 1.0, 1.3	-	ONF	Y	N	-

Figure 2.10 *SDN controllers' comparison [57]*

2.8 Conclusion

In this chapter, we have provided a theoretical background on Software Defined Networks (SDNs), presenting the architecture and advantages of the latter, we have also treated the essential components of this solution in order to apply its concepts to our context.

3 Chapter 3 Implementation and tests

In this chapter, we will present the development environment of our system, specifying the tools we've chosen to build and deploy it, then we will present an overview of the functionalities proposed in the previous chapters, as well as the main interfaces that composes it.

3.1 The working space

First of all, the table 3-1 demonstrate the specification of the hardware used in our project, we will start with a description of the environment and the tools chosen for the realization of our system.

Table 3-1 *Hardware capacity used to the appliance*

Hardware	Capacity
CPU	Intel I7-4800 MQ 4 Cores 8 threads
RAM	16 Giga Bytes DDR3 frequency 1600 MHz
Storage	1 TB of spinner Hard Drive of 7200 rpm

Resource virtualization allows us to benefit from value optimization, currently we use virtualization machines based on VMware Workstation Pro v15 that allow us to isolate and better use our hypervisors and VMs.

3.2 Conception

As can be seen from figure 3.1, the traditional architecture of our organism is composed of three layers which are networking, server computation and storage system. However, the cost efficiency is not optimal for the company use, and the growth of the data will overwork this traditional architecture.

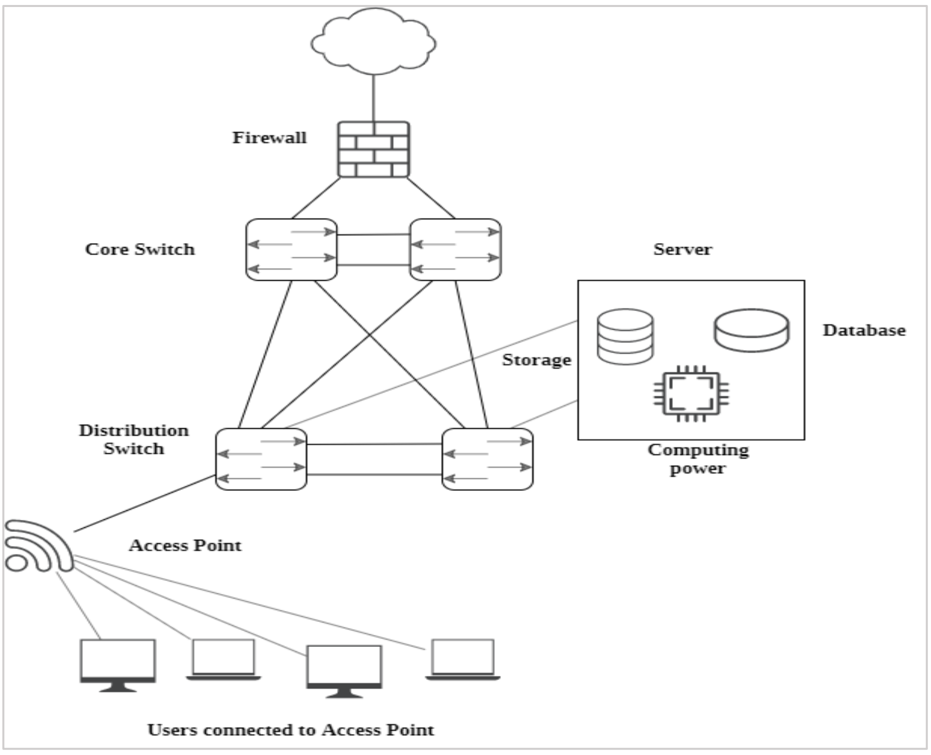


Figure 3.1 *Traditional company Architecture overview*

To solve the issues related to the above and previous architecture we opt in to create a modernized and optimized infrastructure that group up the essential layers of the infrastructure in one appliance and answer the demands of the company, the figure 3.2 shows the intended architecture for OMES Algérie.

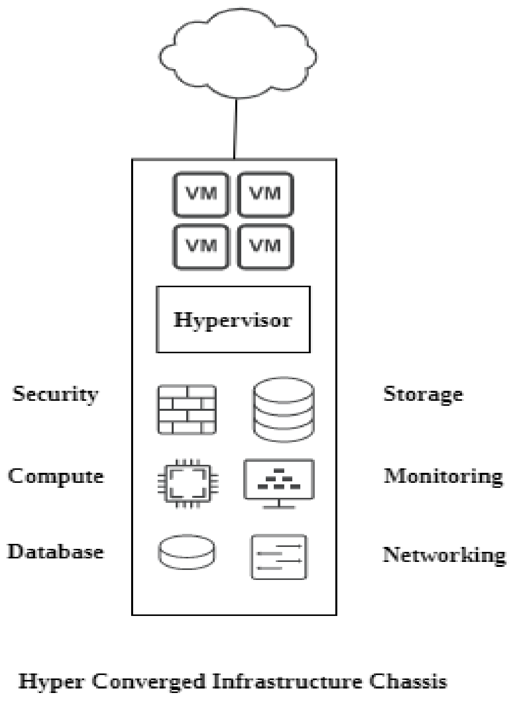


Figure 3.2 *Hyper-converged infrastructure proposition*

3.3 Proxmox

We are going to use PROXMOX VE 6.2-4 hypervisor for the many features it has related to our project goal in this memoir. Proxmox VE is a platform to run virtual machines and containers. It is based on Debian Linux, and completely open source. For maximum flexibility, they implemented two virtualization technologies - Kernel-based Virtual Machine (KVM) and container-based virtualization (LXC). One main design goal was to make administration as easy as possible.

You can use Proxmox VE on a single node, or assemble a cluster of many nodes. All management tasks can be done using our web-based management interface, Proxmox VE uses a RESTful API. They chose JSON as primary data format, and the whole API is formally defined using JSON Schema. This enables fast and easy integration for third party management tools like custom hosting environments.

The Proxmox VE storage model is very flexible. Virtual machine images can either be stored on one or several local storages or on shared storage like NFS and on SAN. There are no limits, you may configure as many storage definitions as you like. You can use all storage technologies available for Debian Linux. One major benefit of storing VMs on shared storage is the ability to live-migrate running machines without any downtime, as all nodes in the cluster have direct access to VM disk images.

Proxmox VE uses a bridged networking model. All VMs can share one bridge as if virtual network cables from each guest were all plugged into the same switch. For connecting VMs to the outside world, bridges are attached to physical network cards and assigned a TCP/IP configuration. For further flexibility, VLANs (IEEE 802.1q) and network bonding/aggregation are possible. In this way it is possible to build complex, flexible virtual networks for the Proxmox VE hosts, leveraging the full power of the Linux network stack.

Reasons on why we chose Proxmox:

- Open source Software
- Linux Kernel
- Web-based interface
- REST API

3.4 Ceph

Ceph is a both, self-healing and self-managing shared, reliable and highly scalable storage system, also is one of the popular and more utilized open source SDS Solutions and we chose it because it works smoothly with our system hardware and is integrated in the latest version of Proxmox as well as for what it offers us to do in our project.

3.5 Software Defined capabilities in our cluster

The reason we chose Hyper converged solution is because how much the software defined solutions helped us, in reaching the goal of the project, creating the cluster and managing it.

3.6 Compute and Storage

Clustered storage is the use of two or more storage servers (nodes) working together to increase performance, capacity, or reliability.

3.6.1 Cluster Creation

In our case, we used a 3-node cluster for each department to ensure all functionalities that we implemented in our project.

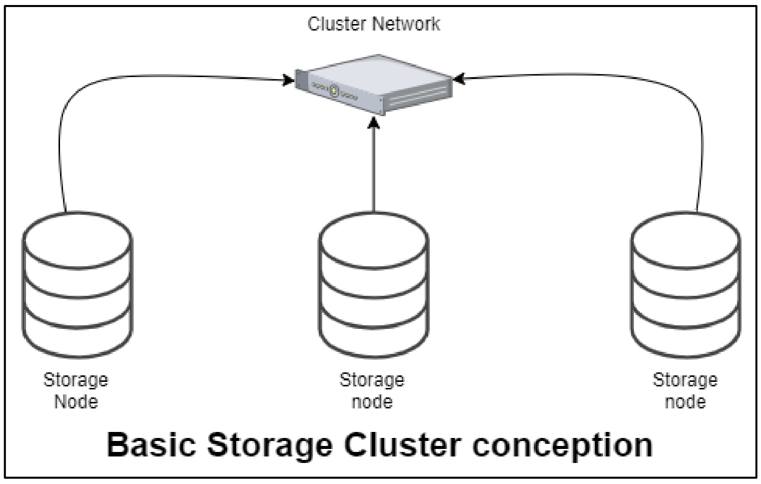


Figure 3.3 *Storage Cluster Components*

First of all, we need to set up 3 Proxmox nodes, for that we will use the iso downloaded from the official website, after installing the nodes like shown in the ANNEXE using these IP addresses

Node 1 GUI manager IP: 192.168.1.111

Node 1 replication IP: 192.168.2.111

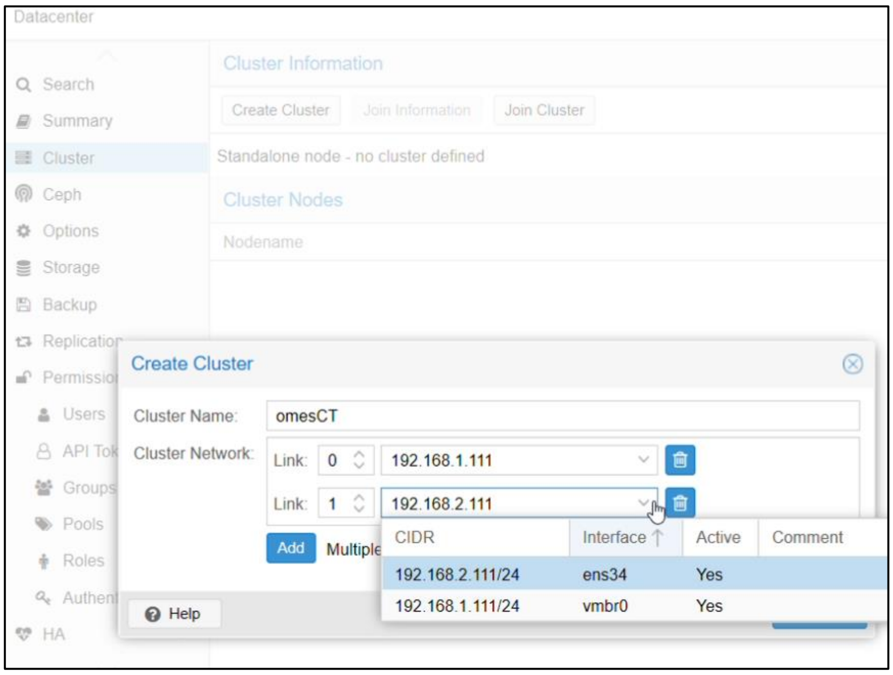
Node 2 GUI manager IP: 192.168.1.112

Node 2 replication IP: 192.168.2.112

Node 3 GUI manager IP: 192.168.1.113

Node 3 replication IP: 192.168.2.113

Then we proceed with the cluster creation, the figure 3.4 below shows how to do so:



Standalone node - no cluster defined

Cluster Name: omesCT

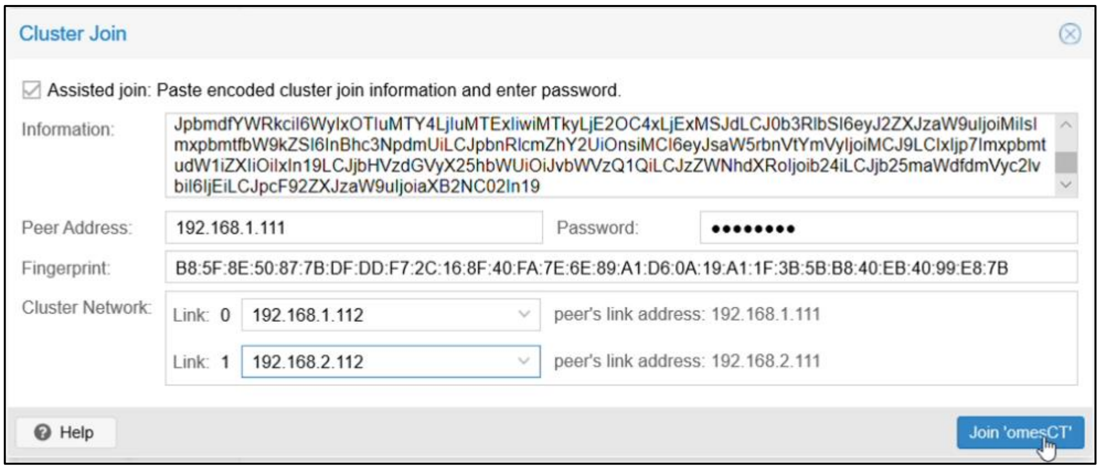
Cluster Network: Link: 0 192.168.1.111

Link: 1 192.168.2.111

CIDR	Interface ↑	Active	Comment
192.168.2.111/24	ens34	Yes	
192.168.1.111/24	vmbr0	Yes	

Figure 3.4 *Cluster creation with replication link*

After creating the cluster in the parent Node in this case we chose Node 1, we proceed with joining the other 2 nodes to the cluster with the following process shown in the figure 3.5 below:


Cluster Join

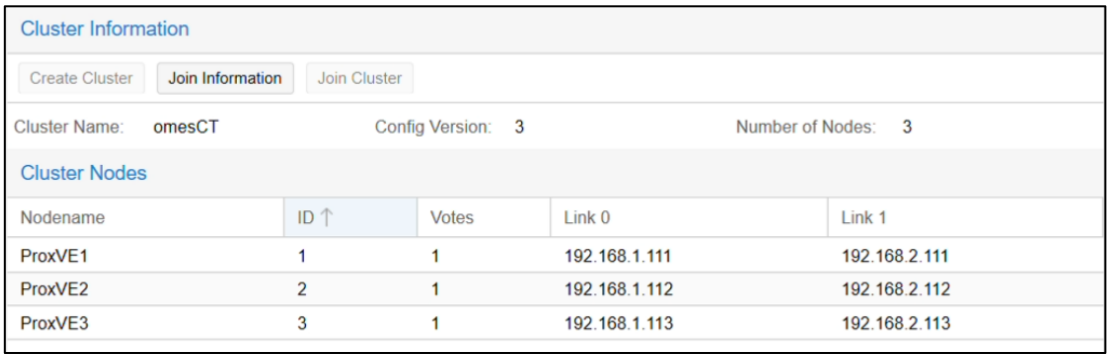
☑ Assisted join: Paste encoded cluster join information and enter password.

