

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdès



Institute of Electrical and Electronic Engineering
Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of

MASTER

In Electronics

Option: **Computer, Telecommunication Engineering**

Title:

**Design and Implementation of an SDN
Network and an API.**

Presented By:

- **KHOUDI Rania**
- **BENZAOUI Ilhem**

Supervisor:

Mr. A. MOHAMMED SAHNOUN

Registration Number:...../2019

Abstract

Traditional networks may be complex and very hard to manage. It is both difficult to configure the network according to predefined policies, and to reconfigure it to respond to faults, load, and changes.

Recently, software-defined networking (SDN) is one of the most promising solutions for today's networking. SDN is characterized by its dynamic programmability on forwarding devices through open southbound interfaces, the decoupling of the control plane and the data plane, and the global view of the network by logical centralization of the controller.

This project, starts by introducing the SDN approach and Explains its main concepts and how it differs from traditional Networking.

Next, it presents the key building blocks of a layered SDN architecture. Then, an in-depth analysis of the control plane, southbound and northbound application programming interfaces (APIs), and the application plane is performed.

One of the SDN approaches is tested and verified on a network implemented by traditional network devices. Finally, this project is concluded with some suggested studies about SDN approaches.

Keywords: SDN, Controller, APIC-EM, APIs, REST.



Dedication

This work is dedicated to my dear father, for his patience and understanding, his long years of immense sacrifices to help me move forward in life and the constant support coming from him.

My eternal gratitude to my dear Mother, who contributed in my success, through her love, support and encouragement, also for her precious advices, and for all the sacrifices she made.

To my dear sisters who have always been there for me ...

To my dearest friend.

To my partner and all her family.

To my supervisor Mr. MOHAMMED SAHNOUN and all the teachers and workers of IGEE

To all IGEE students, especially the promotion of 2014

To all those whom I love and who love me.

Rania





Dedication

To my dear parents who have always supported me and oriented me towards the good. For their help to follow my studies, their encouragement, their prayers and sacrifices, without forgetting my sisters and brothers to whom I wish a good success in their studies and work.

To my family.

To my partner and all her family.

To my supervisor Mr. MOHAMMED SAHNOUN and all the teachers and workers of IGEE.

To all students of IGEE, in particular, the promotion of 2014.

To all those who helped me throughout my university life, without forgetting my dear friends.

To all those who are dear to me.

Ilhem



Acknowledgements

First of all, we thank Allah Almighty who gave us the strength and the courage to achieve this modest work.

*We would like to express our thanks to our supervisor **Mr. A. MOHAMMED SAHNOUN** who agreed to supervise our work, his orientations and advices at all the time he has spent with us. Also, we thank him for his support, patience and his precious help.*

*Our gratitude goes also to **Mr. BENCHLAABAN** for his help, and his support, throughout our internship. We would like to express our thanks to him for benefiting us with his experience and his skills and putting at our disposal all the necessary information and hardware we needed.*

*Our thanks also go to **BENZAOUI Romeïssa** for her valuable support and her precious help.*

We also thank the members of the jury for agreeing to judge and to evaluate our work and for sharing their valuable remarks with us.

Finally, we would like to express our thanks to all the teachers, and friends of IGTE for their support and encouragement.

Table of contents

Abstract	i
Dedication	ii
Acknowledgements	iv
Table of contents	v
List of tables	viii
List of figures	ix
List of abbreviations and acronyms	xi
Introduction	1

CHAPTER I. Introduction to SDN

Introduction.....	4
I.1 Main concepts and elements of modern networking	4
I.1.1 OSI model	4
I.1.2 Router.....	5
I.1.3 Switch.....	5
I.1.4 Ethernet Cabling.....	6
I.1.5 Server	6
I.2 Drawbacks of traditional network architecture	7
I.3 New network requirements	8
I.4 SDN architecture	9
I.4.1 SDN Architecture Principles.....	9
I.4.2 SDN Approaches.....	9
I.4.2.1 Switch-based SDN (The OpenFlow model)	10
I.4.2.2 Hybrid SDN	10
I.4.3 SDN Architectural components	10
I.4.3.1 Data plane	11
I.4.3.2 Control plane.....	11
I.4.3.3 Application plane	11

I.4.4 SDN Operation.....	11
Conclusion	12

CHAPTER II. SDN Control plane

Introduction.....	13
II.1 Southbound interface	13
II.2 SDN Controller	14
II.3 SDN Control Plane functionality	14
II.4 SDN controller models	15
II.5 SDN controller platforms	16
II.6 Application Policy Infrastructure Controller Enterprise Module (APIC-EM)	18
II.6.1 Overview	18
II.6.2 Setup process	19
II.6.3 APIC-EM main functions	19
II.6.4 APIC-EM benefits	20
II.7 Limitations of centralized SDN controller	20
Conclusion	21

CHAPTER III. SDN Application plane

Introduction.....	22
III.1 Application plane	22
III.2 Types of SDN applications	22
III.2.1 Network Security	23
III.2.2 Network Maintenance	23
III.2.3 Traffic Engineering	24
III.2.4 Measurement and Monitoring	24
III.3 APIC-EM applications.....	24
III.3.1 IWAN	24
III.3.2 Network Plug and Play.....	25
III.3.3 Easy Quality of Service (EQoS)	26
III.3.4 Path Trace.....	26
III.4 Northbound interface	27

III.4.1 REST API.....	28
III.4.1.1 REST API's architectural principles	28
III.4.1.2 Benefits of REST as SDN Northbound API.....	30
III.5 APIs	30
III.5.1 Types of APIs.....	30
III.5.2 Web service APIs' Terminology.....	31
III.5.3 Planning an API	31
III.6 Building a REST API Web-based application.....	31
III.6.1 Steps for building a REST API	32
III.6.2 Web-Graphical user interface (GUI).....	32
Conclusion	33

CHAPTER IV. Implementation

Introduction.....	34
IV.1 Installing and configuring VMware ESXI on the server	34
IV.1.1 Installation.....	34
IV.1.2 Configuring VMWare ESXi	36
IV.2 Creating a virtual machine.....	38
IV.3 Installing APIC-EM.....	39
IV.4 Device configurations.....	42
IV.1.3 Basic configuration	43
IV.5 Using APIC-EM	47
IV.1.4 Discovery	48
IV.1.5 Inventory	50
IV.1.6 TOPOLOGY	50
IV.1.7 Path Trace	51
IV.6 API Interfaces	52
IV.1.8 Requirements	52
IV.1.9 Steps of building API interfaces	52
IV.1.10 RESULTS	54
Conclusion	59
Conclusion -----	60
References -----	61

List of tables

Table I.1. SDN vs Tradional networking	12
Table.IV.1. Addressing Table.....	43

List of figures

FIGURE I.1. OSI model.....	4
FIGURE I.2. ONF architecture of SDN.	9
FIGURE I.3. SDN Architecture.	10
FIGURE II.1. SDN Controller southbound APIs.....	14
FIGURE II.2. SDN Control Plane Functions and Interfaces.	15
FIGURE II.3. SDN Controller models.....	16
FIGURE II.4. APIC-EM architecture.	18
FIGURE III.1. SDN applications.....	22
FIGURE III.2. Intelligent WAN solution.	25
FIGURE III.3. Path Trace navigation pane.....	27
FIGURE III.4. Northbound interface.....	28
FIGURE III.5. Web API mechanism.	32
FIGURE IV.1. ESXI Standard Boot Menu.	34
FIGURE IV.2. End user license agreement.....	35
FIGURE IV.3. Welcome screen.	35
FIGURE IV.4. Select a disk to install or upgrade window.	35
FIGURE IV.5. Keyboard layout.....	35
FIGURE IV.6. Root password window.....	35
FIGURE IV.7. Installation complete window.....	36
FIGURE IV.8. Confirm install window.	36
FIGURE IV.9. Customization selection window.	36
FIGURE IV.10. Authentication required window.	36
FIGURE IV.11. IPv4 configuration.	37
FIGURE IV.12. Configuration management Network window.....	37
FIGURE IV.13. DNS configuration.....	37
FIGURE IV.14. vSphere Web client.	38
FIGURE IV.15. Network adapter configuration.	39
FIGURE IV.16. Linux user setting.....	40
FIGURE IV.17. NTP Server setting.....	40
FIGURE IV.18. APIC-EM Admin user setting.....	40
FIGURE IV.19. APIC-EM virtual machine.	41
FIGURE IV.20. APIC-EM services status.	41

FIGURE IV.21. APIC-EM Login window.....	42
FIGURE IV.22. Project network topology.....	42
FIGURE IV.23. Trunk link configuration.....	44
FIGURE IV.24. Status of IPv4 interfaces.	44
FIGURE IV.25. Routing table of router Algiers.	44
FIGURE IV.26. SSH configuration commands.	45
FIGURE IV.27. SNMP configuration commands.....	45
FIGURE IV.28. Routing table of Boumerdes router.....	45
FIGURE IV. 29. Routing table of Blida router.	46
FIGURE IV.30. Switch Interfaces status.	46
FIGURE IV.31. Switch interfaces mode configuration.	46
FIGURE IV.32. Switch trunk interface details.	47
FIGURE IV.33. Switch Vlan interfaces details.....	47
FIGURE IV.34. APIC-EM home page.....	47
FIGURE IV.35. APIC-EM CLI Credentials window.	48
FIGURE IV.36. APIC-EM SNMP configuration window.....	48
FIGURE IV.37. APIC-EM Discovery page.	49
FIGURE IV.38. APIC-EM Discovery process results.	49
FIGURE IV.39. APIC-EM Inventory page.....	50
FIGURE IV.40. APIC-EM Topology page.....	50
FIGURE IV.41. APIC-EM Path trace navigation-pane.	51
FIGURE IV.42. Path-trace results page.	51
FIGURE IV.43. Workflow of. API interfaces program.	53
FIGURE IV.44. Sandbox APIC-EM network topology.....	54
FIGURE IV.45. API Interfaces welcoming page.....	55
FIGURE IV.46. API Interfaces devices list.	56
FIGURE IV.47. Interfaces information page of ‘CAMPUS-Router1’.....	57
FIGURE IV.48. Page of next interfaces information of ‘CAMPUS-Router1’.	58
FIGURE IV.49. Devices menu to reselect a device.	58

List of abbreviations and acronyms

A

API Application
programmable Interface

APIC-EM Application Policy
Infrastructure Controller Enterprise
Module

AVC Application Visibility and
Control

ACL Access Control List

B

BIOS Basic Input Output
System

BGP Border Gateway Protocol

C

CLI Command Line Interface

CPU Central Processing Unit

Cisco Computer Information
System COmpany Business

CRUD Create, Read, Update, and
Delete

CDP Cisco Discovery Protocol

D

DoS Disk Operating **System**

DMVPN Dynamic Multipoint VPN

DNS Domain Name Server

E

ESXi Elastic Sky X Integrated

EQoS Easy Quality of Service

F

Fa FAst ethernet

G

GPS Global Positioning
System

GUI Graphical User Interface

H

HTTP Hypertext Transfer
Protocol

I

IP Internet Protocol

ID IDentifier

ICMP Internet Control
Message Protocol

I2RS Interface To Routing
Systems

IBM International Business
Machines

IWAN Intelligent Wide Area
Network

IPsec Internet Protocol
Security

IDE Integrated
Development Environment

ISO International
Organization for Standardization



JSON JavaScript Object
Notation



LXC Linux Containers

LISP Locator Identifier
Separation Protocol



MAC Media Access Control

MPLS Multiprotocol Label
Switching



NAT Network Address
Translation

NOS Network Operating
System

NTP Network Time Protocol



OSI Open Systems
Interconnection

OS Operating System

ONF Open Networking
Foundation

OF OpenFlow

OvSDB Open vSwitch Data Base

ONOS Open Network Operating
System

Onix Online Information
Exchange.



POST Power On Self Test

PaaS Platform As A Service

PfR Performance Routing



QoS Quality of Service



RAID Redundant Array of
Independent Disks.

RAM Random Access Memory

REST Representational State
Transfer

RPC Remote Procedure Call



SMF Single Mode Fiber Multi
Mode Fiber

SDN Software Defined
Networking

STP Shielded Twisted-Pair

SNMP Simple Network
Management Protocol

SSH Secure Shell

SDMN Software Defined
Mobile Network definition

SOAP Simple Object Access
Protocol

T

TCP Transmission Control
Protocol

U

UTP Unshielded Twisted-
Pair

UDP User Datagram
Protocol

URL Uniform Resource
Locator

USB Universal Serial Bus

V

VLAN Virtual Local Area
Network

vCPU Virtual CPU

W

WAAS Wide Area Application
Services

WAN Wide Area Network

X

XML Extensible Markup
Language

Y

YAML Yet Another Markup
Language.

Introduction

Networks, like all evolving systems, incorporate much of the structure and concepts of previous network systems. As the first revolution from circuit-switched mode to packet-switched mode, following by the second from hardwired to wireless mode to the third which this project examines, from hardware to software mode.

In a traditional network, control and forwarding data is exercised in data plane and control plane that are implemented both in each network node either routers or switches, this approach is relatively inflexible and requires all the network nodes to implement the same protocols.

Currently, With the advent of network virtualization, networking is experiencing its third major wave of revolution by applying the virtualization technology to network devices, that allowed to introduce new concepts to simplify network management and bring innovation through network programmability.

One of these adopted concepts is Software-Defined Networking (SDN), which became the focus of the main networking research topic, so as to eliminate the network infrastructure maintenance processes and guarantee easy resources, control and change on it via open interfaces. With SDN, the two planes are decoupled in a way a central

controller (SDN controller) performs all complex functionality of the control plane, including routing, naming, policy declaration and security checks using standardized or open, one or more application program interfaces (APIs) at different bounds, to accomplish the work of controlling and managing the network or invoke analytic or reporting services residing on the controller. In the other hand, only data plane resides in network hardware devices, which no longer need to understand and process thousand of protocol standards and merely accepts instructions from the SDN controller.

The theme was proposed by ICOSNET company, based on the fact that SDN is a new technology underdevelopment that is getting place in Algeria today.