Information:
JpbmdfYWRkcil6WylxOTIuMTY4LjluMTExliwiMTkyLjE2OC4xLjExMSJdLCJ0b3RlbSI6eyJ2ZXJzaW9uljoiMilsl
mxpbmtfbW9kZSI6InBhc3NpdmUiLCJpbnRlcmZhY2UiOnsiMCI6eyJsaW5rbnVtYmVyljoiMCJ9LCIxljp7Imxpbmt
udW1iZXliOilxIn19LCJjbHVzdGVyX25hbWUiOiJvbWVzQ1QiLCJzZWNhdXRoljoib24iLCJjb25maWdfdmVyc2lv
bil6IjEiLCJpcF92ZXJzaW9uljoiaXB2NC02In19

Peer Address:	192.168.1.111	Password:	•••••••••
Fingerprint:	B8:5F:8E:50:87:7B:DF:DD:F7:2C:16:8F:40:FA:7E:6E:89:A1:D6:0A:19:A1:1F:3B:5B:B8:40:EB:40:99:E8:7B		
Cluster Network:	Link: 0	192.168.1.112	peer's link address: 192.168.1.111
	Link: 1	192.168.2.112	peer's link address: 192.168.2.111

Figure 3.5 *Node 2 joining the cluster using GUI in Proxmox*

We do the same thing with node 3, hence we end up with Proxmox cluster that can be managed by any GUI managing IP from the cluster in the figure 3.6:


Cluster Information

Cluster Name: omesCT Config Version: 3 Number of Nodes: 3

Cluster Nodes

Nodename	ID ↑	Votes	Link 0	Link 1
ProxVE1	1	1	192.168.1.111	192.168.2.111
ProxVE2	2	1	192.168.1.112	192.168.2.112
ProxVE3	3	1	192.168.1.113	192.168.2.113

Figure 3.6 *Cluster Nodes*

3.6.2 High Availability

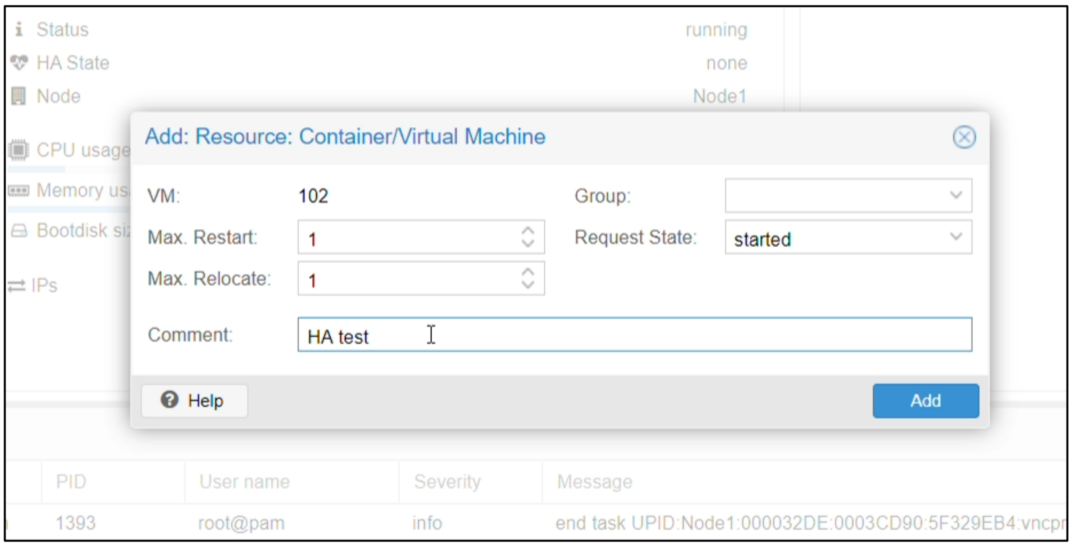
Virtualization environments like Proxmox VE make it much easier to reach high availability because they remove the “hardware” dependency. They also support redundancy all over the cluster. So, if one host fail, you can simply start those services on another host within your cluster.

To reach the HA in Proxmox VE we must follow certain requirement:

- At least three nodes in the cluster, preferably any odd number.
- Shared storage for VMs or containers.
- Hardware redundancy.

To ensure the availability of the different functionalities intended to OMES Algérie, we used Proxmox VE for its easy use and implementation of those important features.

First of all, we are going to activate the HA on the VM we want to keep safe and functioning in case of a node failure, there are two methods for this, either we use the GUI like on the figure 3.7 which is the easy way.



Status	running
HA State	none
Node	Node1

Add: Resource: Container/Virtual Machine

VM:	102	Group:	
Max. Restart:	1	Request State:	started
Max. Relocate:	1		
Comment:	HA test		

PID	User name	Severity	Message
1393	root@pam	info	end task UPID:Node1:000032DE:0003CD90:5F329EB4:vncpr

***Figure 3.7** High Availability Activated on VM 102*

Otherwise we can do it via the shell (CLI) by inserting the follow commands:

```
# ha-manager add vm:102
```

```
# ha-manager set vm:102 --state stopped
```

```
# ha-manager set vm:102 --state started
```

```
# qm start 102
```

As you can see in the figure 3.8 after activating high availability on the functioning VM in node 1, we are going to simulate the node failure to do so we will shut it down manually.

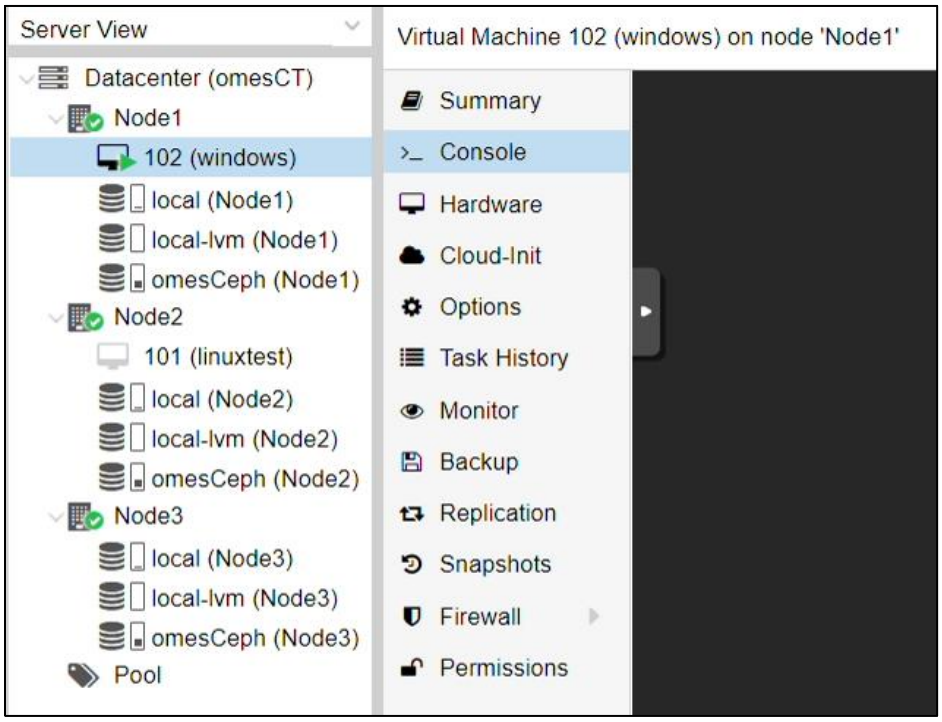
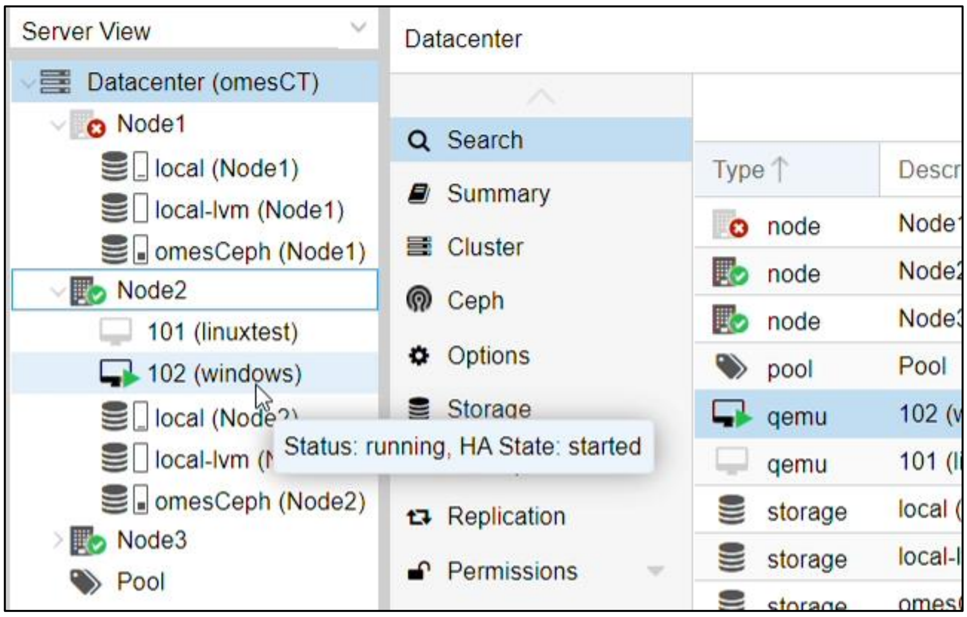


Figure 3.8 Node 1 running VM before failure

After simulating the node failure, the VM migrated with all its resources to node 2 due to the HA feature that was enabled on it as shown in figure 3.9:



Status: running, HA State: started

Figure 3.9 VM Migration to Node 2 due HA

3.6.3 Live migration

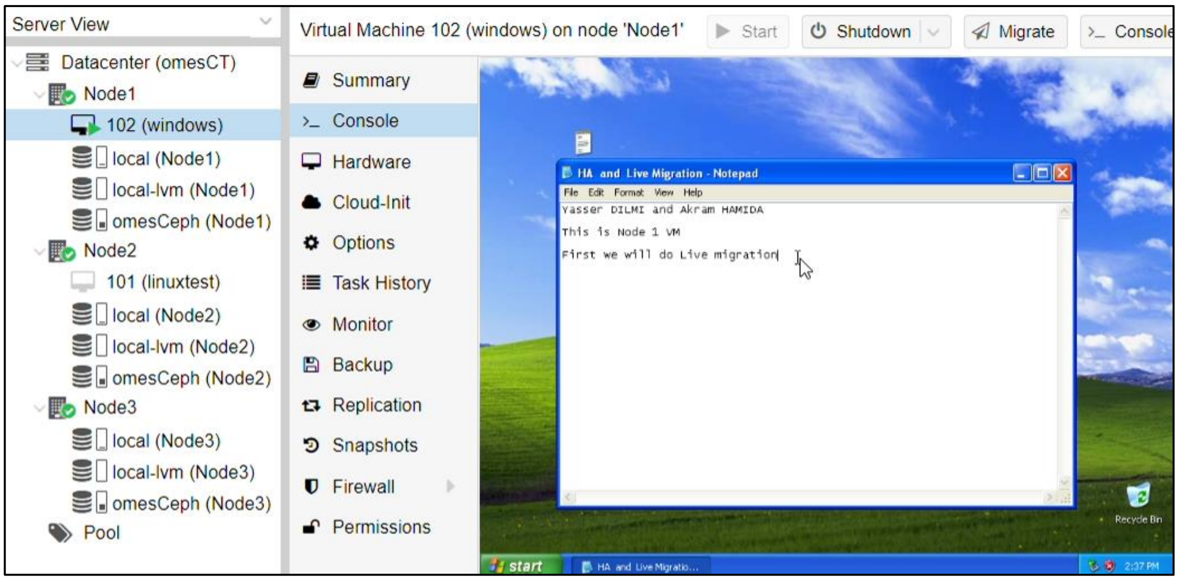
VM migration is possible, but usually it’s required to be offline, so live migration is the movement of a virtual machine from one node to another while continuously powered-up. When properly carried out, this process takes place without any noticeable effect from the point of view of the end user. Live migration allows an administrator to take a node offline for maintenance or upgrading without subjecting the system's users to downtime.

One of the most significant advantages of live migration is the fact that it facilitates proactive maintenance.

We will demonstrate how we can use this feature in our cluster either with CLI:

```
# qm migrate <vmid> <target>
```

live migration using the GUI is much simpler and easier, all you need to do is to select your VM after that you click on the migrate button on the Proxmox GUI to migrate it to a specific node in your cluster, this will allow us mostly to keep the functions of that Virtual machine usable and give us the ability to do the maintenance, the figure 3.10 below will show the process.

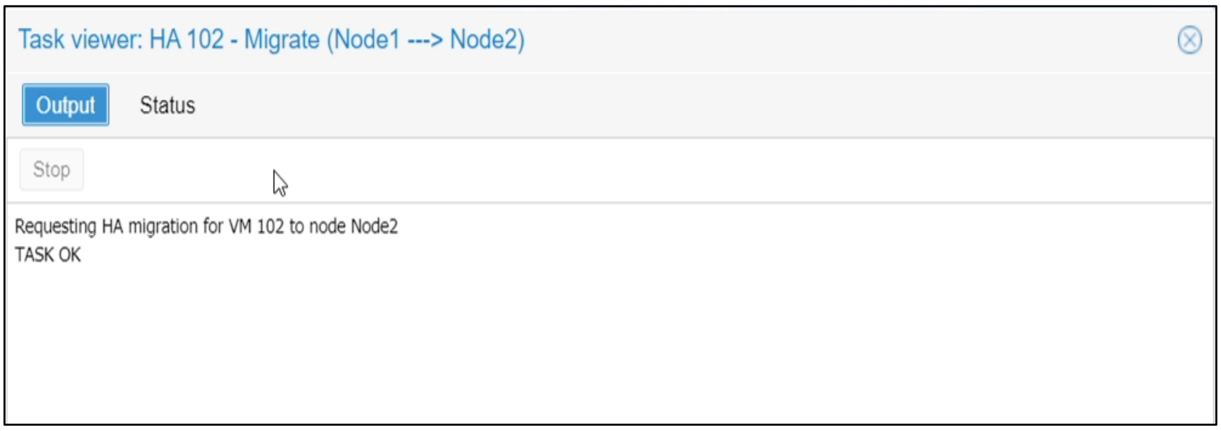

Virtual Machine 102 (windows) on node 'Node1'

HA and Live Migration - Notepad

Yasser DILMI and Akram HAMIDA
This is Node 1 VM
First we will do Live migration

Figure 3.10 VM functioning on Node 1

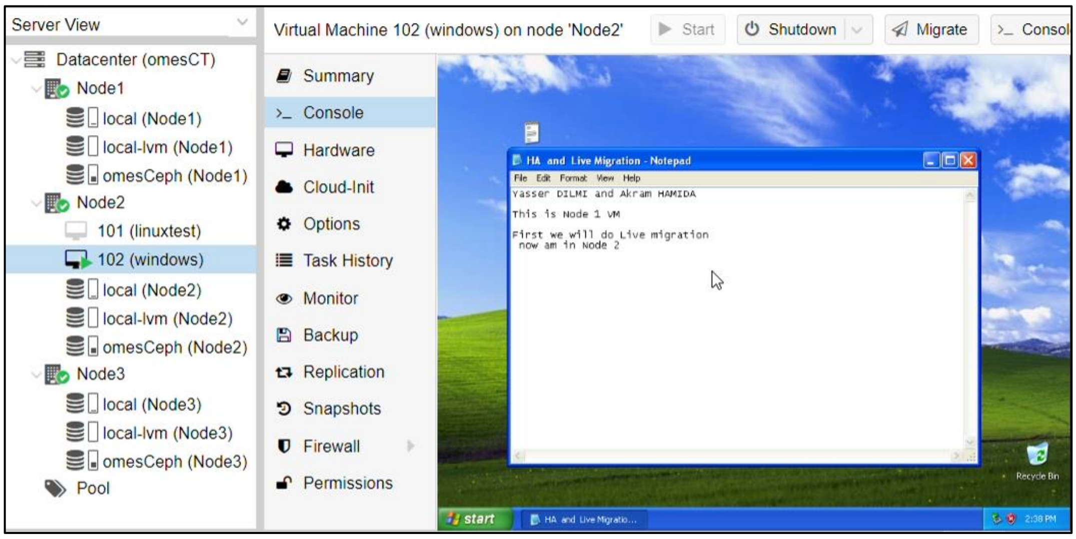
The moment you commence the live migration for a specific virtual machine (VM) a prompt on the screen will pop up showing you the requested migration and if the High availability option is enabled, the live migration process takes time depending on the hardware and resources.



```
Requesting HA migration for VM 102 to node Node2
TASK OK
```

Figure 3.11 *Process of Live Migrating a VM*

After this, Live migration is fully achieved with no perceived downtime, as a consequence the running system stays unaffected and there’s no data loss or any sort of technical issue while doing so.



```
Yasser DILMI and Akram HAMIDA

This is Node 1 VM

First we will do Live migration
 now am in Node 2
```

Figure 3.12 *VM 102 running on Node 2 after migration*

3.6.4 Snapshot

By using Proxmox VE snapshots, you can preserve the virtual machine state. A snapshot includes the contents of the virtual machine memory, virtual machine settings, and the state of all the virtual disks. The figure 3.13 below represents the process used to create a snapshot for a specific VM.

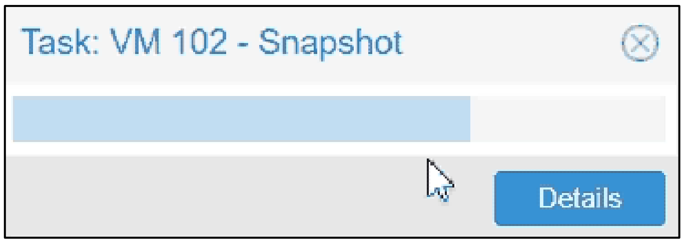
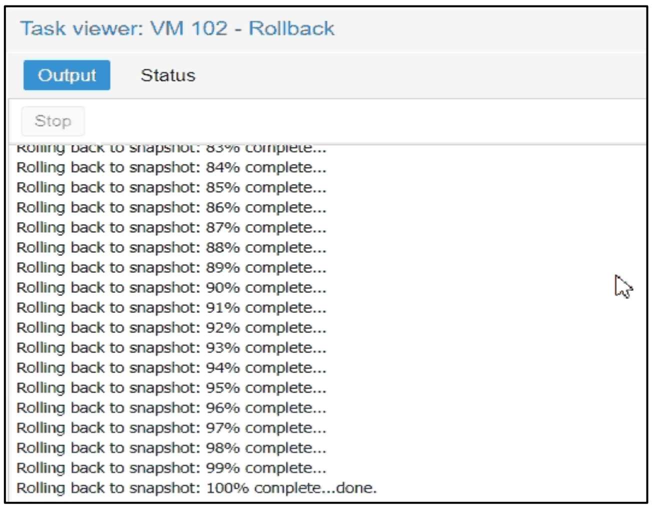


Figure 3.13 *Snapshot process for VM 102*

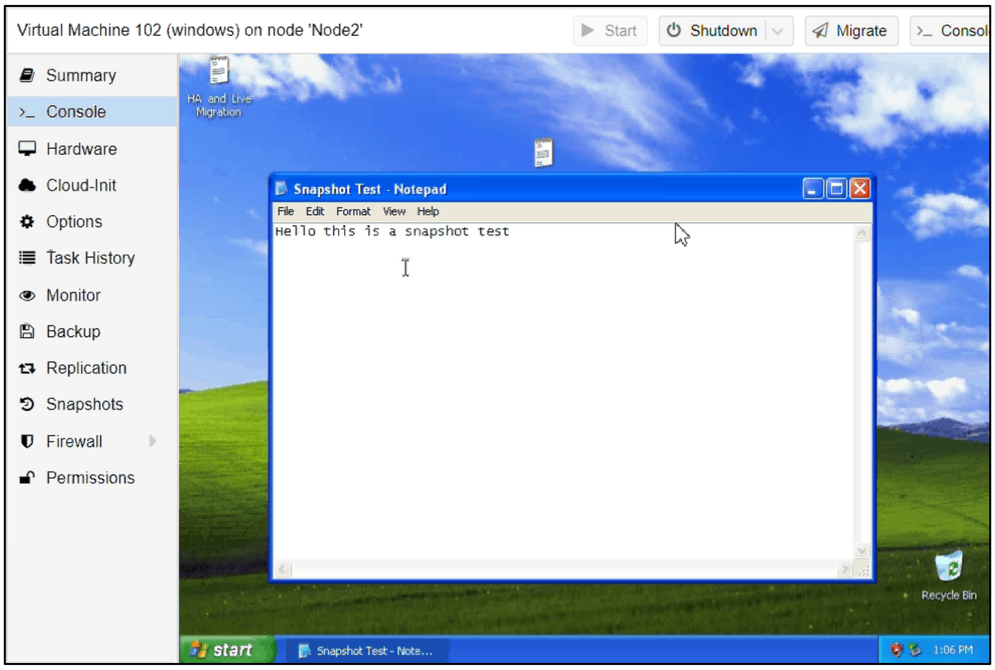
When the rollback shown in figure 3.14, you restore the memory, virtual disks and all settings of the virtual machine to the state they were in when you took the snapshot.



```
Rolling back to snapshot: 83% complete...
Rolling back to snapshot: 84% complete...
Rolling back to snapshot: 85% complete...
Rolling back to snapshot: 86% complete...
Rolling back to snapshot: 87% complete...
Rolling back to snapshot: 88% complete...
Rolling back to snapshot: 89% complete...
Rolling back to snapshot: 90% complete...
Rolling back to snapshot: 91% complete...
Rolling back to snapshot: 92% complete...
Rolling back to snapshot: 93% complete...
Rolling back to snapshot: 94% complete...
Rolling back to snapshot: 95% complete...
Rolling back to snapshot: 96% complete...
Rolling back to snapshot: 97% complete...
Rolling back to snapshot: 98% complete...
Rolling back to snapshot: 99% complete...
Rolling back to snapshot: 100% complete...done.
```

Figure 3.14 *Rollback process of VM 102*

The reason the snapshots are important is that when we execute some commands or do some operation the cluster, we need to make sure to make a replicate to the VM as a pre-caution for any unexpected events, that’s why before any maintenance to any function in the cluster a snapshot is required by the administration team.



```
Hello this is a snapshot test
```

Figure 3.15 *VM 102 after rollback status*

3.6.5 Backup

The difference between snapshots and backups is that backup are independent from the VM, in other words if you VM is removed you will still be able to restore it using a backup, on the same state the day you took it, unlike snapshots which are dependent to the VM, also the backup take more storage and could be stored anywhere in the cluster. The table 3-2 shows the differences between snapshots and backup.

Table 3-2 *Difference between backup and snapshot*

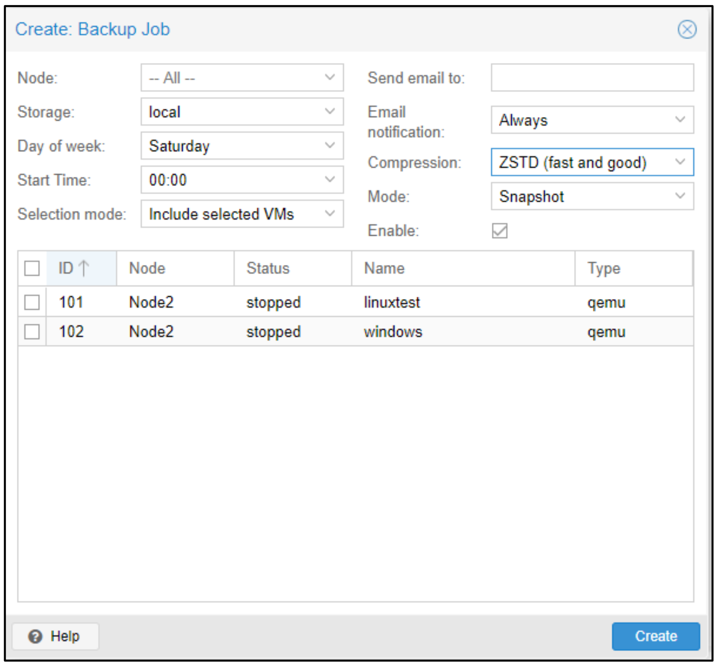
	Backup	**Snapshot**
Storage Capacity	Big	Small
Storage flexibility on cluster	✔	✕
VM independent	✔	✕
Automation	✔	✕
Time Consumption	Slow	Fast

Also, there are backup modes could be used depending on our needs, the table 3-3 explains those modes.