In this project, we are looking to find a complete and clear answer to the following question:

« What are the key requirements for an SDN architecture and how a new API code interacts with an SDN controller? »

This project consists of four chapters followed by a general conclusion. The first chapter Provides an overview of modern networking and the elements that form the network infrastructure, issues concerning the old traditional network, and it continues by laying out what the SDN approach is, why it is needed and providing an overview of the SDN architecture. At the end, the chapter concludes by a small comparison between SDN network and the traditional one.

The second chapter is dedicated to general information about the controller part of SDN architecture and southbound interface that allow the controller to communicate with the infrastructure layer. It also includes a demonstration of APIC-EM, an implementation of the control plane.

The third chapter talks about programs and applications that represent, form the SDN application plane and interact with the SDN controller. The chapter also explains how they communicate their network requirements and desired network behavior to the SDN controller via north bound API's.

In the fourth chapter, the SDN network constructed in this project is demonstrated, as well as its basic installations and configurations. Then, 'API Interfaces' the built API in this project is tested by extracting device interfaces information from the controller.

Finally, a conclusion that answers the question posed in the beginning of the project, and proposes other functions of SDN as new projects.

CHAPTER I.

Introduction to SDN

Introduction

Software Defined Networking (SDN) is an emerging technology trend, which came after the need of abstraction and redefinition of network constructs in order to manage better, and hold the promise to revolutionize communication networking [1].

This chapter provides an overview of software defined networking and introduces its architecture, as well as its concepts and approaches, beginning with a brief survey of the key elements of modern networking and stating the performed modifications on it to emerge the new SDN architecture. Finally, the chapter is summarized by a small comparison between the new SDN network and the traditional one.

I.1 Main concepts and elements of modern networking

I.1.1 OSI model

It is a conceptual model which defines a networking framework to implement protocols in seven layers. It is a reference to understand how a network operates. The seven abstraction layers of the OSI model that are shown in figure I.1, can be defined as follows, from top to bottom [2]:

- **Application Layer (layer 7):**
The closest to the end user in which the application and user communicates, also, it is responsible for presenting

meaningful data to the user. HTTP is an example of an application layer protocol.

- **Presentation Layer (layer 6):** Responsible for translating incoming data into an understandable syntax. It serves also for encrypting, and compressing data.
- **The session Layer (layer 5):** This layer is responsible for establishment and termination of connections between devices.

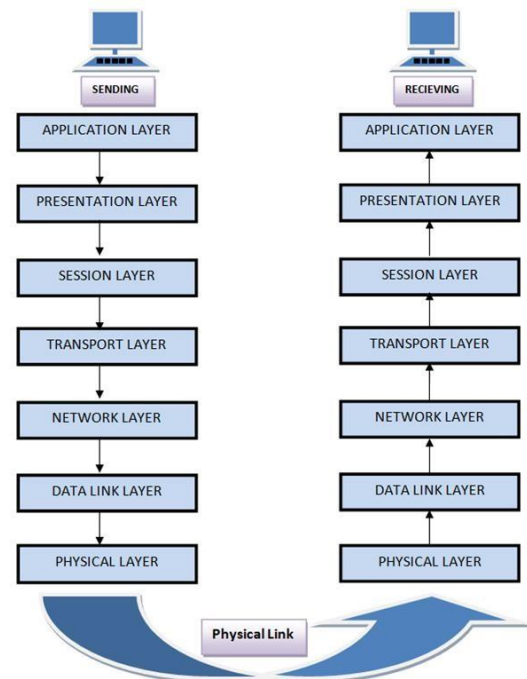


FIGURE I.1. OSI model.

- **The Transport Layer (layer 4):** It is responsible for end-to-end communication between the two devices. It prevents loss of data by handling errors and sequencing. This layer also adds source and destination port numbers.
- **The Network Layer (layer 3):** It Provides transmission of data from node to node. It has many functions like routing, forwarding, as well as addressing, internetworking, error handling, congestion control and packet sequencing. At this stage of the OSI model the source and destination IP address is added to the data.
- **The Data Link Layer (layer2):** It is similar to the network layer but with switching function, this layer facilitates data transfer between two devices on the same network.
- **The Physical Layer (layer 1):** This layer includes the physical equipment involved in the data transfer, such as the cables and switches. This is also the layer where the data gets converted into a bit stream.

I.1.2 Router

A Layer 3 network gateway device that forwards data packets along networks. Routers use headers and forwarding tables to determine the best path for forwarding the packets, and they use protocols such as ICMP to communicate with each other. They also perform many functions like Network Address Translation (NAT), Dynamic Host Configuration and act as firewalls [3].

I.1.3 Switch

Switches are multi-port intelligent devices that have a decision-making capacity in filtering and forwarding data packets between devices on a network. Since it works in the Data link layer, a switch has knowledge of the MAC addresses of the ports in the network. Switches are the foundation of most business networks. They act as a controller, connecting computers, printers and servers to a network in a building or a campus [4].

Each port on a switch can be configured as either an access or a trunk port. The type of a port specifies how the switch determines the incoming frame's VLAN:

- **Access port:** A port that can be assigned to a single VLAN and do not provide any identifying marks on the frames that are passed between switches. Also, it carries traffic that comes from only the VLAN assigned to the port.

- **Trunk port:** Trunk links are required to pass VLAN information between switches or switch and router. This port type can carry traffic of multiple VLANs, in which frames are tagged by assigning a VLAN ID to each frame as they traverse between those devices.

After configuring routers and switches, show commands displays the running configuration on them. The most used ones are :

- **Show ip interface brief:** Displays a summary of IP address, status (shut or no shutdown), and protocol (connected to media or not) of device interfaces.
- **Show ip route:** Displays the IPv4 routing table of the device.
- **Show interfaces trunk:** Displays information on the interface-trunk.
- **Show vlan:** Displays VLAN information.

I.1.4 Ethernet Cabling

There are three cable types commonly used as transmission media:

- Coaxial.
 - twisted pair (two versions, UTP (Unshielded Twisted-Pair) and STP (Shielded Twisted-Pair)).
 - fiber-optic cabling (Single-mode fiber (SMF) or Multi-mode fiber (MMF)).
- Coaxial cabling was used for computer networks, but today it is largely replaced by twisted-pair cabling which became the most popular type of cabling in today's networks. In addition, the fiber-optic cabling usage is increasing, especially in high performance and long distances networks [5].

I.1.5 Server

A server is a computer and a program that is dedicated to manage larger network resources specially in Client-server networks. Dedicated servers may have high performance RAM, multiple faster processor that work in parallel form, several hot-plug high-capacity hard drives and it may be connected to redundant power supplies [6].

This central computer (server) acts as the storage location for files and applications shared on the network [7].

Its boot operation is started by a small software program contained in BIOS. The BIOS is responsible for the power-on self-test (POST), which is performed at startup. The POST does a quick, initial check of the major components, such as memory, disk drives, keyboard, mouse, and monitor, to be sure that a minimum working system is available. After a general check is made of the major hardware components, the BIOS turns control over to bootstrap or virtual machine [8].

A virtual machine is an operating system (OS) that imitates dedicated hardware; it is installed on server drives. The end user has the same experience on a virtual machine as they would have on dedicated hardware. Several vendors offer virtual machine software, but two main vendors dominate in the marketplace: VMware and Microsoft [9].

I.2 Drawbacks of traditional network architecture

The world of networks is changing greatly, more quickly than might have been expected few years ago. Even with the greater capacity of transmission schemes and the greater performance of network devices, traditional network architectures are increasingly inadequate in the face of the growing complexity, variability, and high volume of the imposed load, which must be handled in a sophisticated manner [10]. Traditional networking devices are composed of an embedded control plane that manages switching, routing and traffic engineering activities while the data plane forwards packet/frames based on traffic. A change in any part of the network can cause a failure of the whole.

Several limitations and drawbacks of traditional network architectures are cited bellow, these drawbacks are increasing very fast with the growth of data communication.

First, networking technology has become more complex and difficult to manage while responding for demands such as differing levels of QoS, high and changing traffic volumes and security requirements. Second, to add or move any device, managers must reconfigure multiple hardware/software entities using device-level management tools by considering topology, vendor device model, software version in configuring all routers, switches, firewalls, and updating ACLs, VLANs, quality of services (QoS), and other protocol-based mechanisms. Third, today's networks configurations are performed relatively in static way to minimize the risk of service disruption, which means, manual procedures must be used to configure each vendor's equipment on a per-application and

even per-session basis. This makes it difficult to implement and configure thousands of devices and mechanisms [11].

Finally, the lack of open interfaces has affected the ability of companies to deploy new services that rapidly respond to the needs of user demands.

I.3 New network requirements

In order to avoid huge dilemma in implementing and maintaining complex modern networks, and in order to meet business and technical requirements, an optimal and an efficient infrastructure must be built.

To eliminate or reduce the traditional network limitations and drawbacks mentioned above, a list of principle requirements and some key ideas of modern networking approach are discussed here:

- **Adaptability:** Networks must adjust and respond dynamically, based on application needs, business policy, and network conditions.
- **Automation:** Policy changes must be automatically propagated so that manual work and errors can be reduced.
- **Maintainability:** Introduction of new features and capabilities (software upgrades, patches) must be seamless with minimal disruption of operations.
- **Model management:** Network management software must allow management of the network at a model level, rather than implementing conceptual changes by reconfiguring individual network elements.
- **Mobility:** Control functionality must accommodate mobility, including mobile user devices and virtual servers.
- **Integrated security:** Network applications must integrate seamless security as a core service instead of as an add-on solution.
- **On-demand scaling:** Implementations must have the ability to scale up or scale down the network and its services to support on-demand requests [11].

I.4 SDN architecture

As demonstrated in the preceding section, traditional network architectures are inadequate to meet the demands of the growing volume and variety of traffic.

Software-Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable. In order for this new world of SDN to have a chance to be successful, it has to be standardized. This standardization was carried out by the ONF (Open Network Foundation). The architecture proposed by the ONF is shown in figure I.2 [12].

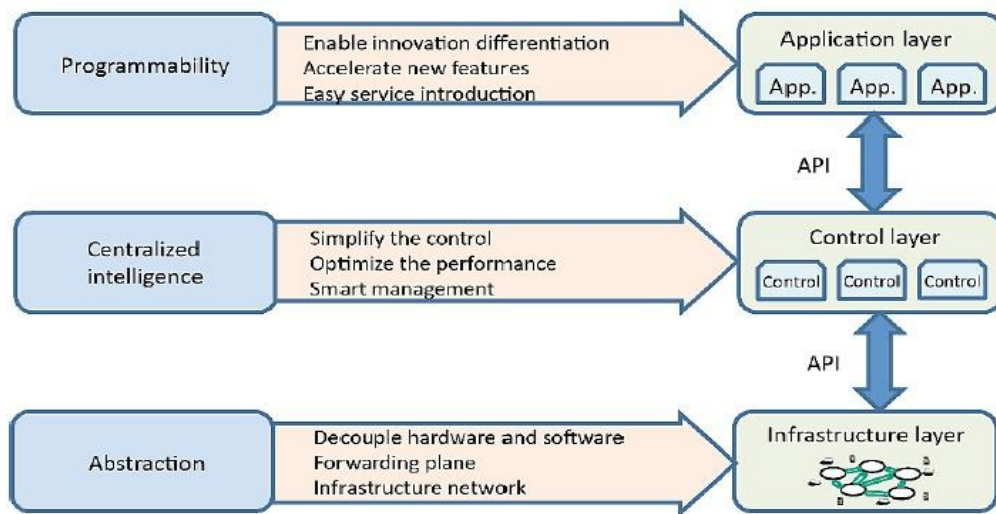


FIGURE I.2. ONF architecture of SDN.

I.4.1 SDN Architecture Principles

The basic principles of the SDN Architecture are [13]:

- ✓ Decoupling of control and data planes.
- ✓ The control plane is implemented in a centralized controller or set of coordinated centralized controllers.
- ✓ Open interfaces are defined between the devices in the control plane (controllers) and those in the data plane.
- ✓ The network is programmable by applications running on top of the SDN controllers.

I.4.2 SDN Approaches

SDN was investigated from various perspectives with the goal of identifying the approaches that can be defined, deployed, and used in the near term. The two common approaches are presented on the following sub-sections.

I.4.2.1 Switch-based SDN (The OpenFlow model)

This is on which the concept of SDN was originally built, a model that uses a mix of controllers and switches that support compatible versions of OpenFlow. OpenFlow dictates the behavior of the network switches in packet processing by programming their forwarding tables, and provides a central control point (controller) that manages network traffic handling. This model requires purchasing new switching fabric, because not all switches support OpenFlow.

I.4.2.2 Hybrid SDN

It provides flexibility by combining two or more networking technologies in a single environment, in which, a controller configures and manages specific traffic flows in a traditional network, with the help of traditional networking protocols as SNMP that manage other traffic flows on the network. This project studies this particular type of SDN.

I.4.3 SDN Architectural components

SDN architecture comprises three planes (layers) that communicate through bounds APIs as shown in figure I.3.

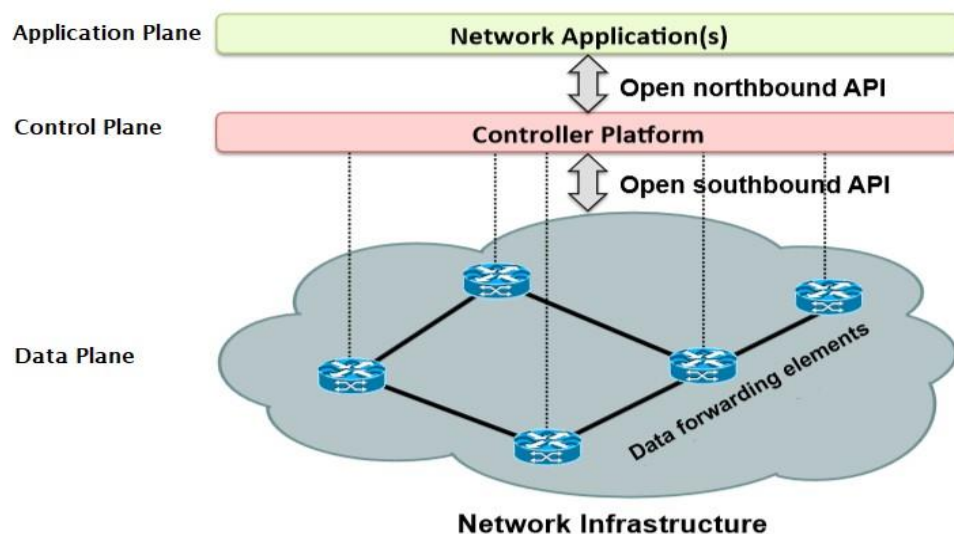


FIGURE I.3. SDN Architecture.