Table 3-3 *Backup modes explanation*

Backup mode	Explanation
Stop	This mode provides the highest consistency of the backup, at the cost of a short downtime in the VM operation. It works by executing an orderly shutdown of the VM, and then runs a background Qemu process to back up the VM data. After the backup is started, the VM goes to full operation mode if it was previously running. Consistency is guaranteed by using the live backup feature.
Suspend	This mode is provided for compatibility reason, and suspends the VM before calling the snapshot mode. Since suspending the VM results in a longer downtime and does not necessarily improve the data consistency, the use of the snapshot mode is recommended instead.
Snapshot	This mode provides the lowest operation downtime, at the cost of a small inconsistency risk. It works by performing a Proxmox VE live backup, in which data blocks are copied while the VM is running

The reason backups could help us in our project is that there will be some features installed in one of the node's VMs and those features are crucial so for an added safety measure to the cluster functioning we do those backups the figure 3.16 shows the process:

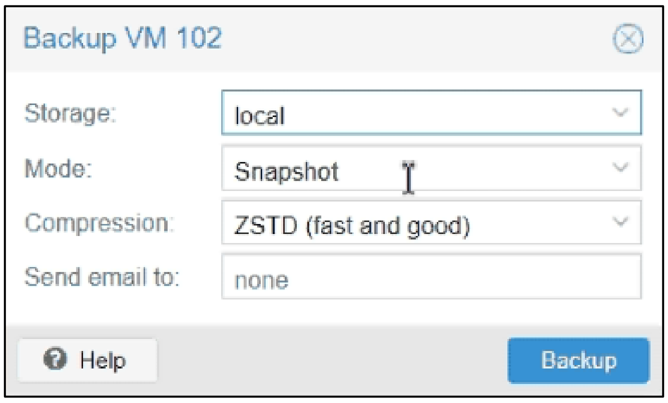

Create: Backup Job

Node:	-- All --
Storage:	local
Day of week:	Saturday
Start Time:	00:00
Selection mode:	Include selected VMs
Send email to:	
Email notification:	Always
Compression:	ZSTD (fast and good)
Mode:	Snapshot
Enable:	✓

ID	Node	Status	Name	Type
101	Node2	stopped	linuxtest	qemu
102	Node2	stopped	windows	qemu

Figure 3.16 *Backup job from the cluster*

As well as the full cluster backup, there is also an only VM backup when we tried to do maintenance risking the data loss the figure 3.17 shows how we did that:



Backup VM 102

Storage:	local
Mode:	Snapshot
Compression:	ZSTD (fast and good)
Send email to:	none

Figure 3.17 *Backup job from the VM*

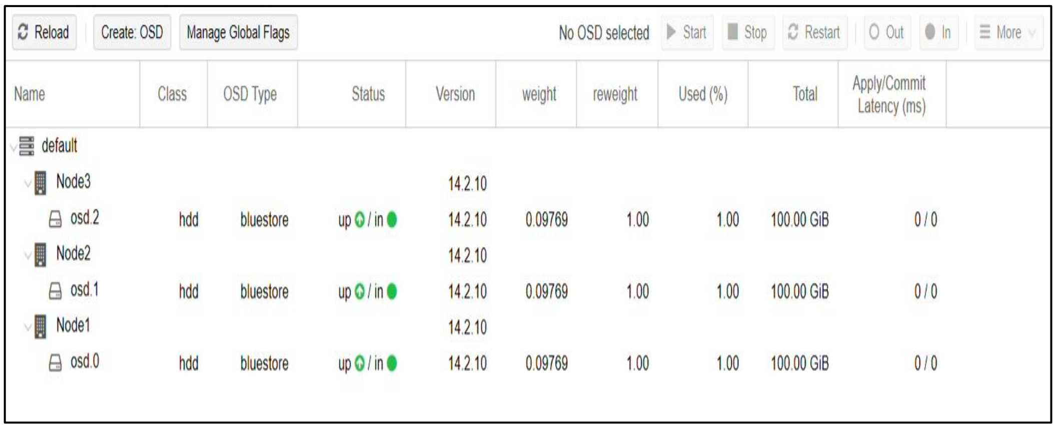
There are multiple compression methods we could use, we used the fast and good compression ZSTD type because it offers a very wide range of compression / speed trade-off, while being backed by a very fast decoder, also we chose mode snapshot to take the state of the VM on real time.

3.6.6 Object storage daemon (OSD)

ceph-osd is the object storage daemon for the Ceph distributed file system. It is responsible for storing objects on a local file system and providing access to them over the network.

3.6.6.1 OSD Creation

The following process shows the creation of ceph-osd, we will use the extra disk of 100GB to make OSDs for each node then we group them in a pool storage called omesCeph:



Name	Class	OSD Type	Status	Version	weight	reweight	Used (%)	Total	Apply/Commit Latency (ms)
default									
Node3				14.2.10					
osd.2	hdd	bluestore	up ● / in ●	14.2.10	0.09769	1.00	1.00	100.00 GiB	0 / 0
Node2				14.2.10					
osd.1	hdd	bluestore	up ● / in ●	14.2.10	0.09769	1.00	1.00	100.00 GiB	0 / 0
Node1				14.2.10					
osd.0	hdd	bluestore	up ● / in ●	14.2.10	0.09769	1.00	1.00	100.00 GiB	0 / 0

Figure 3.18 *OSDs of each node*

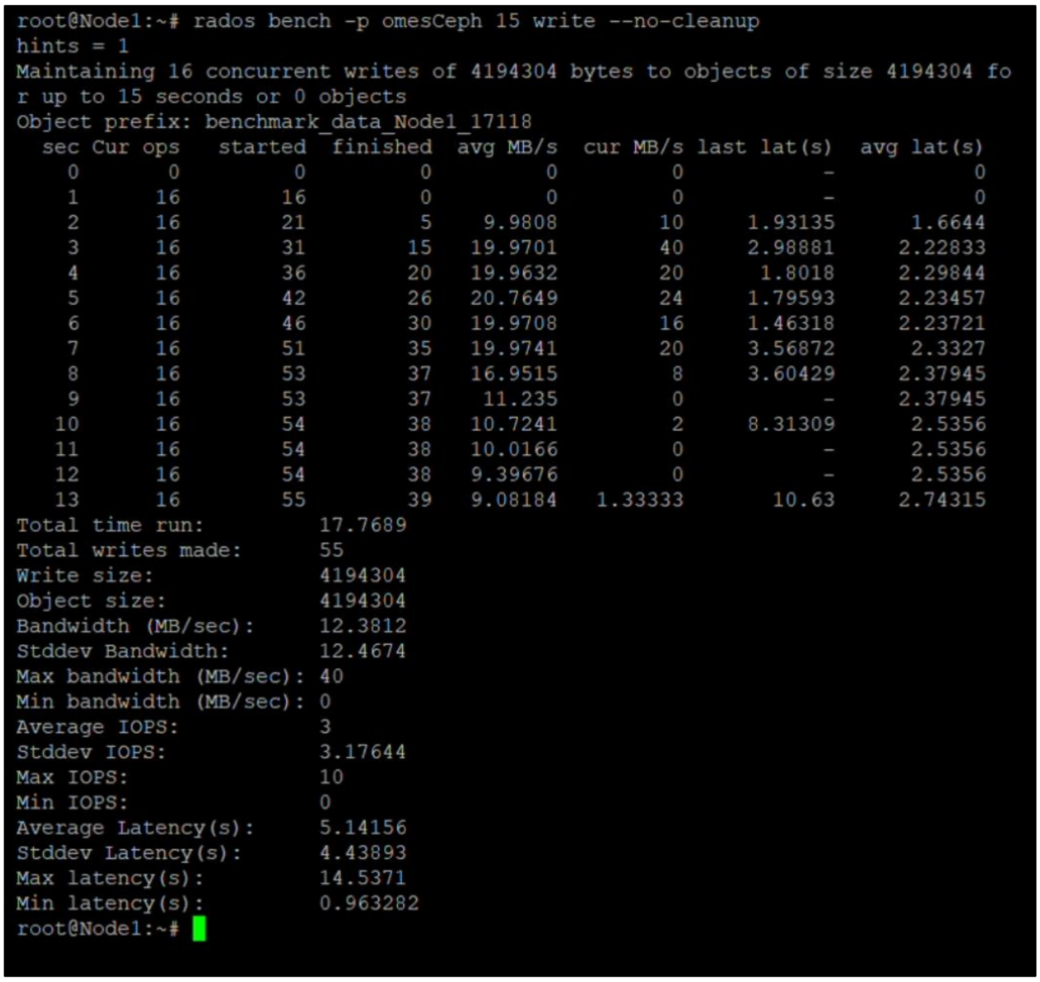
3.6.6.2 OSD benchmark

Ceph includes the **rados bench** command, designed specifically to benchmark a RADOS storage cluster. On the created OSD pool that is running on 7200 rpm hard drive we will perform a rados bench write, as shown below.

We can run it from any node in the cluster since they are all connected to the same OSD pool then we will execute the following command on Node1 for example:

```
#rados bench -p omesCeph 15 write –no-cleanup
```

The output of the above command is shown in the figure 3.19, as we can see that it executing concurrent writes of 4194304 bytes to objects of the same size.



```
root@Node1:~# rados bench -p omesCeph 15 write --no-cleanup
hints = 1
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304 fo
r up to 15 seconds or 0 objects
Object prefix: benchmark_data_Node1_17118
  sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)
    0       0         0         0         0         0           -           0
    1      16        16         0         0         0           -           0
    2      16        21         5    9.9808        10     1.93135      1.6644
    3      16        31        15   19.9701        40     2.98881     2.22833
    4      16        36        20   19.9632        20      1.8018     2.29844
    5      16        42        26   20.7649        24     1.79593     2.23457
    6      16        46        30   19.9708        16     1.46318     2.23721
    7      16        51        35   19.9741        20     3.56872      2.3327
    8      16        53        37   16.9515         8     3.60429     2.37945
    9      16        53        37    11.235         0           -     2.37945
   10      16        54        38   10.7241         2     8.31309      2.5356
   11      16        54        38   10.0166         0           -      2.5356
   12      16        54        38   9.39676         0           -      2.5356
   13      16        55        39   9.08184   1.33333       10.63     2.74315
Total time run:         17.7689
Total writes made:      55
Write size:             4194304
Object size:            4194304
Bandwidth (MB/sec):     12.3812
Stddev Bandwidth:       12.4674
Max bandwidth (MB/sec): 40
Min bandwidth (MB/sec): 0
Average IOPS:           3
Stddev IOPS:            3.17644
Max IOPS:               10
Min IOPS:               0
Average Latency(s):     5.14156
Stddev Latency(s):      4.43893
Max latency(s):         14.5371
Min latency(s):         0.963282
root@Node1:~#
```

Figure 3.19 *OSD Pool benchmark*

The output is giving us a good indicator how fast our cluster is writing data, which is 12MB/sec and the maximum bandwidth is 40MB/sec.

3.7 Network

Our aim after studying the OMES Algérie structure is to achieve this topology to provide the optimal network use for each department with good quality of service and security.

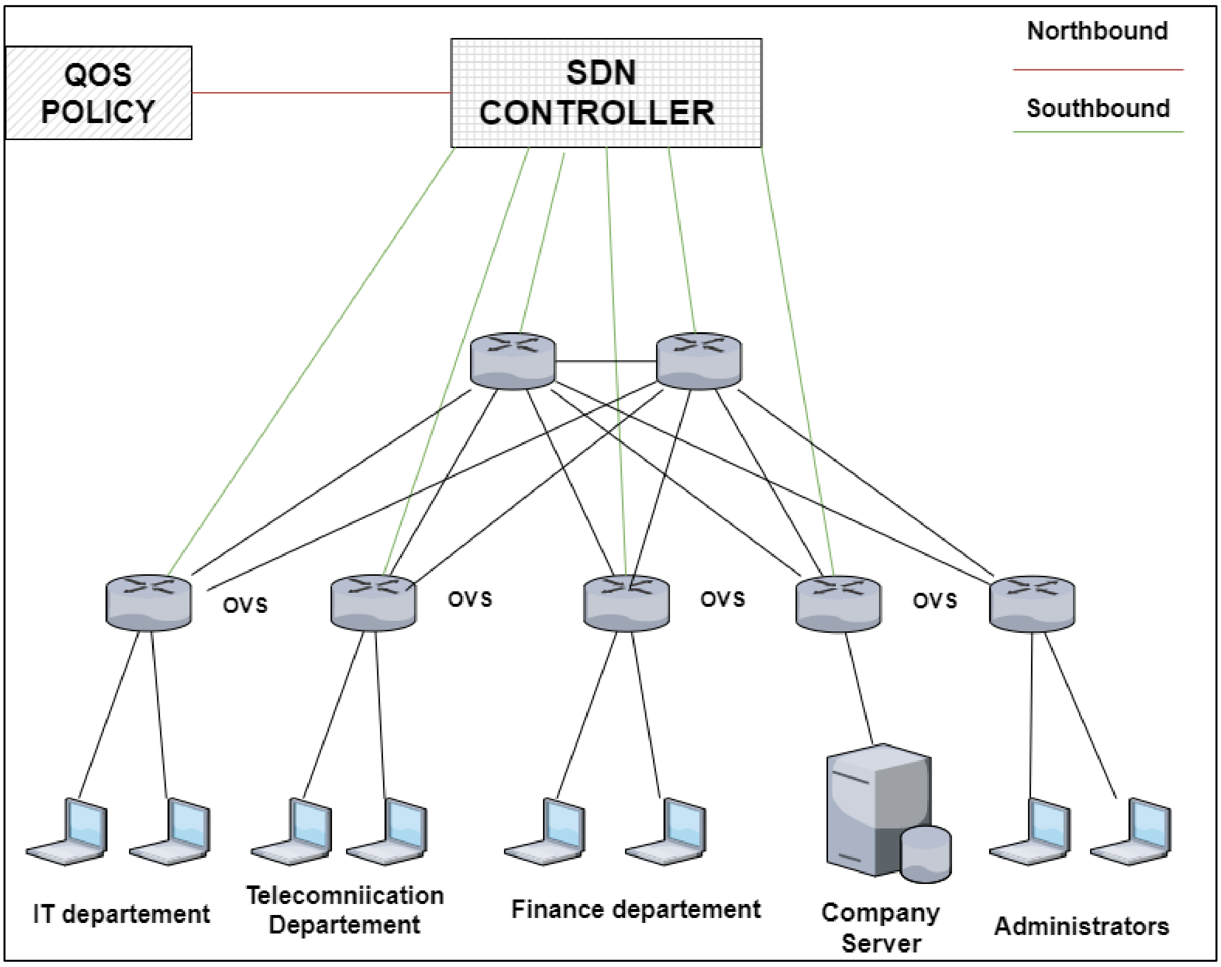


Figure 3.20 *Network overview of OMES Algérie structure*

3.7.1 Floodlight

One of Big Switch Networks major contributions to the open source community is an OpenFlow controller based on Java and under Apache license named Floodlight.

The architecture of this controller is based on Big Network Controller (BNC). Like many other controllers, when we run Floodlight, the operations of the north and south-facing controllers become active. In other words, when you run Floodlight, the controller, and all configured module applications execute as well. Nordic API (Application Programming Interface) REST (representational state transfer) exposed by all current modules become available via the specified REST port. All application can interact (retrieve information and invoke services) with the controller by sending HTTP REST commands. On the other hand, towards the south, the provider module of Floodlight will start listening on the Transmission Control Protocol (TCP) port specified by OpenFlow connections from OpenFlow switches.

The term "modular architecture" is used to describe the architecture of the controller Floodlight. The basic architecture includes various modules, this module also translates the messages OpenFlow into Floodlight events. Floodlight also offers examples of applications, including a firewall.

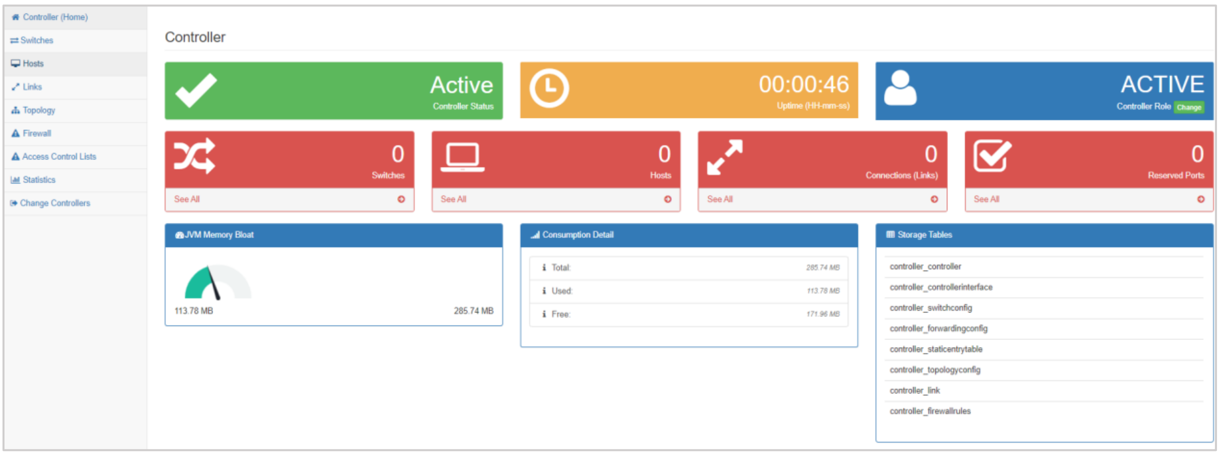


Figure 3.21 *Floodlight web GUI*

The table 3-4 shows the characteristics of floodlight controller:

Table 3-4 *Characteristics of Floodlight*

Characteristics	Floodlight
Developed by	Big Switch Networks
Supported by	Big Switch Networks
Language written in	Java
Supported Languages	Java, Python
Open source	Yes
User Interface	Web
Virtualization	Mininet, OpenvSwitch
Interface	Southbound (OpenFlow) Northbound (Java, REST)
Platform	Linux, MAC, Windows

3.7.2 Mininet

Mininet is a network emulator we used, it creates a network of virtual hosts, switches, and controllers and links. In other words, Mininet is an SDN environment emulator, developed by Stanford University, and can be used to build virtual networks. Mininet supports research, development, learning, prototyping, testing, debugging and any other task that can benefit from a complete experimental network on a computer.

Mininet:

- Can be used out of the box without programming, but also provides a simple Python API
- and scalable for network creation and experimentation.
- Supports custom topologies and includes a basic set of topologies set up.
- Provides a simple and inexpensive network testbed for application development OpenFlow
- Allows complex topology tests to be performed without the need to wire a physical network
- Includes a topology and OpenFlow-sensitive CLI for debugging or running network-wide tests.
- Allows several simultaneous developers to work independently on the same topology.

***Table 3-5** Mininet overview*

General	Mininet
Free software	Yes
Open Source	Yes
Supported in Windows	No
Supported in Linux	Yes
Simulation Mode	No
Emulation Mode	Yes

Hosts created by Mininet run standard Linux network software (standard Linux network software), the switches are of the OpenvSwitch type which support the OpenFlow protocol. and SDN, via a:

- Command line.

- Interactive interface.
- Python Script.

3.7.3 OpenvSwitch

It is an essential component in our emulation, OpenvSwitch (OVS) is a widely used tool in the SDN world. It is a virtual switch under the Apache 2.0 open source license that is used to interconnect virtual machines.

OpenvSwitch is therefore a software implementation of an Ethernet switch. Concretely, it consists of a service (ovs-vswitchd) and a kernel module (openvswitch_mod). The service allows to effectively switch packets to the right virtual ports, while the kernel module allows to capture the traffic coming from the network interfaces. To work like any switch, OpenvSwitch uses the notion of ports. Each port is made of one or several interfaces, which correspond to interfaces of the host system (logical or physical). It has many of the features of a layer 2 switch and even a layer 3 switch, for example:

- VLAN tag (802.1Q standard).
- Protocol spanning tree.
- Quality of Service (QoS).
- Supports the OpenFlow Protocol.

To know the global state of a switch we use this command line:

```
#sudo ovs-vsctl show
```

The creation of a switch is done by the following command line:

```
#sudo ovs-vsctl add-br   <virtual name of the switch>
```

Then we add ports to the created switch by the following command line:

```
#sudo ovs-vsctl add-port <virtual name of the switch>   <name of the port>
```

3.7.4 iPerf

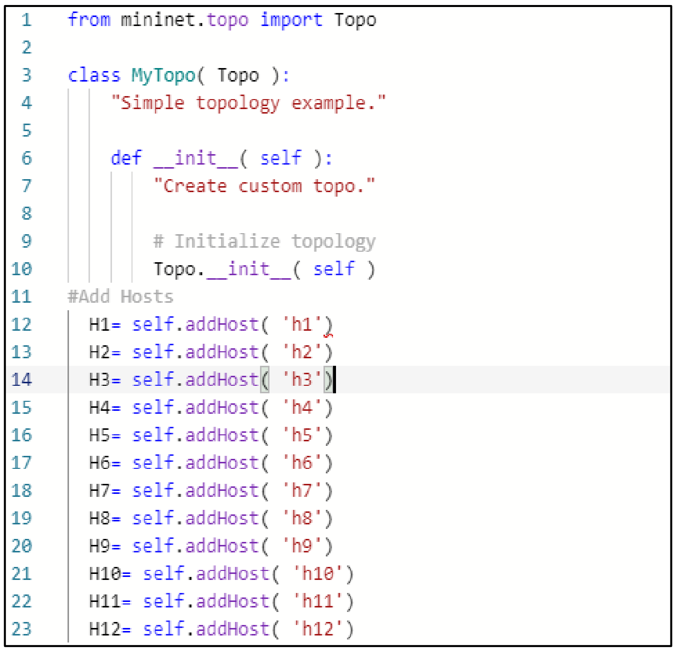
iPerf is a live network measurement tool whose main objective is to determine the maximum achievable throughput on the network. It can be used for other measurements such as jitter and frame/packet loss. It is customizable to generate UDP and TCP packets from different MTUs with a specified rate and transmission interval.

This tool acquires its statistics based on the number of packets transmitted during the transmission time and interval. It consists of two elements, a client and a server. The client generates traffic to the server according to the desired flow specifications, and determines an average transmitted bandwidth, while the server receives the generated traffic, performs statistics, and sends the results back to the client. iPerf works at the application level, which means that the measured throughput is not exclusive of link performance, but also of packet processing via the TCP / IP model. Additionally, iPerf runs at the user space level of the Linux operating system, working on top of the system architecture and using system calls to utilize resources.

3.7.5 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. we used Python to create our own topology in Mininet network emulator based on OMES Algérie department conception, we used the following script to create our own topology that we worked on.

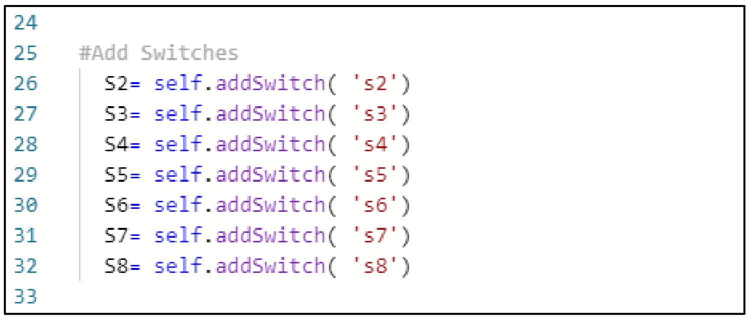
We started with writing the script of adding hosts to the topology we are working on and the figure 3.22 shows it.



```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )
#Add Hosts
    H1= self.addHost( 'h1')
    H2= self.addHost( 'h2')
    H3= self.addHost( 'h3')
    H4= self.addHost( 'h4')
    H5= self.addHost( 'h5')
    H6= self.addHost( 'h6')
    H7= self.addHost( 'h7')
    H8= self.addHost( 'h8')
    H9= self.addHost( 'h9')
    H10= self.addHost( 'h10')
    H11= self.addHost( 'h11')
    H12= self.addHost( 'h12')
```

Figure 3.22 *Mininet custom topology host creation snippet*

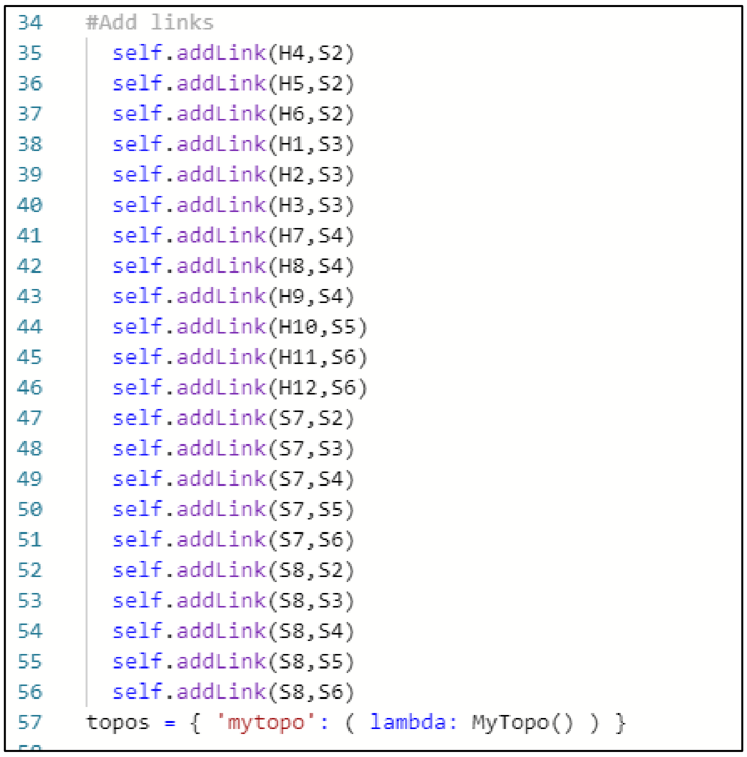
After that we continue the script writing by adding the needed switches



```

 #Add Switches
   S2= self.addSwitch( 's2')
   S3= self.addSwitch( 's3')
   S4= self.addSwitch( 's4')
   S5= self.addSwitch( 's5')
   S6= self.addSwitch( 's6')
   S7= self.addSwitch( 's7')
   S8= self.addSwitch( 's8')