These layers are listed from bottom to up as follow:

I.4.3.1 Data plane

An SDN infrastructure, similarly to a traditional network, is composed of a set of networking equipment as switches and routers, which are interconnected with different transmission media to construct a network. The main difference between the two networks infrastructures resides in the fact that traditional physical devices are now simple forwarding elements, without embedded control or software to take autonomous decisions. The network intelligence is removed from the data plane devices to a logically centralized control system.

- The principal functions of the data plane are the following [14]:
 - **Control support function:** The infrastructure layer communicates with the controller and the controller manages it via southbound interfaces.
 - **Data forwarding function:** Infrastructure layer accepts incoming data flows from other network devices and end systems, and forwards them along the data forwarding paths that have been computed and established according to the rules defined by the SDN applications.

I.4.3.2 Control plane

The SDN Controller is a logically centralized entity in charge of translating the requirements from the SDN application layer down to the SDN Data plane and providing the SDN Applications with an abstract view of the network via northbound interfaces [15].

I.4.3.3 Application plane

SDN applications reside in the application plane; they are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to the SDN Controller [15].

I.4.4 SDN Operation

The first fundamental characteristic of SDN is the separation of the forwarding and control planes. Forwarding functionality, including the logic and tables for choosing how to deal with incoming packets based on characteristics such as MAC address and IP address, resides in the data plane. Protocols, logic, and algorithms that are used to program the data plane reside in the control plane.

The control plane determines how forwarding tables in the data plane should be programmed [16]. The centralized software-based controller in SDN provides an open interface on the controller to allow for automated control of the network. The terms northbound and southbound are often used to distinguish whether the interface is to the applications or to the devices. In addition to these two interfaces, there are also the eastbound and westbound interfaces. The eastbound interface enables two controllers of the same type to communicate with one another and make decisions together. The westbound interface must also facilitate communication between two controllers that belong to different sub-networks. Finally, the application plane is responsible for the applications needed by the clients and for their requirements. It sends to the controller their network requirements to translate them to infrastructure layer [17].

The main fundamental criteria which differentiate SDN from traditional networking are summarized in a small comparison shown in table I.1.

Table I.1. SDN vs Traditional networking.

Criteria	Traditional network	SDN
Network management	<ul style="list-style-type: none"> - Static and inflexible; difficult because changes are implemented separately at each device. - Works using protocols. - Distributed control plane 	<ul style="list-style-type: none"> - Dynamic; easier with the help of controller - Use APIs. - Logically centralized control plane.
Global network view	Difficult	Central view at controller
Maintenance cost	Higher	Less
Time required for update/error handling	Sometimes it takes months	Quite easy because of central controller.
Network state	Devices do not expose information and network state to the applications using them.	SDN Applications can monitor network state and adapt accordingly.

Conclusion

SDN is a concept that has many approaches, these SDN solutions are according to each organization way of work. Beside these approaches a fundamental architecture is well defined for this technology.

In the next chapter, the most important component of SDN architecture, which is the control plane, is presented in more details.

CHAPTER II.

SDN control plane

Introduction

In the SDN architecture, control and data planes are decoupled, providing open interfaces that enable the development of software that can control the connectivity and the flow of network traffic through them [18].

The control plane is the centralized intelligent logic that controls and specifies the way data traffic is managed and handled, it also provides a global view of the underlying network situation to the upper application layer [19].

This chapter focuses on the controller and the southbound interfaces of SDN network, by providing a definition of SDN controller, as well as discussing its functions and performances with mentioning the different existing controller platforms of different vendors. Next, the chapter continues by giving a detailed description of Cisco's SDN controller implementation which is "Application Policy Infrastructure Controller Enterprise Module" known as "APIC-EM". Finally, the chapter is summarized by some limitations and issues concerning SDN controller.

II.1 Southbound interface

The southbound interface as shown in figure II.1, is situated between the SDN controller and the data plane devices. It provides the logical connection between them, facilitating control over the network and enabling the controller to better adapt to changing business requirements, and to quickly respond to real-time traffic demands [20].

OpenFlow (OF), which was developed by the Open Networking Foundation (ONF), is considered one of the first SDN standards. It defines the communication protocol in southbound interface. Working on OF environment requires that any device that wants to communicate to an SDN Controller must support the OpenFlow protocol.

For traditional network devices, they work with SNMP as management southbound API to collect information about the traffic and status of the elements. Beside OpenFlow and SNMP, there exist other southbound API protocols like I2RS (Interface to Routing Systems), the Open vSwitch Data Base (OvSDB), NetConf, and LISP and BGP.

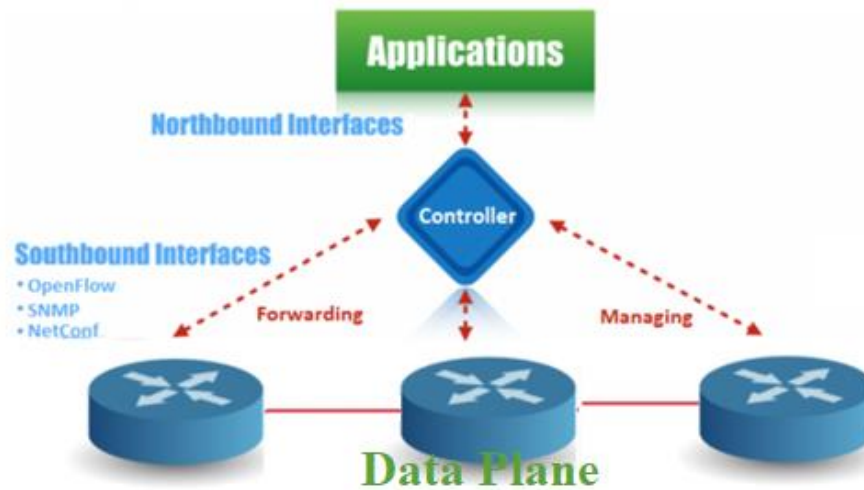


FIGURE II.1. SDN Controller southbound APIs.

II.2 SDN Controller

The controller is the most important component in the SDN architecture. It is where the complexity resides. The SDN Controller is a logically centralized entity that receives instructions or requirements from the SDN application layer through northbound APIs and relays them to the networking components through southbound APIs. It also extracts information about the network from the hardware devices and communicates back to the SDN applications with an abstract view of the network, including statistics and events about what is happening.

The controller has numerous pluggable decision-making modules that handle different functions necessary for a proper work of the network. They can perform different network tasks like discovering, inventorying network devices and gathering network statistics, etc [10].

II.3 SDN Control Plane functionality

The controller is designed to control the data plane and take instructions from the application plane in order to exert it on the networking elements. This will be achieved by means of several functions performed by the SDN controller, these functions as shown in figure II.2 [11], are essentially provided in any controller and considered to be a network operating system.

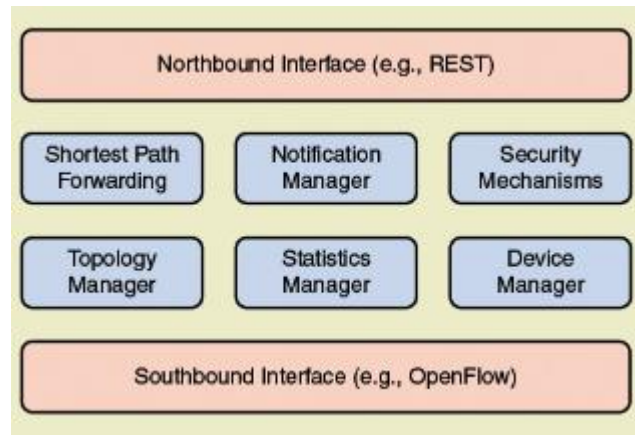


FIGURE II.2. SDN Control Plane Functions and Interfaces.

The controller functions are summarized as following:

- **Shortest path forwarding:** selecting and establishing the best route by gathering information from the networking elements.
- **Notification manager:** Receives, processes, and forwards to an application events, such as alarm notifications, security alarms, and state changes.
- **Security mechanisms:** Provides isolation and security enforcement between applications and services.
- **Topology manager:** Builds and maintains switch interconnection topology information.
- **Statistics manager:** Collects data on traffic through the switches.
- **Device manager:** Configures switch parameters and attributes and manages flow tables [11].

II.4 SDN controller models

SDN supports both centralized and distributed controller models as demonstrated in figure II.3 [21]. Each model has requirements to consider.

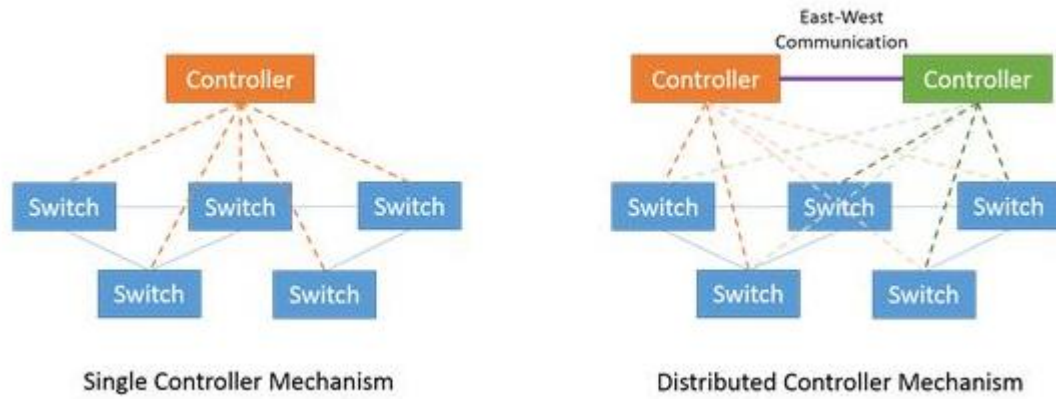


FIGURE II.3. SDN Controller models.

- **The centralized SDN model:** is based on a single centralized controller that manages and supervises the entire network. This model is supported by ONF. The network intelligence and states are logically centralized inside a single decision point [22].
- **The distributed SDN model:** aims to eliminate the single point of failure and enables scale up by sharing the load among several controllers. Distributed SDN control planes have been designed to be more responsive to handle local network events in data centers, where the controller instances share a huge amount of information to network-wide consistency. In particular, for multi-domain SDNs with a large variety of network technologies ranging from high-capacity optical fiber to bandwidth-limited wireless links, the distributed SDN architecture is easily able to adapt to the users' and applications' requirements. Moreover, a distributed controller is more responsive, robust and can react faster and efficiently to global event handling using the east and west bounds interfaces to communicate between the different controllers [23].

II.5 SDN controller platforms

In the SDN model, the controller platform is a critical pillar of the architecture, and efforts are being devoted to turn SDN controllers into high-performance, scalable, and highly available programmer-friendly software [24].

Today, there exist many implementations of SDN controller, either commercial ones or even open source, few of these initiatives are described as following:

- **OpenDaylight:** founded by Cisco and IBM, it is an open source platform for network programmability to enable SDN. It is part of the Linux Foundation. This

Controller is Java-based and derived from the original Beacon design. It supports OpenFlow and other southbound APIs and includes critical features, such as high-availability and clustering [25].

- **Open Network Operating System (ONOS):** it is an open source SDN NOS developed by a number of carriers, such as AT&T and NTT, it was initially released in 2014. ONOS is supported by the Open Networking Foundation, it is designed in order to be used as a distributed controller and to provide abstractions for partitioning and distributing network state onto multiple distributed controllers [11].
- **POX:** an open source development platform for OpenFlow SDN controllers, POX provides a framework for communicating with SDN switches, it provides a web-based graphical user interface (GUI) and is written in python [26].
- **Beacon:** a Java-based open source controller that was created in 2010, working with OpenFlow, it is the basis of Floodlight and It has been used for teaching and research purposes. Beacon was the first controller that made it possible for beginner programmers to work with and create a working SDN environment [27].
- **Floodlight:** An open source package developed by Big Switch Networks. Floodlight is a Java-based OpenFlow Controller that was built after Beacon, using a software called Apache, which made its development easier. It gives the ability to easily adapt software and develop applications and most of its functionality is exposed through a REST API [28].
- **Ryu:** it is an open source ,fully developed in python., software-defined networking (SDN) Controller developed by NTT Labs, designed to have a better management of the network and an easy traffic handling by providing well defined APIs that help to have an easy and simple creation of network control applications [25].
- **Onix:** developed by VMWare, Google, and NTT. Onix is a commercially available SDN controller [11].
- **APIC-EM:** an intelligent module and the heart of Cisco's architecture for SDN, this platform is detailed in the next section.

II.6 Application Policy Infrastructure Controller Enterprise Module (APIC-EM)

Nowadays, there is a lot of hardware that is up and running which doesn't support new protocols like OpenFlow. To start using SDN, all hardware must be replaced with new one that do support some form of SDN. In this case, cisco thought of new approach to SDN which is Cisco APIC-EM (application policy infrastructure controller enterprise module), which uses the same networking devices that are used in a traditional network [29].

II.6.1 Overview

APIC-EM is an SDN controller created for Enterprise hardware, and designed to orchestrate and manage local and wide area networks composed of cisco infrastructure. It uses open Northbound REST APIs that drive core network automation solutions. For the southbound interface, it uses common protocols like Telnet, SSH and SNMP to communicate with the hardware. APIC-EM platform supports both wired and wireless enterprise networks [30]. Figure II.4 [31] describes APIC-EM architecture.

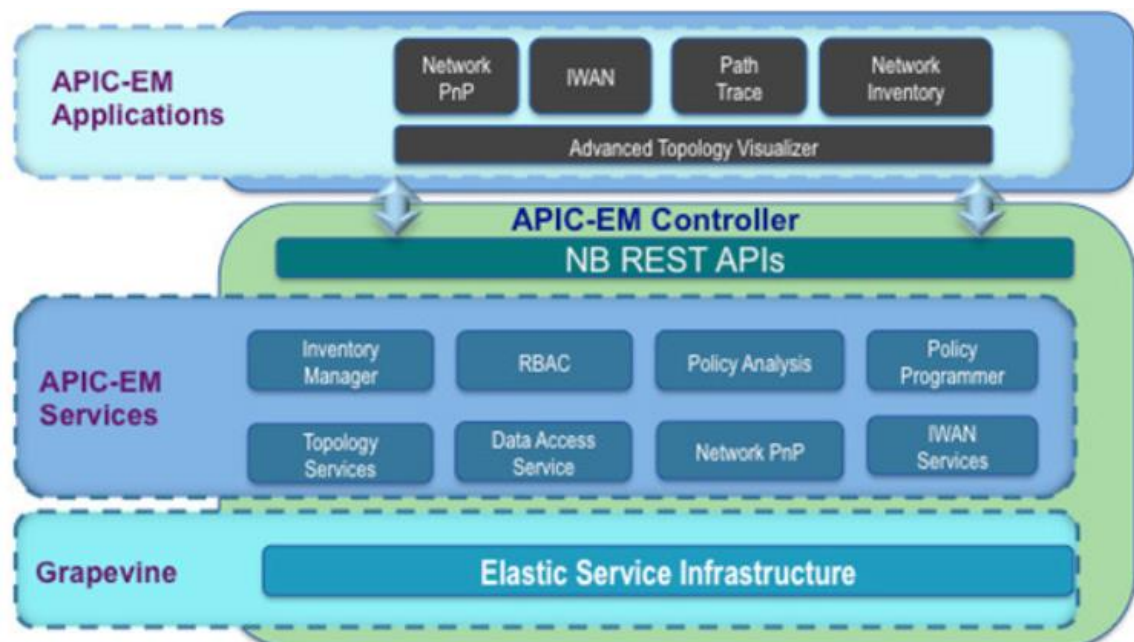


FIGURE II.4. APIC-EM architecture.

It provides built-in applications as:

- IWAN (Automated configuration. Detailed in section III.3.1).
- Network topology (Detailed in section II.6.3).
- Path trace (Analysis of end to end network path. Detailed in section III.3.4).
- Plug and play (Zero touch deployment of network devices. Detailed in section III.3.2).

II.6.2 Setup process

Cisco offers an ISO image that can be downloaded to install APIC-EM on either a physical server or virtual machine. It needs 6 CPU cores with 2.4Ghz at least, 64GB of RAM and a 500GB hard disk as hardware requirements for a production install. For evaluation, it can be installed on less hardware [32].

The APIC-EM creates a Platform as a Service (PaaS) environment for network, using Grapevine that makes use of the Ubuntu operating system environment and Linux containers (LXC) in the host (server) as an Elastic Services platform (A middleware for the execution of applications).

The Grapevine root handles all policy management, service updates, and the service lifecycle for both itself and the Grapevine client. The Grapevine clients run the different controller's services. Those services provide the controller with its core functionality. It can be used for troubleshooting case [30].

APIC-EM controller can be up to use after following a basic installation wizard and confirming that all its services are on running status through grapevine clients.

II.6.3 APIC-EM main functions

Discovery: The first step when using the controller is to discover the devices in the network. So, the Discovery function scans the devices and hosts using network information provided as IP addresses and confirming southbound interface communications processes using ICMP, SNMP, CLI credentials, telnet and SSH. it builds the APIC-EM controller database with the information that it retrieves from network.

Inventory: The Device Inventory page displays all network elements that the scan identified as non-host devices. In a table format, it displays the detailed information about

any discovered device such as device name, configuration information of the device, etc. Additionally, inventory function enables user to perform a number of actions on individual devices and groups of devices as assigning device role (core, access...), location and adding tags to devices. After the initial discovery, network devices listed in the inventory are polled every several minutes. Polling occurs for each device, link, host, and interface to confirm the reachable and managed status.

Topology: The topology function displays a graphical view of the network. Using the inventory elements and settings. It allows user to perform many actions as topology structure, changing device role and applying tags to devices. [30]

II.6.4 APIC-EM benefits

- **Collects information about the entire network:** interface statistics, routing protocols, and builds topology picture.
- **Simplified configuration and provisioning:** The controller automates the deployment of network policies across the entire network.
- **Investment protection:** The controller works with existing network infrastructure, so no infrastructure replacements and no new network hardware are needed.
- **Openness, programmability and customization:** The APIC-EM is highly programmable through REST (representational state transfer) APIs. It can enable independent software developers to create innovative network services and applications to fuel business growth. [33]

II.7 Limitations of centralized SDN controller

SDN is a logically centralized technology. Therefore, it may face some problems and challenges, considering scalability, reliability and privacy as the surrounding environment of the controller is evolving. All those challenges need more attention in order to increase network performance.

Scalability: scalability is one of the main raised concerns in SDN, because a centralized logic control represents a big challenge in wide-area networks when adopting SDN approach. This can be seen in data centers, where network traffic intensity is enormous and the number of devices an SDN controller can manage is limited, that poses a threat to the scaling capabilities of SDN.

Reliability: One of the things that software defined networks are supposed to bring to the enterprise is greater reliability, without having to manually provision new resources and configure network pathways; in a centralized controller network, if the controller fails for any reason, the overall network fails, this is known as a single point of failure.

Privacy: In order to protect domain information, and to choose to implement different privacy policies in different SDN domains, some networking information in one domain should not be disclosed to an external entity, which is impossible to make in case of a one centralized controller. [34]

Conclusion

Control plane is the brain of SDN that promised to facilitate network management through the controller whose crucial value is to provide abstractions, essential services, and common APIs to developers. APIC-EM as an SDN controller provides common functionalities as network state and network topology information, device discovery, and distribution of network configuration.

The programmable part of SDN architecture which is the application plane is presented in the next chapter.

CHAPTER III.

SDN Application plane

Introduction

While the control plane is well defined as being an essential part in the SDN architecture, questions rise on the nature and scope of the application plane that may include program abstractions and contain services that might also be viewed as part of the functionality of the control plane [11].

This chapter focuses on the application plane that resides above the control plane, its scope, and how it communicates with the control plane. Beginning with an introduction to the application layer and its architecture. Then it continues by looking at the types and categories of SDN applications. After that, it talks about the SDN northbound interface with a description of the most used one in SDN implementations, which is the REST northbound interface. Finally, it gives general information about APIs with describing the REST based application as a developable example.

III.1 Application plane

Network applications can be seen as the “network brains”. An SDN application is a software program designed to fulfill user requirements in an SDN environment. The application layer contains SDN applications and services; which access a global network view with instantaneous status, and control devices at the infrastructure layer (Defined in section I.4.3.1) through the programmable platform (northbound interfaces) provided by the control layer [35].

III.2 Types of SDN applications

SDN can be deployed on any traditional network environment, from home and enterprise networks to data centers and Internet exchange points. Such variety of environments has led to a wide array of network applications.

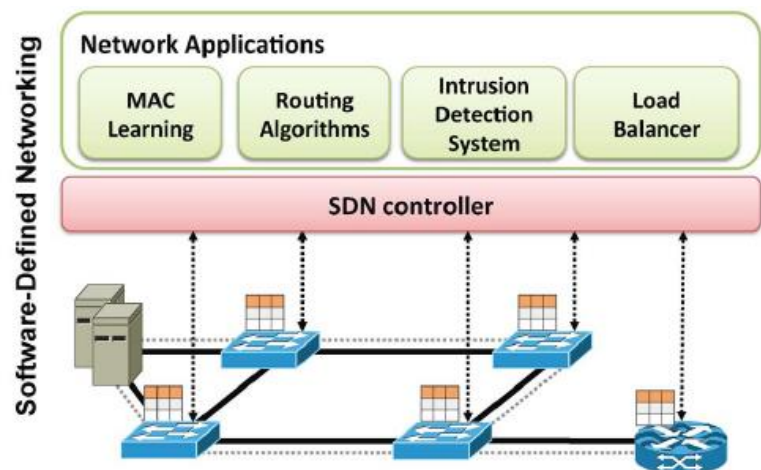


FIGURE III.1. SDN applications.

SDN network applications perform traditional functionality, as shown in figure III.1 routing, load balancing (distributing incoming network traffic across a group of backend servers) network monitoring, intrusion detection (software application that monitors a network for malicious activity or policy violation), and security policy enforcement. Moreover, they offer new services as end-to-end QoS enforcement, network virtualization and mobility management in wireless networks [36].

Despite the wide variety of use cases, most SDN applications are summarized in the following categories:

III.2.1 Network Security

Network security is a notable part of cyber security and is gaining attentions. Traditional network security practices deploy firewalls and proxy servers to protect a physical network. For SDN network security, there are essentially two approaches, one involves using SDN to improve network security, and another for improving the security of the SDN itself.

SDN offers a convenient platform to centralize, merge and check policies and configurations to make sure that the implementation meets required protection thus preventing security breaking. Moreover, SDN provides better ways to detect and react to attacks. Furthermore, the ability to collect network status and allows analysis of traffic patterns for potential security threats.

Most applications take advantage of SDN for improving services required to secure systems and networks, such as policy enforcement for access control, firewalling, DoS attacks detection and traffic anomaly detection [37].

III.2.2 Network Maintenance

Configuration errors are common causes of network failures. What makes it worse is that existing network tools that deal with individual diagnosis such as ping and traceroute, fail to provide an automated and comprehensive network maintenance solution. As a comparison, centralized and automated management and consistent policy enforcement, inherent in SDN networks, help reduce configuration errors. Moreover, with a global view and central control of configuration, SDN offers opportunities to design comprehensive network applications for automated network maintenance [38].

III.2.3 Traffic Engineering

The main goal of most traffic optimization applications is to engineer traffic with the aim of minimizing power consumption, maximizing whole network utilization, providing optimized load balancing.

Among those applications, there are flow management, fault tolerance, topology update, traffic characterization and QoS [39].

III.2.4 Measurement and Monitoring

Measurement and monitoring solutions can be divided into two classes: first, applications that provide new functionality for other networking services; and second, proposals that target to improve features of OpenFlow-based SDNs, such as to reduce control plane overload due to the collection of statistics [40].

III.3 APIC-EM applications

The following applications provide an example of how the control plane interacts with the application plane when studying their way of work.

III.3.1 IWAN

So far, MPLS and the leased line service were used in private WANs in order to have reliable connectivity with predictable performance, but they were expensive and not always cost effective.

Intelligent WAN (IWAN) is a product that serves to better collaborate and to improve cloud application performance when connecting branch network to data center, and in the same time as shown in figure.III.2, it lowers the operating cost by letting the choice to select alternative network connections, either dual MPLS or dual Internet or hybrid [41]. IWAN Provides many benefits, which are represented in Four Main Architectural Framework Components:

- 1. Secure and flexible transport-independent design:** with Dynamic Multipoint VPN (DMVPN) network security is the same as the expensive MPLS connection.
- 2. Intelligent path control :** using Performance Routing (PfR), which connects two or more WAN connections and dynamically do a path selection between them, improving WAN efficiency, and dynamically controlling data packets forwarding

decisions, taking in consideration application type, performance, policies, path status, and circuit costs.

3. **Application optimization:** Application Visibility and Control (AVC) and Wide Area Application Services (WAAS) together provide application performance visibility and optimization over the WAN:
 - **Application Visibility and Control (AVC)** provides network traffic visibility from application Layer point of view, and lets the user see the actual application name, the protocol used. It also gives visibility into the bandwidth usage when recognizing the unique signatures of different applications.
 - **Wide Area Application Services (WAAS)** serves to optimize WAN traffic, reduce WAN bandwidth needs and latency, and maximize WAN connection.
4. **Secure connectivity:** It protects the WAN and offloads user traffic directly to the Internet by using Strong IPsec encryption, zone-based firewalls, and strict access lists. With IWAN a new router can be provisioned in a matter of minutes without any knowledge of the Command Line Interface (CLI) just by its IOS image.

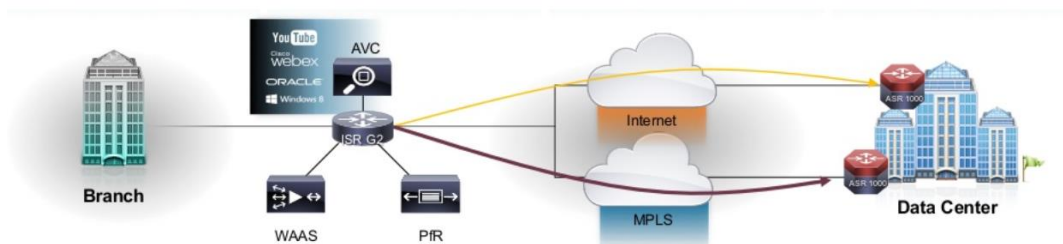


FIGURE III.2. Intelligent WAN solution.

III.3.2 Network Plug and Play

The Network Plug and Play solution reduces the burden on enterprises by simplifying deployment of new devices, by providing a unified, secure, flexible and easy approach to provision these network devices with a near zero touch deployment experience. With this new approach, the on-site non-technical staff adds new devices to the network without having to provide basic configuration, while the network administrator centrally manages device configuration through the controller [42].

III.3.3 Easy Quality of Service (EQoS)

The EasyQoS feature reflects organizations business intent by abstracting away the complexity of deploying QoS across the network. It is an application that runs on top of APIC-EM, that defines and specifies how these applications are to be marked and treated over the IP network.