```

Figure 3.23 *Mininet custom topology switches creation snippet*

After the hosts and switches creation we need to ensure their connectivity by adding bidirectional links between hosts and switches and eliminate the one single point of failure, hence having multiple routes.



```
 #Add links
   self.addLink(H4,S2)
   self.addLink(H5,S2)
   self.addLink(H6,S2)
   self.addLink(H1,S3)
   self.addLink(H2,S3)
   self.addLink(H3,S3)
   self.addLink(H7,S4)
   self.addLink(H8,S4)
   self.addLink(H9,S4)
   self.addLink(H10,S5)
   self.addLink(H11,S6)
   self.addLink(H12,S6)
   self.addLink(S7,S2)
   self.addLink(S7,S3)
   self.addLink(S7,S4)
   self.addLink(S7,S5)
   self.addLink(S7,S6)
   self.addLink(S8,S2)
   self.addLink(S8,S3)
   self.addLink(S8,S4)
   self.addLink(S8,S5)
   self.addLink(S8,S6)
 topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Figure 3.24 *Mininet custom topology link attachment snippet*

After executing the above python script resulted in the following topology that represent the OMES Algérie.

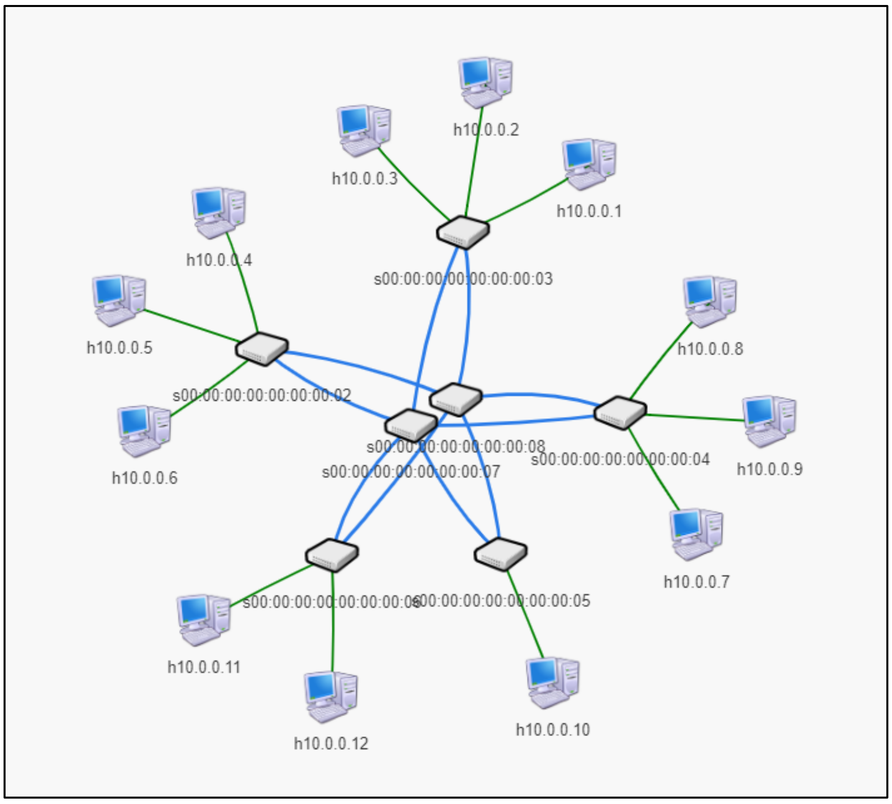


Figure 3.25 *Floodlight topology of OMES Algérie*

3.7.6 Quality of Service

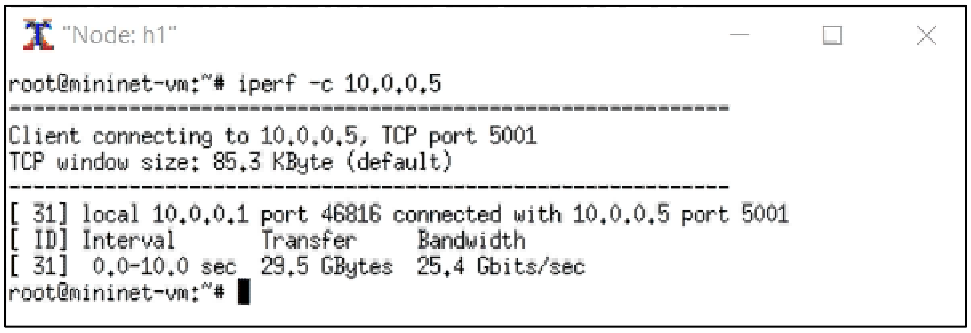
QoS controls and manages network resources by setting priorities for specific types of data on the network, in our Case we Applied a Bandwidth control type of quality of Service, we limited certain hosts to a certain Bandwidth speed so that it will be used optimally by each employee.

This Bandwidth control is Flexible, we can change it at any time however we like and easily apply it to the whole network.

First of all, we launch the topology on Mininet using the below command line:

```
#sudo mn --custom ~/mininet/custom/tp.py --topo network  --switch ovsk --
controller=remote,ip=192.168.1.47,port=6633 --ipbase=10.0.0.0/8
```

Figure 3.26 shows the output of ping test from host 1 to host 5 before applying QoS bandwidth limitation:



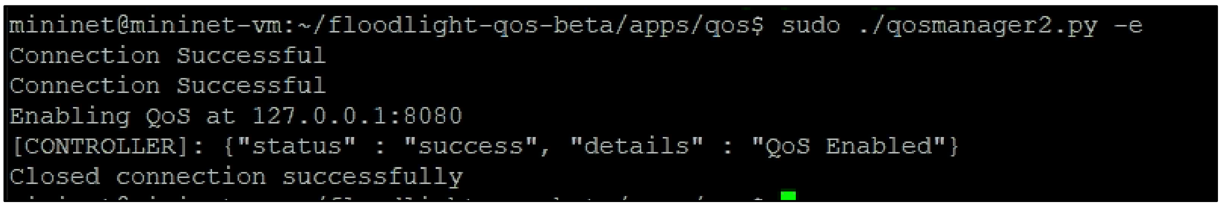
```
"Node: h1"
root@mininet-vm:~# iperf -c 10.0.0.5
------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 31] local 10.0.0.1 port 46816 connected with 10.0.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 31]  0.0-10.0 sec  29.5 GBytes  25.4 Gbits/sec
root@mininet-vm:~# 
```

Figure 3.26 *Host 1 Bandwidth before applying QoS*

As you can see the bandwidth is 25.4 Gbits/s without the limitation now we enable the quality of service queue using the below command

#sudo./qosmanager2.py -e

The figure 3.27 shows the output after activating the above command.



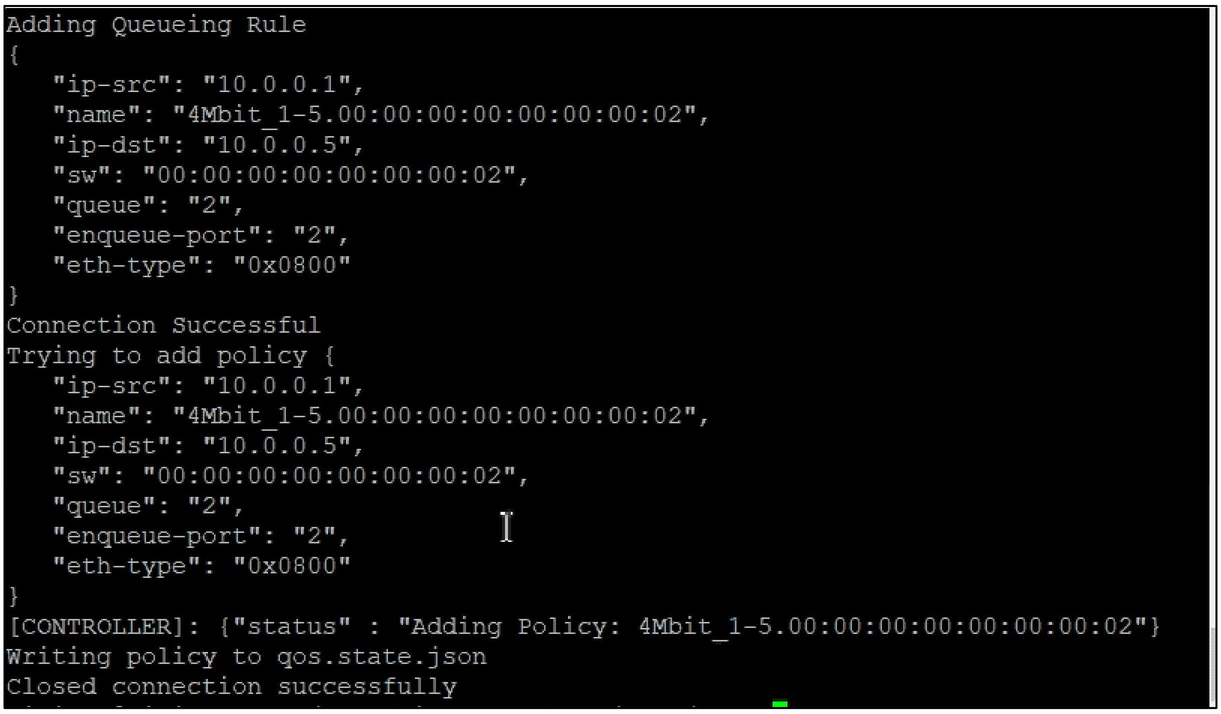
```
mininet@mininet-vm:~/floodlight-qos-beta/apps/qos$ sudo ./qosmanager2.py -e
Connection Successful
Connection Successful
Enabling QoS at 127.0.0.1:8080
[CONTROLLER]: {"status" : "success", "details" : "QoS Enabled"}
Closed connection successfully
```

Figure 3.27 *QoS Enabled in the controller*

Now we will be activating the quality of service on host 1 and we will limit the bandwidth to 4Mbits using the below command in the shell.

#sudo ./qospath2.py -a -S 10.0.0.1 -D 10.0.0.5 -N 4Mbit_1-5 -J '{"eth-type":"0x0800","queue":"2"}'

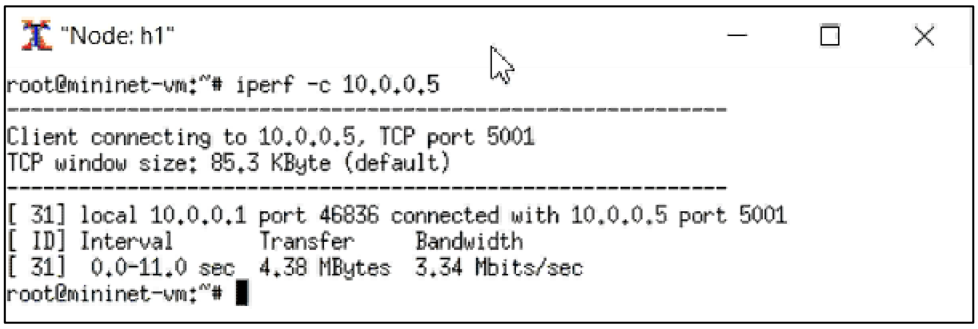
The output of the above command is demonstrated in the figure 3.27 showing that the queue rule has been added to the qosmanager in the controller.



```
Adding Queueing Rule
{
   "ip-src": "10.0.0.1",
   "name": "4Mbit_1-5.00:00:00:00:00:00:00:02",
   "ip-dst": "10.0.0.5",
   "sw": "00:00:00:00:00:00:00:02",
   "queue": "2",
   "enqueue-port": "2",
   "eth-type": "0x0800"
}
Connection Successful
Trying to add policy {
   "ip-src": "10.0.0.1",
   "name": "4Mbit_1-5.00:00:00:00:00:00:00:02",
   "ip-dst": "10.0.0.5",
   "sw": "00:00:00:00:00:00:00:02",
   "queue": "2",
   "enqueue-port": "2",
   "eth-type": "0x0800"
}
[CONTROLLER]: {"status" : "Adding Policy: 4Mbit_1-5.00:00:00:00:00:00:00:02"}
Writing policy to qos.state.json
Closed connection successfully
```

Figure 3.28 *QoS Added Policy Rule*

Figure 3.29 shows the bandwidth of the path from host 1 to host 5 after applying the Bandwidth QoS.



```
root@mininet-vm:~# iperf -c 10.0.0.5
------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 31] local 10.0.0.1 port 46836 connected with 10.0.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 31]  0.0-11.0 sec  4.38 MBytes  3.34 Mbits/sec
root@mininet-vm:~#
```

Figure 3.29 *Host 1 Bandwidth after applying QoS*

3.7.7 Firewall

The security aspect is one of the crucial things in datacenters, so to secure our implemented approach we are going to use the firewall in the SDN controller to do as intended, first we need to enable it in the controller using this command:

```
#curl http://192.168.1.47:8080/wm/firewall/module/enable/json -X PUT -d ' '
```

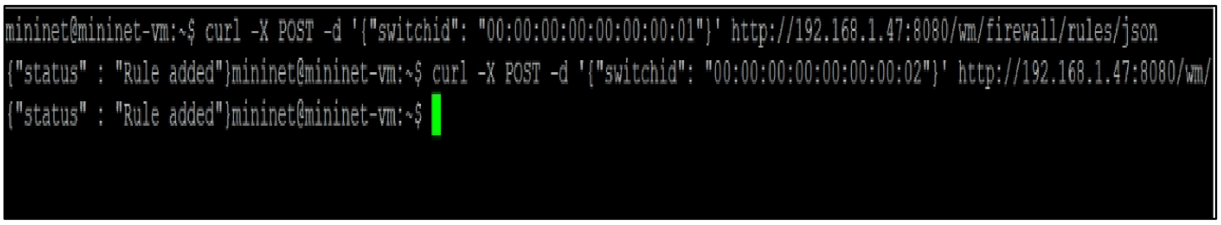
After we activate the firewall all the flows will be dropped due to implicit deny. To enable communication, rules must be inserted into the firewall so that any incoming packet is checked with the rule before an ALLOW or DENY action is performed on the packet.

To implement the firewall, we will be using a scenario where we authorize all flow passing between switch with MAC «00:00:00:00:00:00:00:01» and switch with MAC «00:00:00:00:00:00:00:02»

```
#curl -X POST -d '{"switchid": "00:00:00:00:00:00:00:01"}' http://192.168.1.47:8080/wm/firewall/rules/json
```

```
#curl -X POST -d '{"switchid": "00:00:00:00:00:00:00:02"}' http://192.168.1.47:8080/wm/firewall/rules/json
```

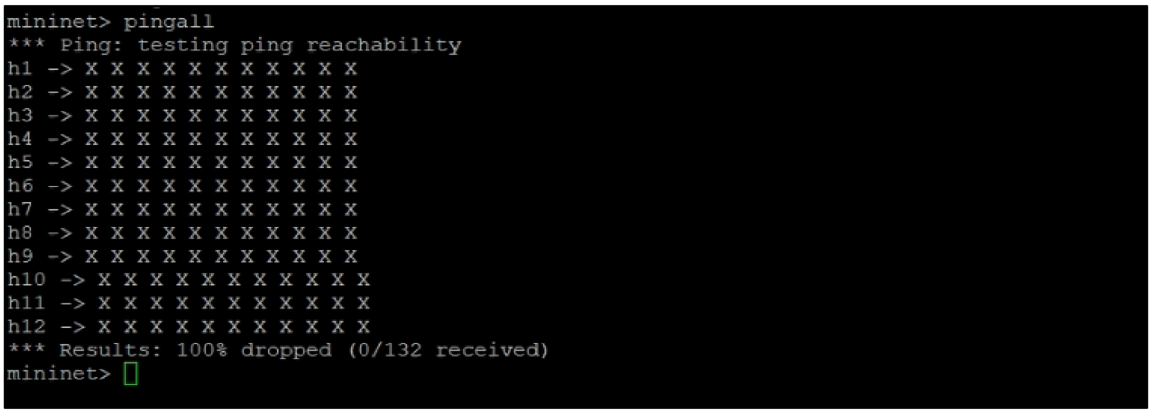
The output of the above commands is shown in figure 3.30:



```
mininet@mininet-vm:~$ curl -X POST -d '{"switchid": "00:00:00:00:00:00:00:01"}' http://192.168.1.47:8080/wm/firewall/rules/json
{"status" : "Rule added"}mininet@mininet-vm:~$ curl -X POST -d '{"switchid": "00:00:00:00:00:00:00:02"}' http://192.168.1.47:8080/wm/
{"status" : "Rule added"}mininet@mininet-vm:~$
```

Figure 3.30 *Rule added to the firewall*

The "POST" method will insert the rules into the firewall rules file, each incoming packet will be checked against these rules and if a match is found the action (ALLOW / DENY) will be applied depending to the rule.

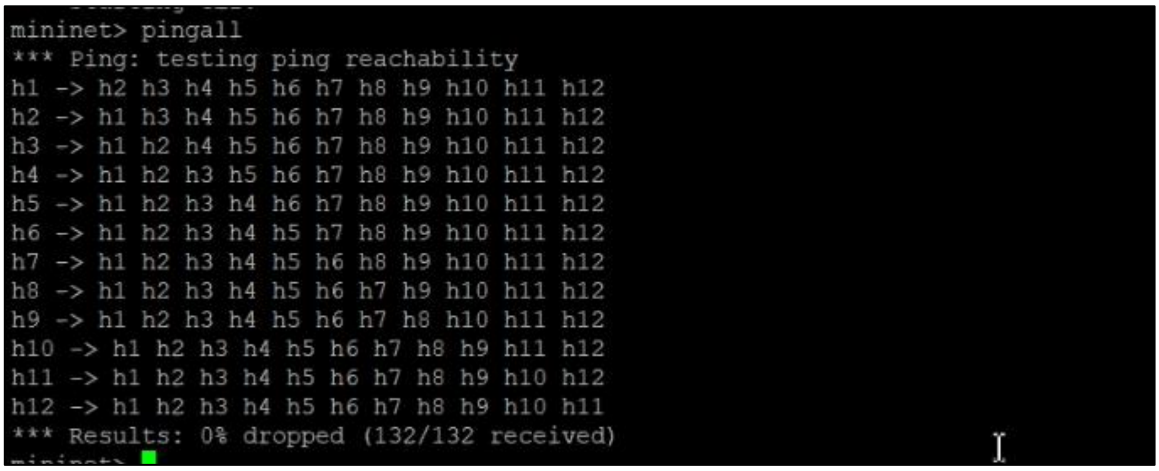


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X X X X X X X
h2 -> X X X X X X X X X X X
h3 -> X X X X X X X X X X X
h4 -> X X X X X X X X X X X
h5 -> X X X X X X X X X X X
h6 -> X X X X X X X X X X X
h7 -> X X X X X X X X X X X
h8 -> X X X X X X X X X X X
h9 -> X X X X X X X X X X X
h10 -> X X X X X X X X X X X
h11 -> X X X X X X X X X X X
h12 -> X X X X X X X X X X X
*** Results: 100% dropped (0/132 received)
mininet>
```

Figure 3.31 *Ping before adding the rules*

The figure 3.31 demonstrate the results of a ping after activating the firewall and before adding the rules, we can see that the flow is dropped.

The figure 3.32 demonstrate that the ping now is reachable due the rules added to the firewall:



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
```

Figure 3.32 *Ping after adding rules to firewall*

We can add rules that DENY the traffic between a specific source MAC to a specific destination MAC using these commands:

```
#curl -X POST -d '{"src-mac": "00:00:00:00:00:00:00:20", "dst-mac":
"00:00:00:00:00:00:00:10","action" : "DENY" }' http://192.168.1.47:8080/wm/firewall/rules/json
```

```
#curl -X POST -d '{"src-mac": "00:00:00:00:00:00:00:10", "dst-mac":
"00:00:00:00:00:00:00:20","action" : "DENY" }' http://192.168.1.47:8080/wm/firewall/rules/json
```

Further to what we already demonstrated, we can also DENY or ALLOW access between hosts IPs, this will allow us to restrict undesired access from within the LAN to the datacenter as well, an example is shown in the commands that follows between host 10.0.0.1 and 10.0.0.5

```
#curl -X POST -d ‘{“src- ip”: “10.0.0.1”, “dst-ip": "10.0.0.5","dl-type":"ARP", "action" : "DENY" }'
http://192.168.1.47:8080/wm/firewall/rules/json

#curl -X POST -d ‘{“src- ip”: “10.0.0.5”, “dst- ip ": "10.0.0.1","dl-type":"ARP", "action" : "DENY" }'
http://192.168.1.47:8080/wm/firewall/rules/json
```

```
#curl -X POST -d '{"src- ip": "10.0.0.1", "dst- ip ": "10.0.0.5","nw-proto":"ICMP", "action" : "DENY"
}' http://192.168.1.47:8080/wm/firewall/rules/json

#curl -X POST -d '{"src- ip": "10.0.0.5", "dst- ip ": "10.0.0.1," nw-proto":"ICMP", "action" : "DENY"
}' http://192.168.1.47:8080/wm/firewall/rules/json
```

3.8 Conclusion

This chapter has been devoted to details of the implementation and performance study of the suggested cluster; It first gives an overview of the intended architecture and the used tools. After that, the design and implementation of the functionalities requested. Finally, we discuss the results obtained after each test either on the compute and storage part or the Network part.

4 General Conclusion

IT businesses need an agile and reliable infrastructure, to deal with the explosion of massive data and handling the increasingly gigantic volumes of traffic. This led to the emergence of new software defined solutions embedding ever more intelligence in the datacenters. These IT trends is where the strength of hyper-convergence lies, which no longer evolves with the heaviness of hardware but with the speed of the software.

We first highlighted the importance of virtualization in evolving the traditional IT datacenters, we then explained the story of hyper-convergence, talking about two of its key components, the SDC and SDS. After that we introduced SDN and a comparison between the traditional network and SDN, explaining this latter architecture and how it functions. We finally showed how we implement those three software-defined solutions together to create our hyper converged architecture solution for this project.

The results of our project were satisfying, and we can say that we have reached the goal of the requested objective which was:

- Datacenter high availability, recovery and maintenance fundamental functionalities.
- Network access restriction.
- Bandwidth limitation on controller QoS.

This internship was an enriching experience from a technical point of view, in terms of the technologies covered. Both in terms of network and system administration and in terms of discovering the business world.

However, the full development of functionalities for such a project was never fully achieved for due to constraints such as Covid-19, hardware resources and time. So, in order to improve this present work, we propose to:

- Apply disaster recovery system in the datacenter.
- Apply intelligent QoS based on flow and traffic.
- Add more security to the datacenter.

Annex 1: Nodes and Ceph installations

To create our cluster, we need first to install Proxmox on the nodes following these steps:

1. Proxmox Download.

We start by downloading Proxmox from the official website

https://www.proxmox.com/en/downloads/item/proxmox-ve-6-2-iso-installer

2. VM Configuration.

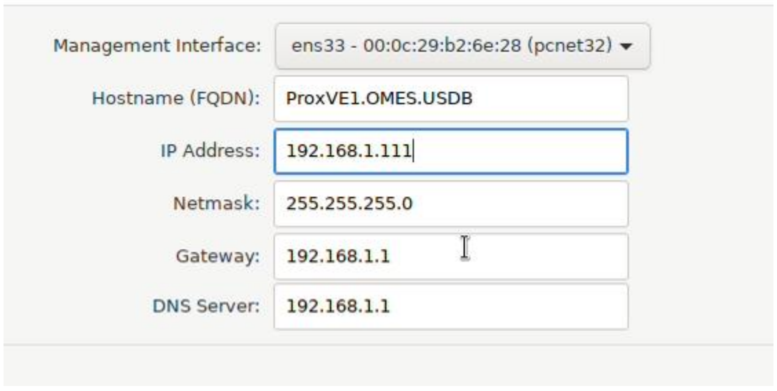
After downloading Proxmox we create our first node using this VM specifications:

- 2 network adapters for replications.
- 4 gigs of ram at least.
- 2 vCPU at least.
- 20GB of storage at least.

3. Proxmox installation.

Now we proceed with the installation process:

- The node names.
- The IP address to access web GUI.

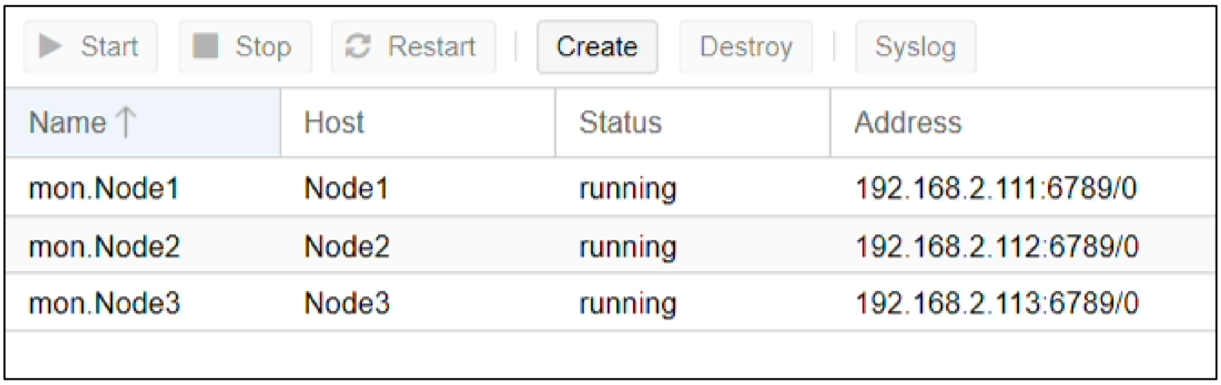


Management Interface:	ens33 - 00:0c:29:b2:6e:28 (pcnet32)
Hostname (FQDN):	ProxVE1.OMES.USDB
IP Address:	192.168.1.111
Netmask:	255.255.255.0
Gateway:	192.168.1.1
DNS Server:	192.168.1.1

Figure 0Annex1.1 Proxmox network management

4. Ceph installation.

After the creation of the cluster like shown in Chapter 3.7.1 we proceed with the setup of monitors for each node:



Name ↑	Host	Status	Address
mon.Node1	Node1	running	192.168.2.111:6789/0
mon.Node2	Node2	running	192.168.2.112:6789/0
mon.Node3	Node3	running	192.168.2.113:6789/0

Figure 0.2 *Monitor creation for each node*

Then we need to install the Ceph storage features in every node:

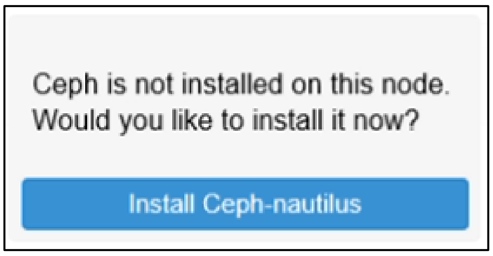
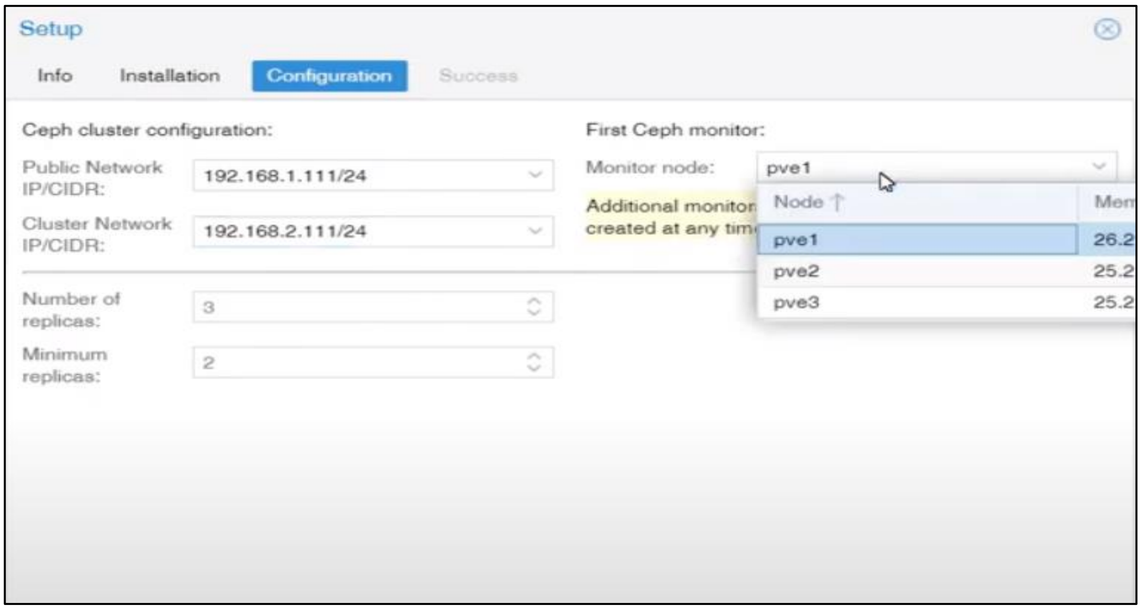

Ceph is not installed on this node.
Would you like to install it now?

Figure Annex 1.3 *Ceph installation*

We will end up with the configuration for only the parent node, the other nodes will be automatically configured after the installation:



Ceph cluster configuration:

Public Network IP/CIDR: 192.168.1.111/24

Cluster Network IP/CIDR: 192.168.2.111/24

Number of replicas: 3

Minimum replicas: 2

First Ceph monitor:

Monitor node: pve1

Additional monitor created at any tim

Node ↑	Mem
pve1	26.2
pve2	25.2
pve3	25.2

Figure Annex 1.4 *Ceph cluster configuration*

Annex 2: Floodlight installation

1. Pre-requisites

-OS: Any Linux distribution.

-JDK and ANT:

```
#sudo apt-get install java-1.8.0-openjdk java-1.8.0-devel
#sudo apt-get install ant
```

2. Floodlight download

-Download the source from Github.

-use the ANT command. (The ant command is a tool that creates the construct the application from the compilation process until the execution)

```
#git clone git://github.com/floodlight/floodlight.git

#cd floodlight

#ant
```

3. Floodlight run

-execute the file **floodlight.jar** that been created from the previous ant command:

```
#java -jar target/floodlight.jar
```

4. Access to floodlight controller

-open any web browser and access using the URL below:

```
http://floodlight_ip:8080/ui/index.html
```

Annex 3: Mininet installation

The installation and configuration of Mininet is done on a node independent of our controller, but it is possible to install them on the same machine, the configuration steps can be summarized in the following points:

- To install natively from source, first you need to get the source code:

```
#git clone git://github.com/mininet/mininet
```

- Note that the above git command will check out the latest Mininet and download it then we will need to check the version:

```
#cd mininet

#git tag # list available versions

#git checkout –b 2.2.0b3 # or whatever version you wish to install
```

- Now we install Mininet using the following command:

```
#mininet/util/install.sh -a
```

References

[1] "NFS: Network File System Protocol Specification," [Online]. Available: https://tools.ietf.org/html/rfc1094. [Accessed 29 07 2020].

[2] M. Rouse, "Bit Rot," [Online]. Available: https://searchstorage.techtarget.com/definition/bit-rot#:~:text=Bit%20rot%20is%20the%20slow,data%20decay%20and%20silent%20corruption..

[3] "Data Replication," [Online]. Available: https://searchdisasterrecovery.techtarget.com/definition/data-replication#:~:text=Data%20replication%20is%20the%20process,the%20event%20of%20a%20disaster.. [Accessed 29 07 2020].

[4] "Snapshot," [Online]. Available: https://searchstorage.techtarget.com/definition/point-in-time-snapshot-PIT-snapshot. [Accessed 29 07 2020].

[5] "Block Storage," [Online]. Available: https://searchstorage.techtarget.com/definition/block-storage. [Accessed 29 07 2020].

[6] "Object Storage," [Online]. Available: https://searchstorage.techtarget.com/definition/object-storage. [Accessed 29 07 2020].

[7] "Ceph Storage Documentation," [Online]. Available: https://docs.ceph.com/docs/master/. [Accessed 28 07 2020].

[8] S. D. Lowe, in *Hyperconverged infrastructure for dummies*, USA, HP, 2019, p. 12.

[9] J. G. a. D. D. Scott D. Lowe, Building a Modern Data Center Principles and Strategies of Design, USA: Atlantis Computing, 2016.

[10] "What Is Virtualization," [Online]. Available: https://opensource.com/resources/virtualization. [Accessed 28 06 2020].

[11] "What is Software-Defined Compute? Definition, Benefits, and Infrastructure," [Online]. Available: https://www.sdxcentral.com/networking/sdn/definitions/what-is-software-defined-compute/. [Accessed 30 07 2020].

[12] "What is a hypervisor?," [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor. [Accessed 15 07 2020].

[13] "Legacy systems & digital transformation, a love-hate relationship," [Online]. Available: https://www.tremend.com/legacy-systems-digital-transformation-love-hate-relationship#:~:text=have%20to%20reinvent%20their%20entire%20digital%20infrastructure%20to%20take%20advantage. [Accessed 02 08 2020].

[14] A. Kingatua, "Hyper-converged Infrastructure Guide," 8 04 2018. [Online]. Available: https://hostadvice.com/hosting-guides/hyper-converged-infrastructure-guide/. [Accessed 22 07 2020].

[15] "Software-Defined Storage," [Online]. Available: https://www.datacore.com/software-defined-storage/. [Accessed 25 07 2020].

[16] C. Harvey, "Software defined storage combines storage virtualization with advanced storage management capabilities to offer enterprises benefits like lower costs and greater flexibility.," 22 02 2018. [Online]. Available: https://www.enterprisestorageforum.com/storage-software/what-is-software-defined-storage.html. [Accessed 25 07 2020].

[17] "What is software-defined storage?," [Online]. Available: https://www.redhat.com/en/topics/data-storage/software-defined-storage. [Accessed 25 07 2020].

[18] R. Turk, "Ceph Intro & Architectual Overview," 22 05 2013. [Online]. Available: https://www.slideshare.net/buildacloud/ceph-intro-and-architectural-overview-by-ross-turk. [Accessed 25 07 2020].

[19] M. Rouse, "Direct Attached Storage," [Online]. Available: https://searchstorage.techtarget.com/definition/direct-attached-storage. [Accessed 25 07 2020].

[20] M. Rouse, "Network Attached Storage," [Online]. Available: https://searchstorage.techtarget.com/definition/network-attached-storage. [Accessed 25 07 2020].

[21] M. Rouse, "Storage Area Network (SAN)," [Online]. Available: https://searchstorage.techtarget.com/definition/storage-area-network-SAN. [Accessed 25 07 2020].

[22] "What Is Software-Defined Storage NetApp?," [Online]. Available: https://www.netapp.com/us/info/what-is-software-defined-storage.aspx. [Accessed 28 07 2020].

[23] "Software Defined Storage," [Online]. Available: https://www.datacore.com/software-defined-storage/. [Accessed 27 07 2020].

[24] "Introducing Gluster File System," [Online]. Available: https://staged-gluster-docs.readthedocs.io/en/release3.7.0beta1/Administrator%20Guide/GlusterFS%20Introduction/. [Accessed 29 07 2020].

[25] "About GlusterFS," [Online]. Available: https://rajeshjoseph.gitbooks.io/test-guide/content/?fbclid=IwAR0EvPJY_QmYjeVM8iHg1k9_OdvMS0iO430LL9S84f7uYqBhF7WAtdBAbCE. [Accessed 29 08 2020].

[26] "Ceph, la technologie qui "disrupte" le marché du stockage," [Online]. Available: https://www.journaldunet.com/solutions/cloud-computing/1179361-ceph-la-technologie-qui-disrupte-le-marche-du-stockage/. [Accessed 28 07 2020].

[27] "GlusterFS vs. Ceph: a comparison of two storage systems," 28 07 2020. [Online]. Available: https://www.ionos.com/digitalguide/server/know-how/glusterfs-vs-ceph/. [Accessed 27 08 2020].

[28] ""Software-Defined" for Dummies," [Online]. Available: https://ssh.guru/software-defined-for-dummies/. [Accessed 05 08 2020].

[29] F. Benamrane, Etude des Performance des Architectures du Plan de Contrôle des Réseaux Software-Defined network, Rabat: PhD thesis, Mohammed V University, January 18, 2017.

[30] M. Said Seddiki, Allocation dynamique des ressources et gestion de la qualité de service, 14 december 2015.

[31] A. Anwar, "SDN (Software-Defined Networking) Primer, and Why we Need SDN," 30 10 2015. [Online]. Available: https://blog.turbonomic.com/blog/on-technology/sdn-software-defined-networking-primer-and-why-we-need-sdn. [Accessed 27 08 2020].

[32] R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: An Overview," Jan, 1994.

[33] M. Altufaili, "Intelligent Network Bandwidth Allocation using SDN," *International Journal of Engineering Research & Technology,* vol. 4, 2015.

[34] A. Malik, A. Benjamin and K. Chih-heng, "THRIFTY: Towards High Reduction In Flow Table memory," in *2018 Imperial College Computing Student Workshop (ICCSW 2018) Volume: 66*, London, UK , september 2018.

[35] "The Road to SDN- ACM Queue," [Online]. Available: https://queue.acm.org/detail.cfm?id=2560327. [Accessed 28 June 2020].

[36] L. Genge and L. Wen, "A novel industrial control architecture based on Software-Defined Network," 2018.

[37] M. Yu, J. Rexford, M. J. Freedman and J. Wang, "Scalable Flow-based Networking with DIFANE," in *the ACM SIGCOMM 2010 Conference*, New York City, 2010.

[38] Y. Ganjali and A. Tootoonchian, "HyperFlow: A Distributed Control Plane for OpenFlow," INW/WREN, 2010.

[39] T. Eugene and Z. Cai, "Maestro: Achieving scalability and coordination in centralized network control plane," 2012.

[40] B. Rimal, A. Jukan, D. Katsaros and Y. Goeleven, "Architectural requirements for cloud computing systems: an enterprise cloud approach," *journal of Grid Computing,* pp. 3-26, 2011.

[41] M. Armbrust, A. Fox, R. Griffith and Al, "A view of Cloud Computing," Communication of the ACM, 2010.

[42] S. Scott-Hayward, G. O'Callaghan and S. Sezer, "SDN Security; a survey, IEEE SDN for Future Networks and Services," SDN4FNS, 2013, pp. 1-7.

[43] R. Kloeti, "OpenFlow: a security analysis," April 2013. [Online]. Available: ftp://yosemite.ee.ethz.ch/pub/students/2012-HS/MA-2012-20signed.pdf. [Accessed 05 July 2020].

[44] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov and R. Smeliansky, "Advanced Study of SDN/OpenFlow controllers," in *9th central and eastern European Software Engineering Conference*, Russia, 2013.

[45] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll and P. TranGia, "Modeling and Performance evaluation of an OpenFlow architecture.," in *Teletraffic Congress (ITC),* Sept 2011.

[46] "Cbench Controller Benchmarker | BibSonomy," [Online]. Available: https://www.bibsonomy.org/bibtex/111bc70e4d8012f1196123dbc3e310b7. [Accessed 06 July 2020].

[47] M. Jarschel, F. Lehrieder, Z. Magyari and R. Pries, "A flexible OpenFlow controller benchmark," in *Proc. EWSDN*, 2012, pp. 48-53.

[48] D. Tipper, "Resilient network design: challenges and future directions.," *Telecommunication Systems,* vol. 1, no. 56, pp. 5-16, 2014.

[49] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *J. Netw. Comput. Appl*, vol. 67, pp. 1-25, may 2016.

[50] D. B. Hoang, "Software Defined Networking ? Shaping up for the next disruptive step?," in *Telecommunications Association*, 2015.

[51] "An SDN Perspective to Mitigate the Energy Consumption of Core Networks – GEANT2," in *International Seeds Conference 2017*.

[52] "OpenFlow-mininet," [Online]. Available: https://sites.google.com/site/mininetbasic/open-flow/openflow. [Accessed 06 July 2020].

[53] B. Wolfgang and M. Michael, "Software-Defined Networking Using OpenFlow: Protocols,Applications and Architectural Design Choices," in *future internet*, 12 May 2014.

[54] B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol," [Online]. Available: https://tools.ietf.org/html/rfc7047. [Accessed 08 July 2020].

[55] "SDN architecture - PDF," [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf.

[56] V. K. Gubrani, M. Schraf, T. V. Lakshman, V. Hilt and E. Marocco, "Abstracting network state in Software Defined Networks (SDN) for rendezvous services," in *IEEE International Conference on Communications (ICC) ,p.6627-6632*, 2012.

[57] O. Salman, H. E. Imad, A. Kayssi and A. Chehab, "SDN controllers: A comparative study," in *Computer Science 2016 18th Mediterranean Electrotechnical Conference (MELECON)*, 2016.