Once the Discovery feature is used on the APIC-EM to discover a specific network topology, the user creates one or more policy scopes, which are collections of network devices consisting of routers and/switches that have been discovered, these devices that are displayed under Network Devices can be drag and dropped under scopes in order to be able to apply a QoS policy on them at once. Then, user specifies which applications in the network are considered “Business Relevant,” “Default,” and “Business Irrelevant”. In order to clarify which applications are considered to be important to fulfil the work of organization. EasyQoS uses the information provided, and deploys a very robust QoS configuration out to all network devices by sending appropriate sets of commands to each device, despite their different models which need different commands to accomplish the tasks. [43]

III.3.4 Path Trace

It is one of the in-built tools in APIC-EM that gives extra advantage in network troubleshooting. The controller reviews and collects network topology and routing data from discovered devices. Then it uses this data to calculate and provide a network path tracing from one host to the other, giving real time visibility into the entire path the packets take to reach out to the destination, and reverse path to see if it takes the same time to reach out to the source. Either TCP or UDP protocol is used in order to establish the path trace connection, ensuring that the controller has SSH or Telnet access to the devices.

If the controller cannot complete a path trace for the selected hosts or interfaces, it displays the results of a partial trace. In order to start a path trace, the necessary steps are summarized in figure.III.3 and more described in section IV.5.3. [44]

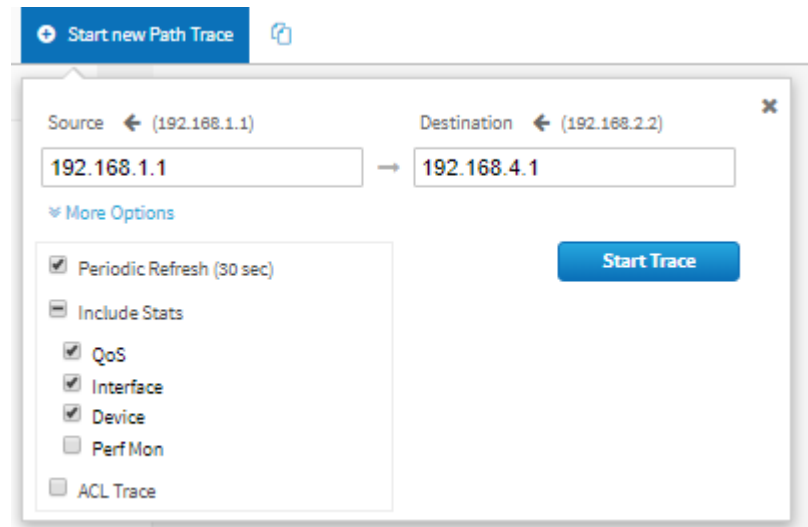


FIGURE III.3. Path Trace navigation pane.

1. In the Navigation pane, clicking 'Path Trace'. Then, 'Start new Path Trace' From the path trace toolbar.
2. In the Source field, the IP address of the host must be entered. If it was manually entered, the device from the list must be selected and then the interfaces for that device. This step will be repeated in the destination field.
3. There are optional steps in the configuration like the specification of the port number of the host where the trace will be ended in the source port field and also in the destination port field, choosing TCP or UDP from the drop-down menu in the protocol field, configuring the path trace to collect additional statistics, by checking the Stats check box and other check boxes (QoS, Interface, Device, Performance Monitory (Perf Mon)).
4. Selecting the ACL (access control list) Trace check box to run an ACL-based path trace. Then clicking 'Start Trace'.

III.4 Northbound interface

The northbound interface is used to access the SDN controller itself. This allows a network administrator, using its multiple platforms as figure.III.4 shows, to access the SDN network to configure it or to retrieve information from it. Seeing that, the value of SDN is tied to the innovative applications it can potentially support and enable, Northbound APIs are the most important components of SDN Architecture. They put applications in control of the network by providing access to controller, all that thanks to their dynamic nature [45].

As they are so critical, Northbound APIs are the most variant component in an SDN environment because they support a different type of applications. As each network applications have different requirements, a variety of Northbound API such as SDMN API, PANE API, NetCore, Pyretic, REST API have been created to implement the applications [46].

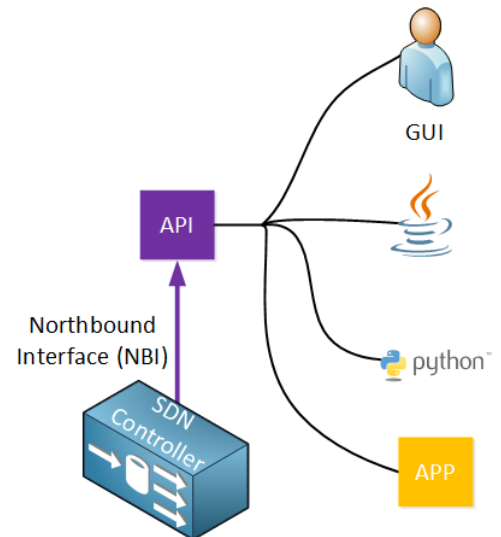


FIGURE III.4. Northbound interface.

III.4.1 REST API

In SDN architecture, Rest APIs are usually SDN northbound application program interfaces (APIs). They are used to expose data from the controller to applications.

REST (Representational State Transfer) is an architectural style, and an approach to communications that is often used in the development of networked applications and Web services. It takes advantage of existing protocols as HTTP protocol's libraries to provide CRUD (Create, Read, Update, and Delete) behavior on a data or resource (collection of data), and to make calls between machines. [47]

The CRUD behavior maps the following HTTP protocol method verbs:

- POST: Create a new, Unique thing.
- GET: Read the information about a thing or resource.
- PUT: Update the information about an existing thing.
- DELETE: Delete a thing.

One of the key advantages of REST APIs is that, they provide a great deal of flexibility. So, it can return different data formats XML, JSON, YAML or any other format depending on what the client requests. [48]

III.4.1.1 REST API's architectural principles

The following constraints are the key when considering whether a REST API is the right type of API when is needed to develop one.

- **Client-Server:** This constraint operates on the concept that the client and the server should evolve individually and independently from each other; so that making changes to application doesn't impact either the data structure or the database design on the server.
- **Stateless:** REST API rely on the data that is provided in the request to understand it, not on data being stored on the server such in case of user ID.
- **Cache:** to Reduce load on servers, Cache constraints require that the data within a response to a request labeled as cacheable or non-cacheable. This can be done by HTTP headers, means when cacheable data (have the ability to be stored as copies in several places along the request-response path), the response is indicated by expire time.

For REST API actions, GET requests is cacheable by default, POST requests are not cacheable by default, PUT and DELETE requests are not cacheable.

- **Uniform Interface:** REST APIs interface for resources must be decided inside the system. This interface should provide an unchanging, standardized means of communicating between the client and the server, such as using HTTP with URL resources, CRUD (Create, Read, Update, Delete), and JSON.
- **Layered System:** a layered system is a system comprised of layers, with each layer having a specific functionality and responsibility. REST APIs have different layers of their architecture, where APIs can be deployed on server A, and store data on server B and authenticate requests in Server C. So that, REST allows clients (browsers) and servers to interact in complex ways without the client knowing anything in advanced about the server and the resources it entertains, or to an intermediary along the way. This can be considered as security measure to prevent attacks to get the actual server architecture.
- **Code on Demand:** this constraint is optional. REST allows client functionality to be extended by downloading and executing code in the form of applets (a very small application designed to perform a specific function within another application) or scripts. [47]

III.4.1.2 Benefits of REST as SDN Northbound API

Adopting REST for the SDN Northbound API has many benefits. These benefits include:

- **Different clients:** REST separates resource representations, identification, and interaction, in case of different clients requests to optimize API performance.
- **Service composition:** REST can provide SDN by service-oriented compositions that allow distribution of data and enables multiple platforms to access, use and manage data in a flexible way.
- **Localized migration:** The functions of SDN are fast evolving and northbound APIs are likely to change accordingly. REST API supports backward compatible service migration through localized migration by which a newly added resource only affects the resources that connect to it.
- **Scalability:** REST achieves server scalability by keeping the server stateless and improves server performance through layered caches. Due to this feature, SDN controller can support a large number of concurrent host-based applications and use network resources in an efficient way. [49]

III.5 APIs

In programming, the term API, short for Application Programming Interface, refers to an intermediary part of a computer program designed to allow two applications to talk to each other, by following a set of rules describing how one application can interact with another, and the mechanisms that allow such interaction to happen.

III.5.1 Types of APIs

APIs are classified as following:

- Web service APIs
- Library-based APIs
- Class-based APIs (object oriented) – a special type of library-based API
- Functions or routines in an OS
- Object remoting APIs
- Hardware APIs

The most common APIs used are Web service APIs. They provide an interface for web applications, or applications that need to connect to each other via the Internet (The World Wide Web). There are further types of Web APIs such as Simple Object Access Protocol

(SOAP), Remote Procedure Call (RPC), and the most popular Representational State Transfer (REST). [50]

III.5.2 Web service APIs' Terminology

When using or building Web APIs, these terms are commonly used:

- **HTTP (Hypertext Transfer Protocol):** Is the primary means of communicating data on the web. HTTP implements a number of “methods,” which tell which direction data is moving and what should happen to it.
- **URL (Uniform Resource Locator):** An address for a resource on the web. It consists of a protocol (http://), domain (programminghistorian.org), and optional path (/about). A URL describes the location of a specific resource.
- **JSON (JavaScript Object Notation):** Is a text-based data storage format that is designed to be easy to read for both humans and machines. JSON is generally the most common format for returning data through an API.
- **REST (REpresentational State Transfer):** An API designed with some or all REST principles. More details are in section III.3.1.

III.5.3 Planning an API

To give an API a long life and making sure that it's used by developers, the planning process should contribute the following steps:

- ✓ Understand what the API should be able to do, and how it will work.
- ✓ List out the actions that the API must accomplish.
- ✓ choosing which type of API to build, understanding why that type is best for the application.
- ✓ designing it to work effectively.

III.6 Building a REST API Web-based application

REST API based application is an application that combines between REST API and Web-GUI. It can be developed by any programming language as python, starting with building a REST API as backend that extracts information from database. Then associate it with a suitable Web-GUI code, in order to expose data to clients [51].

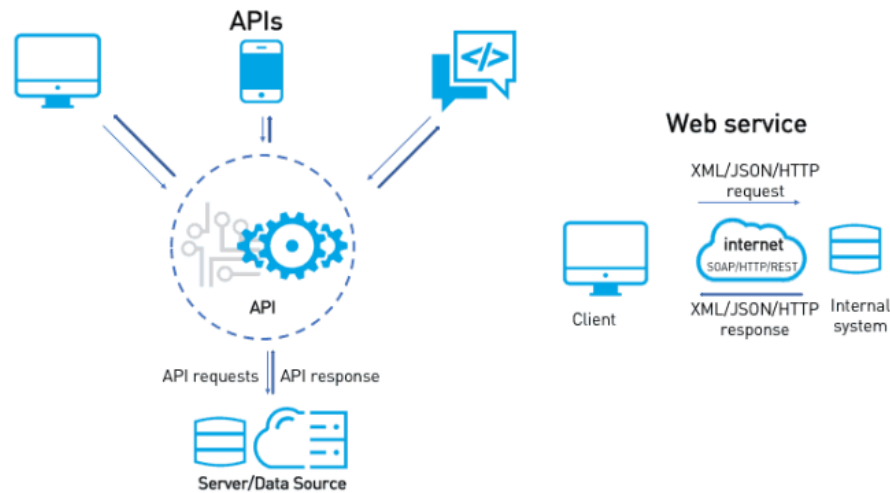


FIGURE III.5. Web API mechanism.

III.6.1 Steps for building a REST API

By applying REST principles and understanding web APIs way of working as figure.III.5 describes. The following steps show how to design a REST API [52]:

1. **Identify Object Model:** The very first step in designing a REST API based application is identifying the objects which will be presented as resource and sub-resources.
2. **Create Model URLs:** By applying uniform interface constraint, this step focuses on: define the relationship between resource and its sub-resource, how information is fetched from database, and creating a URL that identify a unique resource on the web. This resource should be accessible through a common method of HTTP protocol.
3. **Determine Representations:** When resource URLs have been decided, most representations are defined in either XML or JSON format.
4. **Assign HTTP Methods:** In this step, the possible actions in application are decided and mapped on resource URLs. So, connecting to database.

III.6.2 Web-Graphical user interface (GUI)

Web-GUI is an open-source content management system that permits unprofessional users to arrange content in pages and layouts. It is a customized template which keeps the site content and style separate, and permit clients to access, understand, use and interact with various types of data [53].

Choosing appropriate interface elements will help with task completion, efficiency, and satisfaction. These elements include:

- **Input Controls:** buttons, text fields, checkboxes, dropdown lists, date field.
- **Navigational Components:** slider, search field, pagination, slider, tags, icons.
- **Informational Components:** progress bar, notifications, message boxes.
- **Containers:** component bar.

Conclusion

APIC-EM SDN controller uses REST methods to expose its bibliography and allow new designed APIs to get data according to the provided actions.

Because the API of this project will take advantage of the database provided by APIC-EM, and will get data from it, REST API structure is the most suitable for this work.

In the next chapter, one application of SDN approach is implemented and tested on traditional network.

CHAPTER IV.

Implementation

Introduction

The fourth chapter contains the application part of this project, which is a creation and an application of one approach of SDN architecture on a traditional network, then developing an API that shows the results.

First, to test this solution, all necessary VMware ESXI 6.5 installations and configurations are performed on a utilized server in order to install the virtual server. After that, a new virtual machine is created, in which the APIC-EM image is installed. Then, the network is constructed with the necessary cablings and basic configurations on routers and switches, and APIC-EM is used to manage the network as a controller. Finally, a new REST API based application named API Interfaces is developed to extract from the controller interfaces information of all devices, and to show the results obtained from SDN solution.

IV.1 Installing and configuring VMware ESXI on the server

IV.1.1 Installation

The first step when realizing the project is to install the VMware ESXI 6.5 on HP Proliant DL380 – Xeon 3.4GHz, which is a vSphere virtualization layer that runs on physical hardware and on the DOS of the server.

To do so, these steps must be followed:

- Inserting A USB to the physical server, then booting the server.
- Once the server is booted from the ISO, as there is no OS installed on the local disk, the first option which is boot from the standard installer is selected as shown in figureIV.1. This would boot the server and would load the ESXi installer file.



FIGURE IV.1. ESXi Standard Boot Menu.

- After loading the installer files from USB is finished, pressing Enter after the welcome screen shown in figure IV.3 will appear. Then, pressing accept and continuing after reading the “End User License Agreement” (EULA) shown in figure IV.2.

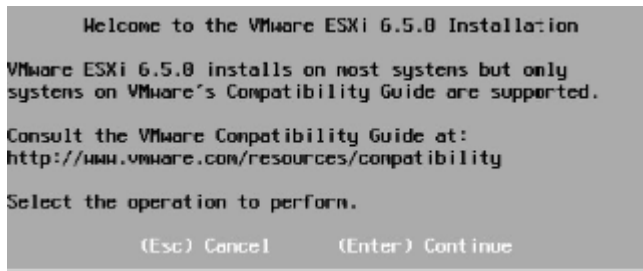


FIGURE IV.3. Welcome screen.

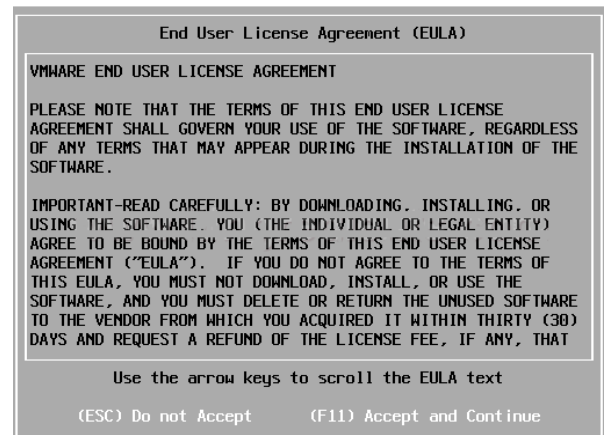


FIGURE IV.2. End user license agreement.

-After the server scans the available devices, the “select a disk to install or upgrade” window in figure IV.4 appears. It shows all the local Disks attached with the server. Choosing RAID0 (A storage technology which splits data across all the drives). Then hitting enter to continue the installation.

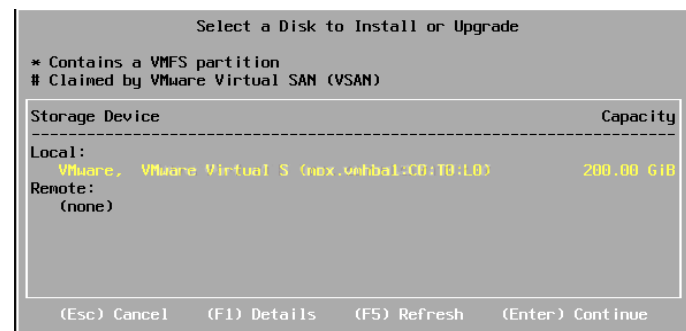


FIGURE IV.4. Select a disk to install or upgrade window.

- To install VMWare ESXi 6.5, the keyboard language must be selected as shown in figure IV.6, and the Root password must be assigned as shown in figure IV.5.



FIGURE IV.5. Keyboard layout.

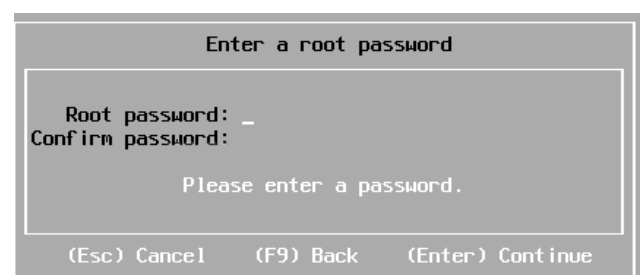


FIGURE IV.6. Root password window.

-The “Confirm install” window shown in figureIV.7 appears after few minutes, F11 should be pressed from keyboard to install **VMWare ESXi**:

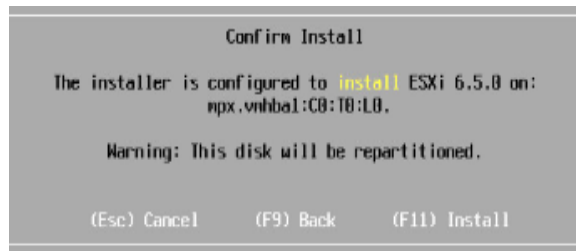


FIGURE IV.7. Confirm install window.

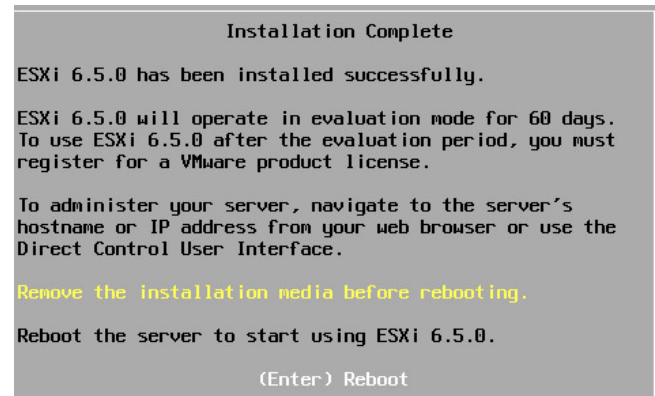


FIGURE IV.8. Installation complete window.

- The installation takes couple of minutes. Once it is completed as shown in figureIV.8, ‘enter’ button should be pressed to reboot the server.

IV.1.2 Configuring VMWare ESXi

After the installation is finished, some configurations are performed.

-Pressing F2 from keyboard for customization as shown in figureIV.9. Then entering the authorized password and login name and clicking “Enter” to start the customization:



FIGURE IV.9.Customization selection window.

-Entering the authorized password and login name on the authentication required window shown in figure IV.10, and clicking “Enter” to start the customization.

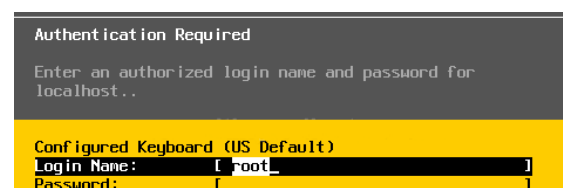


FIGURE IV.10. Authentication required window.

- On the system customization window, the option of “Configure Management Network” must be selected.

-On the “Configure Management Network” shown in figureIV.11, IPv4 Configuration option is selected, then clicking ‘enter’ to change the IP address and to make it static as shown in figureIV.12, then hitting continue.

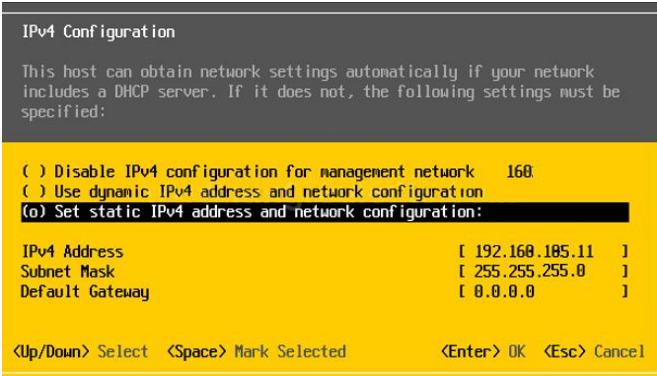


FIGURE IV.11. IPv4 configuration.

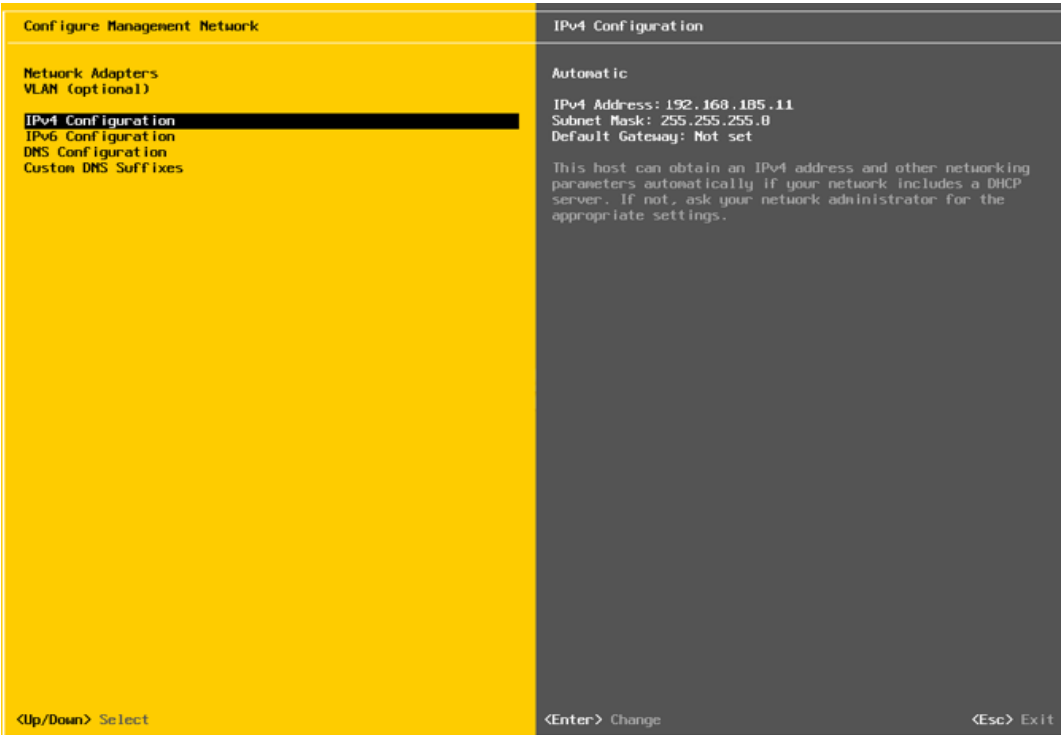


FIGURE IV.12. Configuration management Network window.

- On the “Configuration Management Network” window, “DNS Configuration” is selected as shown in figureIV.13, then hitting ‘Enter’ to change the settings.

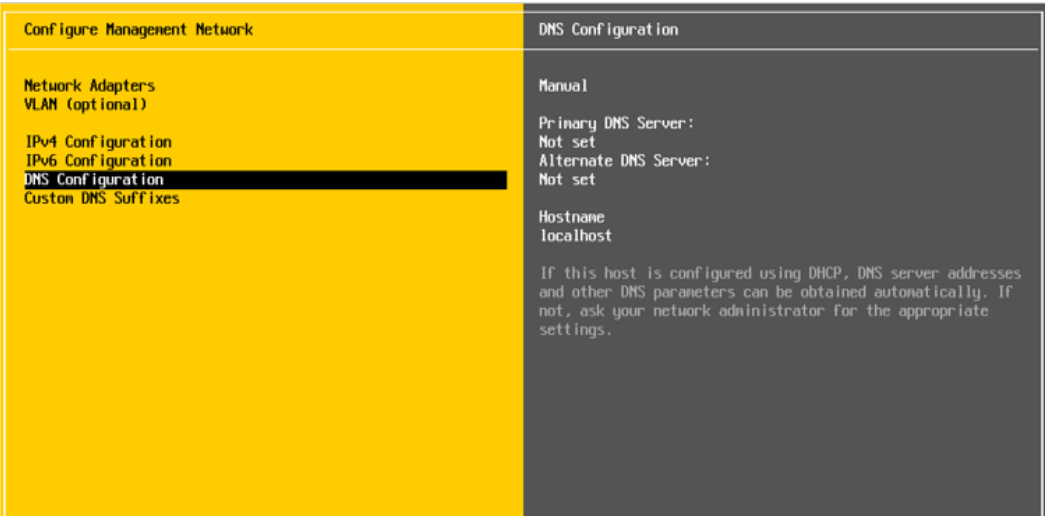


FIGURE IV.13. DNS configuration.

- In the “DNS Configuration” window, the primary DNS Server, alternate DNS server and Hostname must be defined, then hitting enter to continue.
- Once the configuration is completed. ‘ESC’ should be pressed to exit the console, and reboot host to apply the changes.

IV.2 Creating a virtual machine

- To connect to vSphere Web client with SSH connection from the browser, <https://192.168.185.11> is entered. A certificate error page appears.
- Clicking on” Advanced “and proceeding to the website unsafe. It will prompt to **vSphere Web client** as shown in figure IV.14.

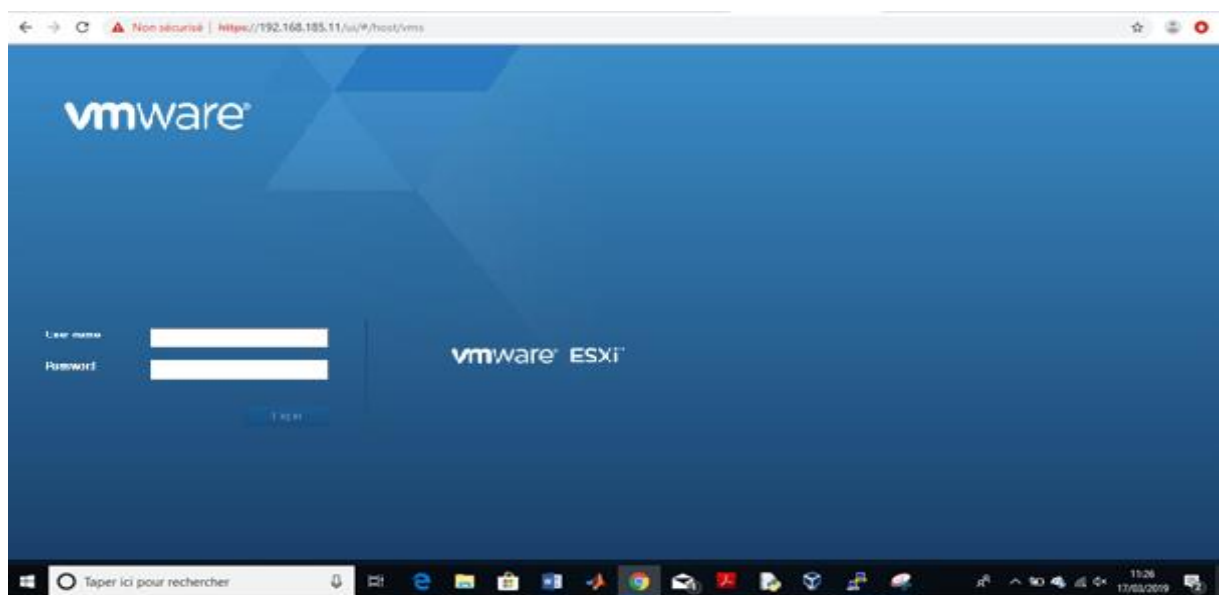


FIGURE IV.14. vSphere Web client.

- Providing the root username and password that was entered in the configuration. Then, login to ESXI Web client.
- First creating a new virtual machine, where the APIC-EM will be installed, in the ESXI Web client.
- ❖ To ensure that the APIC-EM works well within a virtual environment, the virtual machine is configured with the recommended resource pool values listed below:
 - Server Image Format: ISO.
 - Virtual CPU (vCPU): 8
 - CPU (speed) 3.4 GHz
 - Memory: 30 GB

- Disk Capacity: 300 GB
- Network Adapter 1
- Browser: Google Chrome: version 50.0 or later, Mozilla Firefox: version 46.0 or later.

In the creation of new virtual machine, these steps should be followed:

- Clicking on virtual machines from the menu on the left, then clicking new virtual machine. Continuing by stepping by all the necessary configurations like entering the name and choosing LINUX as operating system from the provided list, the storage and edit settings.
- Clicking Ready to complete. Then, clicking Finish to finish the virtual machine configuration.
- Uploading the Cisco APIC-EM ISO image directly to the virtual machine's datastore.
- Attaching the Cisco APIC-EM ISO image as a virtual CD-ROM drive of the virtual machine.
- Booting up the host (virtual machine) and start the configuration wizard.

IV.3 Installing APIC-EM

Once booting the ISO, it will show a license agreement that should be accepted. APIC-EM checks the disk throughput, and warns that the disk needs more then 200Mb/s for production

installation. After ignoring the disk warning, the network adapter is configured as shown in figure IV.15.

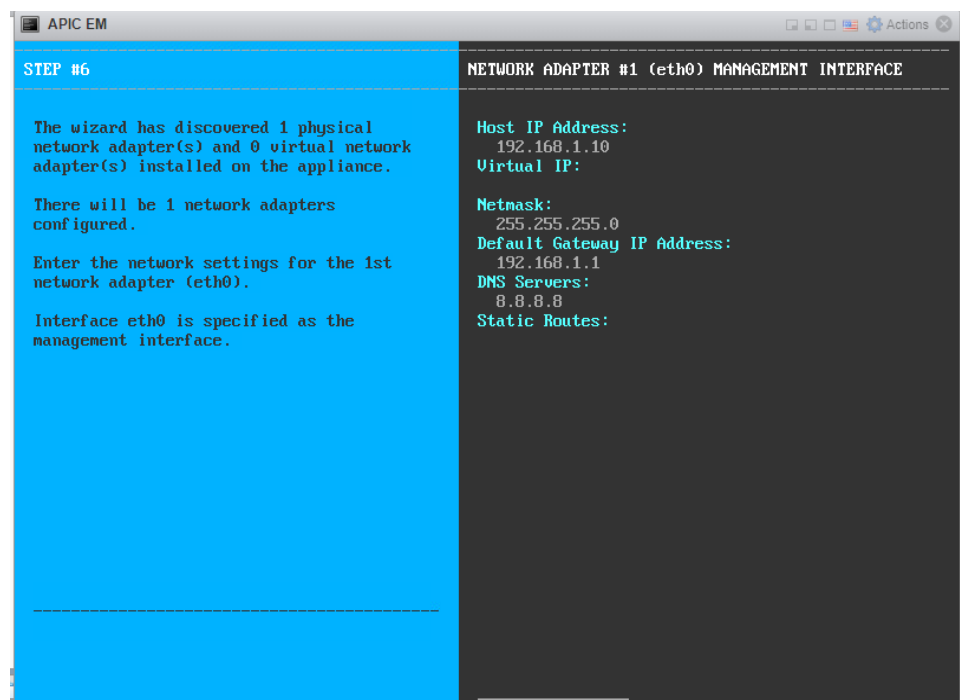


FIGURE IV.15. Network adapter configuration.

-Creating a password to existing Linux user account (grapevine) as shown in figureIV.16.

FIGURE IV.16. Linux user setting.

-Creating an Admin user account to access APIC-EM as shown in figure IV.17. Then Setting the NTP server (Network Time protocol that obtains highly accurate time by synchronizing local clocks to GPS) as shown in figure IV.18.

FIGURE IV.17. APIC-EM Admin user setting.

FIGURE IV.18. NTP Server setting.

- The wizard is now ready, when clicking proceed, it will take a long time before everything is installed and all services are running.
- Clicking on the virtual machine called APIC-EM as shown in figureIV.19, and powering it up after the configuration finishes.

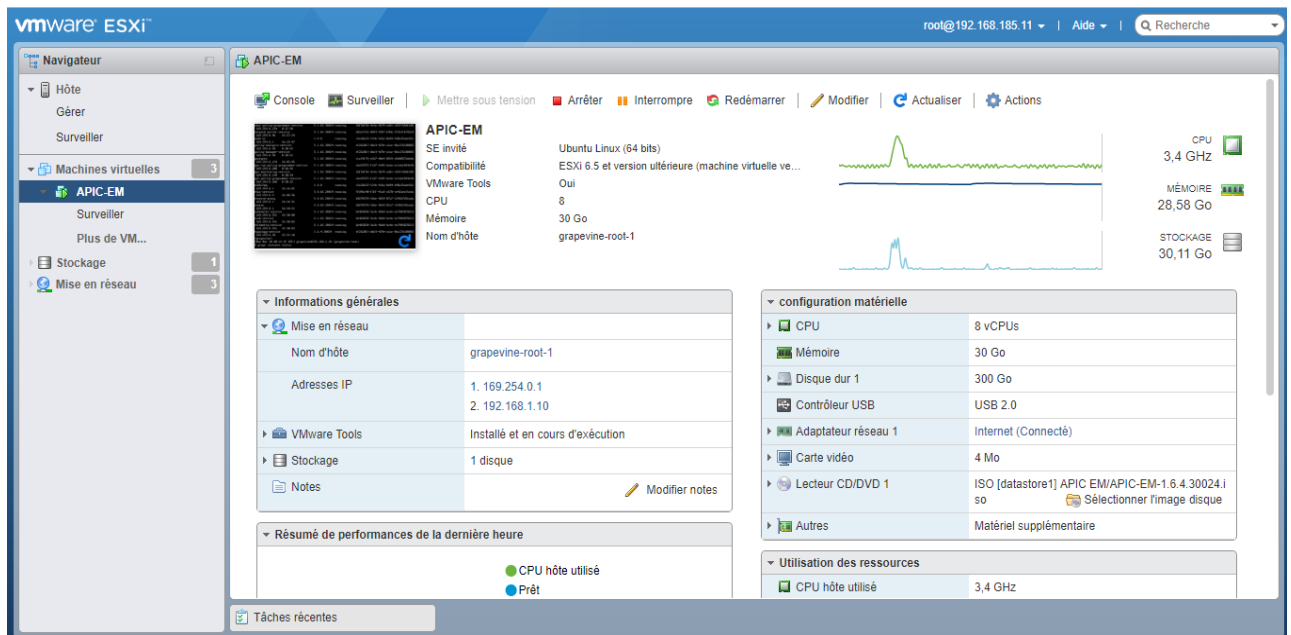


FIGURE IV.19. APIC-EM virtual machine.

To verify the services status:

- Logging in with Linux username ('grapevine') and Linux password.
- Entering the command '**grape instance status**'. The services status is shown in figureIV.20. the running status means that the service is up to use, the starting status means that the service still being uploaded. The services need take time to be on starting status.

```

..1 6 days, 22:31:10
grapevine-log-collector 1.0.0 running 126a1be0-d456-4a41-b2af-68756a76dfde 169.254.0
..1 6 days, 22:31:06
grouping-service 5.1.65.30024 unresponsive b0b24a29-90c2-48ca-9671-4a0d4daac678 169.254.0
..102
identity-manager-pxgrid-service 5.1.65.30024 unresponsive 999cfbd8-8d43-4d68-862a-5d71145b0elf 169.254.0
..85
nbar-policy-programmer-service 5.1.65.30024 starting a7704cd1-9bd6-499d-ab11-9fa3cac54e55 169.254.0
..47
network-poller-service 5.1.65.30024 starting 25e566e1-5702-4873-a55a-6a124ff4d384 169.254.0
..103
node-ui 1.0.0 running 126a1be0-d456-4a41-b2af-68756a76dfde 169.254.0
..1 6 days, 22:31:15
policy-analysis-service 5.1.65.30024 starting 999cfbd8-8d43-4d68-862a-5d71145b0elf 169.254.0
..85
policy-manager-service 5.1.65.30024 unresponsive 2c3a333b-3df1-44a1-aae0-0194b15138be 169.254.0
..169
postgres 5.1.65.30024 running 8a14449b-f031-4602-8bde-81d1cfe0a365 169.254.0
..105 0:25:52
qos-lan-policy-programmer-service 5.1.65.30024 unresponsive 0b98e9f6-9981-4d6e-952b-932af7d079e6 169.254.0
..246
qos-monitoring-service 5.1.65.30024 unresponsive 896fb8f2-ad52-46ce-8ba6-3c6243ff8526 169.254.0
..250
qos-policy-programmer-service 5.1.65.30024 unresponsive 0b98e9f6-9981-4d6e-952b-932af7d079e6 169.254.0
..246
rabbitmq 1.0.0 running 126a1be0-d456-4a41-b2af-68756a76dfde 169.254.0
..1 6 days, 22:31:28
rbac-service 5.0.65.20019 unresponsive b0b24a29-90c2-48ca-9671-4a0d4daac678 169.254.0
..102
reverse-proxy 5.0.65.20019 running 053c59f3-da42-43c0-bd09-b91299d41506 169.254.0
..1 2:26:54
router 5.0.65.20019 running 053c59f3-da42-43c0-bd09-b91299d41506 169.254.0
..1 18:47:58
scheduler-service 5.1.65.30024 starting d4392ed8-ba62-4b6f-8a3f-fa2fdd56c97f 169.254.0
..59
task-service 5.1.65.30024 unresponsive c28e7079-bd8d-41e3-80f3-2eb8ae9cff4a 169.254.0
..188
telemetry-service 5.1.65.30024 starting c28e7079-bd8d-41e3-80f3-2eb8ae9cff4a 169.254.0
..188
topology-service 1.6.4.30024 unresponsive 2d9a5f64-f72f-43ab-b396-9af63bdd4ec0 169.254.0
..242
(grapevine)

```

FIGURE IV.20. APIC-EM services status.

-Then, logging into the controller's GUI by opening the browser and starting an SSH connection by writing the virtual server's IP address (192.168.1.10) that was entered in the configuration

wizard. The results shown in figure IV.21.



FIGURE IV.21. APIC-EM Login window.

IV.4 Device configurations

Before beginning with the network management with APIC-EM. First, initial configuration is given to the network devices shown in figure.IV.22.

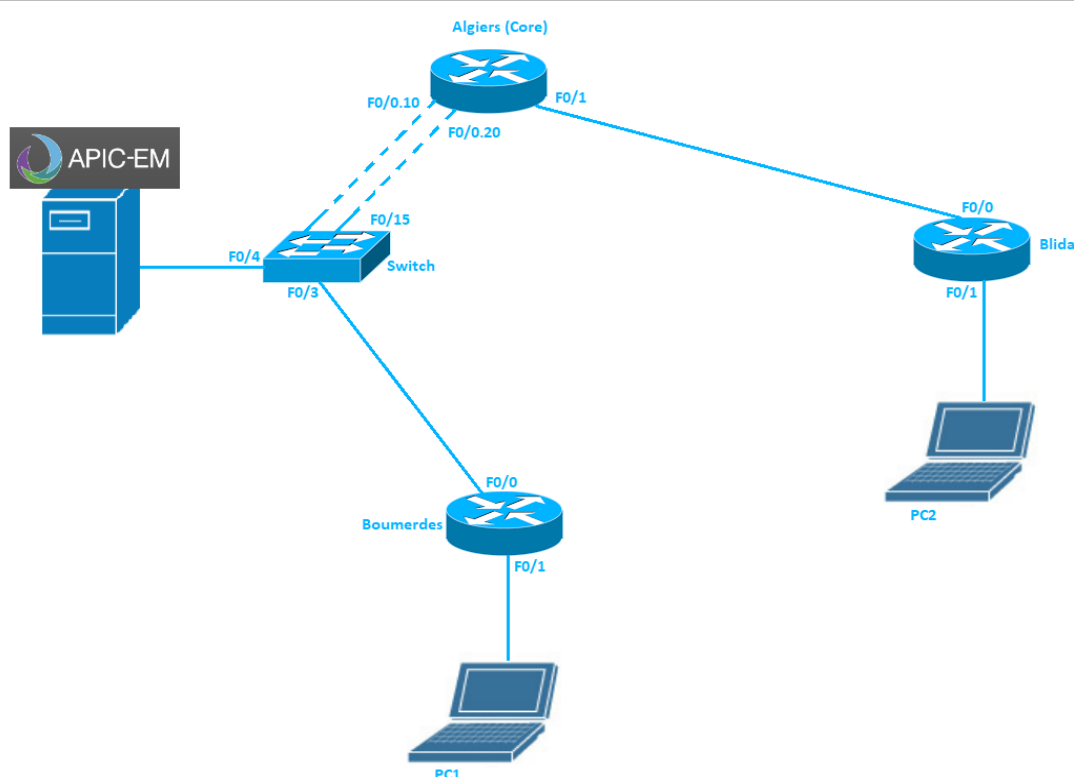


FIGURE IV.22. Project network topology.

The address configuration of the devices is based on the table IV.1 below:

Table.IV.1. Addressing Table.

Device	Interface		IP address	Subnet mask
Router Algiers	F0/0	F0/0.10	192.168.1.3	255.255.255.0
		F0/0.20	192.168.2.3	255.255.255.0
	F0/1		192.168.3.1	255.255.255.0
Switch (In Algiers)	F0/4 (VLAN10)		192.168.1.3	255.255.255.0
	F0/3 (VLAN20)		192.168.2.3	255.255.255.0
Router Boumerdes	F0/0		192.168.2.2	255.255.255.0
	F0/1		192.168.4.1	255.255.255.0
Router Blida	F0/0		192.168.3.2	255.255.255.0
	F0/1		192.168.5.1	255.255.255.0
PC1(In Boumerdes)	/		192.168.4.2	255.255.255.0
PC2(In Blida)	/		192.168.5.2	255.255.255.0

IV.1.3 Basic configuration

As a first step, it is obligatory to establish a basic configuration for each router and switch. This configuration is similar to all network devices. Results are shown in figures below associated with each device of the network.

The basic configuration steps are as follow:

1. Configuration of the names of the routers.
2. Remote Access Configuration (SSH): to access remote devices.
3. Configuration of the console line.
4. Setting the password for the run mode.
5. Encryption service activation.
6. Addressing configuration.
7. Static rout configuration.
8. SNMP protocol configuration.

❖ Router ALGIERS:

In order to form a trunk link with the switch from ALGIERS router, it is necessary to create one sub-interface of interface FastEthernet 0/0 as shown in figure IV.23, for every VLAN configured on the switch. The results of IP address, status and protocol of interfaces by **show ip int brief** command is shown in figure IV.24.

```
Algiers(config)#int fa0/0.10
Algiers(config-subif)#encapsulation dot1Q 10
^
% Invalid input detected at '^' marker.

Algiers(config-subif)#encapsulation dot1Q 10
Algiers(config-subif)#ip address 192.168.1.1 255.255.255.0
Algiers(config-subif)#no shut
Algiers(config-subif)#int fa0/0.20
Algiers(config-subif)#encapsulation dot1Q 20
Algiers(config-subif)#ip address 192.168.2.1 255.255.255.0
Algiers(config-subif)#no shut
Algiers(config-subif)#end
```

FIGURE IV.23. Trunk link configuration.

```
Algiers#show ip int brief
Interface                IP-Address      OK? Method Status      Protocol
FastEthernet0/0          unassigned      YES unset    up          down
FastEthernet0/0.10       192.168.1.1     YES manual    up          down
FastEthernet0/0.20       192.168.2.1     YES manual    up          down
FastEthernet0/1          192.168.3.1     YES manual    up          down
Serial0/0/0              unassigned      YES unset    administratively down down
```

FIGURE IV.24. Status of IPv4 interfaces.

Results of routing table by **show ip route** command shown in figure IV.25.

```
Algiers#sh ip rout
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

S    192.168.4.0/24 [1/0] via 192.168.2.2
S    192.168.5.0/24 [1/0] via 192.168.3.2
C    192.168.1.0/24 is directly connected, FastEthernet0/0.10
C    192.168.2.0/24 is directly connected, FastEthernet0/0.20
C    192.168.3.0/24 is directly connected, FastEthernet0/1
```

FIGURE IV.25. Routing table of router Algiers.

SSH configuration (Same for Boumerdes and Blida routers) is shown in figure IV.26.

```
Algiers(config)#ip domain-name icosnet.com
Algiers(config)#crypto key generate rsa
The name for the keys will be: Algiers.icosnet.com
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
Algiers(config)#
*Mar 19 08:36:21.179: %SSH-5-ENABLED: SSH 1.99 has been enabled
Algiers(config)#username SDN password 07032019
```

FIGURE IV.26. SSH configuration commands.

SNMP configuration (Same for Boumerdes and Blida routers) is shown in figure IV.27.

```
Algiers>enable
Password:
Algiers#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Algiers(config)#snmp-server community public RO
Algiers(config)#end
Algiers#wr
Building configuration...
[OK]
Algiers#
```

FIGURE IV.27. SNMP configuration commands.

❖ Router BOUMERDES:

Results of routing table by **show ip route** command shown in figure IV.28.

```
Boumerdes#sh ip rout
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.4.0/24 is directly connected, FastEthernet0/1
S    192.168.5.0/24 [1/0] via 192.168.2.1
S    192.168.1.0/24 [1/0] via 192.168.2.1
C    192.168.2.0/24 is directly connected, FastEthernet0/0
S    192.168.3.0/24 [1/0] via 192.168.2.1
```

FIGURE IV.28. Routing table of Boumerdes router.

❖ Router BLIDA

Results of routing table by **show ip rout** command is shown in figure IV.29.

```
Blida#sh ip rout
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

S    192.168.4.0/24 [1/0] via 192.168.3.1
C    192.168.5.0/24 is directly connected, FastEthernet0/1
S    192.168.1.0/24 [1/0] via 192.168.3.1
S    192.168.2.0/24 [1/0] via 192.168.3.1
C    192.168.3.0/24 is directly connected, FastEthernet0/0
```

FIGURE IV. 29. Routing table of Blida router.

❖ SWITCH (Positioned in Algiers):

Results of **show ip int brief** command to show address, status and protocol of Vlan interfaces described in figure IV.30.

```
Switch#sh ip int brief
Interface                IP-Address      OK? Method Status      Protocol
Vlan1                    unassigned      YES manual up          up
Vlan10                   192.168.1.3     YES manual up          up
Vlan20                   192.168.2.3     YES manual up          up
```

FIGURE IV.30. Switch Interfaces status.

Mode configuration to each switch port, Fa0/15 mode trunk, Fa0/3 mode access, also for Fa0/4 mode access is shown in figure IV.31.

```
Switch(config-if)# switchport mode trunk
Switch(config-if)# switchport trunk native vlan 1
Switch(config-if)# switchport trunk allowed vlan 10,20
Switch(config-if)#exit
Switch(config)#int fa0/3
Switch(config-if)#switchport mode access
Switch(config-if)#switchport access vlan 20
Switch(config-if)#end
Switch#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#int fa0/4
Switch(config-if)#switchport mode access
Switch(config-if)#switchport access vlan 10
Switch(config-if)#end
```

FIGURE IV.31. Switch interfaces mode configuration.

Results of **show int trunk** command is shown in figure IV.32.

```
Switch#sh int trunk

Port      Mode      Encapsulation  Status      Native vlan
Fa0/15    on        802.1q         trunking    1

Port      Vlans allowed on trunk
Fa0/15    1-4094

Port      Vlans allowed and active in management domain
Fa0/15    1,10,20

Port      Vlans in spanning tree forwarding state and not pruned
Fa0/15    none
```

FIGURE IV.32. Switch trunk interface details.

Results of **show vlan** command showing each interface vlan name, status and ports described in figure IV.33.

```
Switch#sh vlan

VLAN Name                Status    Ports
-----
1    default                active    Fa0/1, Fa0/2, Fa0/5, Fa0/6
                                           Fa0/8, Fa0/9, Fa0/10, Fa0/11
                                           Fa0/12, Fa0/13, Fa0/14, Fa0/16
                                           Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                           Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                           Gi0/1, Gi0/2
10   server                  active    Fa0/4, Fa0/7
20   bmds                    active    Fa0/3
1002 fddi-default          act/unsup
1003 token-ring-default    act/unsup
1004 fddinet-default       act/unsup
1005 trnet-default         act/unsup

VLAN Type  SAID      MTU    Parent RingNo BridgeNo Stp    BrdgMode Trans1 Trans2
-----
1    default  100000000 1500    0      0      0      0      0      0      0
10   server   100000000 1500    0      0      0      0      0      0      0
20   bmds     100000000 1500    0      0      0      0      0      0      0
1002 fddi-default  100000000 1500    0      0      0      0      0      0      0
1003 token-ring-default 100000000 1500    0      0      0      0      0      0      0
1004 fddinet-default  100000000 1500    0      0      0      0      0      0      0
1005 trnet-default    100000000 1500    0      0      0      0      0      0      0
```

FIGURE IV.33. Switch Vlan interfaces details.

IV.5 Using APIC-EM

After devices initial configurations, logging in into APIC-EM as shown in figure IV.34.

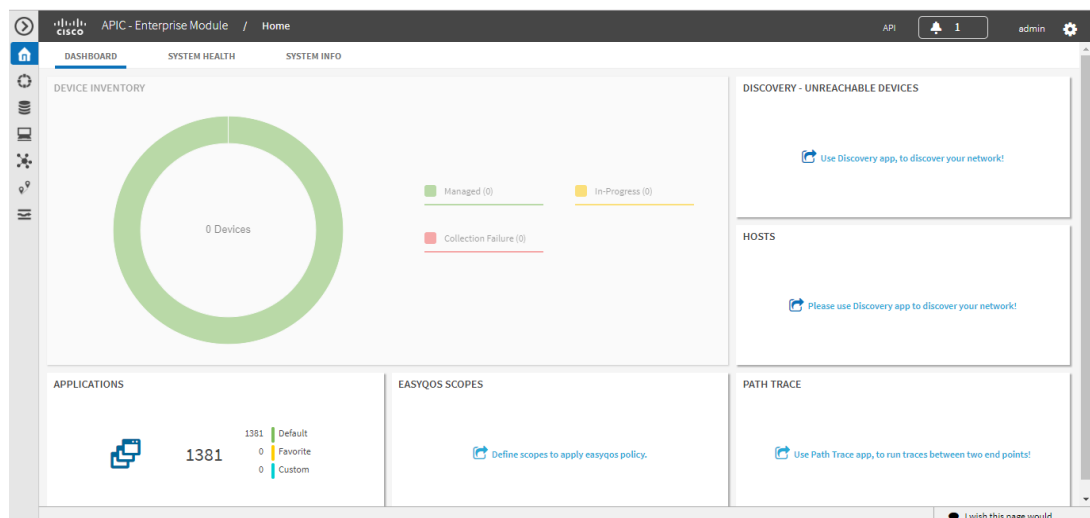
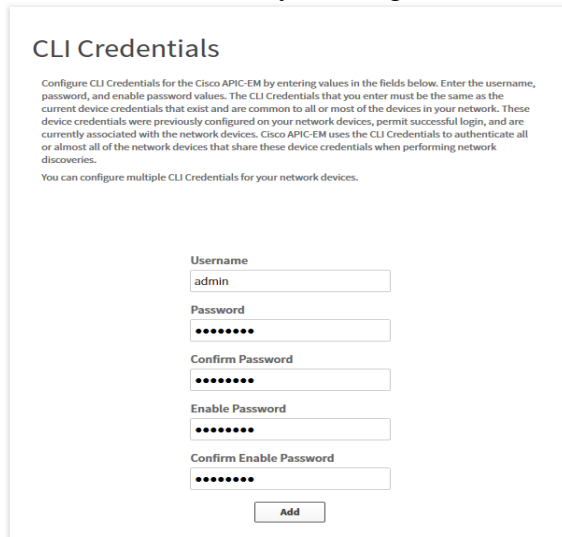


FIGURE IV.34. APIC-EM home page.

-In order to discover these devices on the network, CDP, SSH and SNMP are used. CLI, SSH and SNMP are required so that APIC-EM can access devices. So, they are configured by clicking on the “gear” icon in the top right corner, and selecting “settings”.

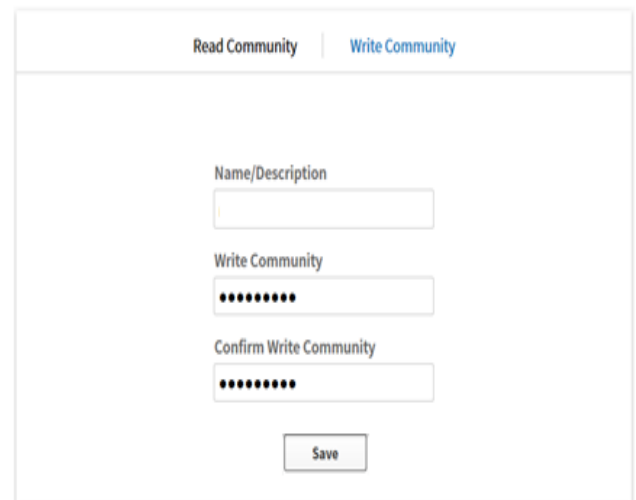
-In the discovery credentials menu, “CLI Credentials” is selected.

Entering the username and password used for SSH and the enable password as shown in figure IV.35, and clicking on the Add button. Then, choosing the read only, the write and the SNMP community as in figure IV.36. Then, clicking on the save button.



The screenshot shows the 'CLI Credentials' configuration window. It contains a text area with instructions: 'Configure CLI Credentials for the Cisco APIC-EM by entering values in the fields below. Enter the username, password, and enable password values. The CLI Credentials that you enter must be the same as the current device credentials that exist and are common to all or most of the devices in your network. These device credentials were previously configured on your network devices, permit successful login, and are currently associated with the network devices. Cisco APIC-EM uses the CLI Credentials to authenticate all or almost all of the network devices that share these device credentials when performing network discoveries. You can configure multiple CLI Credentials for your network devices.' Below the text are five input fields: 'Username' (containing 'admin'), 'Password' (masked with dots), 'Confirm Password' (masked with dots), 'Enable Password' (masked with dots), and 'Confirm Enable Password' (masked with dots). An 'Add' button is at the bottom.

FIGURE IV.35. APIC-EM CLI Credentials window.



The screenshot shows the 'SNMP configuration' window. It has two tabs: 'Read Community' (selected) and 'Write Community'. Under 'Read Community', there is a 'Name/Description' field (empty) and a 'Write Community' field (masked with dots). Under 'Write Community', there is a 'Confirm Write Community' field (masked with dots). A 'Save' button is at the bottom.

FIGURE IV.36. APIC-EM SNMP configuration window.

IV.1.4 Discovery

-In order to discover the network, the discovery icon on the left side is selected.

First, entering the name of the discovery and the seed IP address from the network as shown in figure IV.37, leaving the subnet filter empty so that the controller does not exclude any subnet, with the CDP level with its default value.

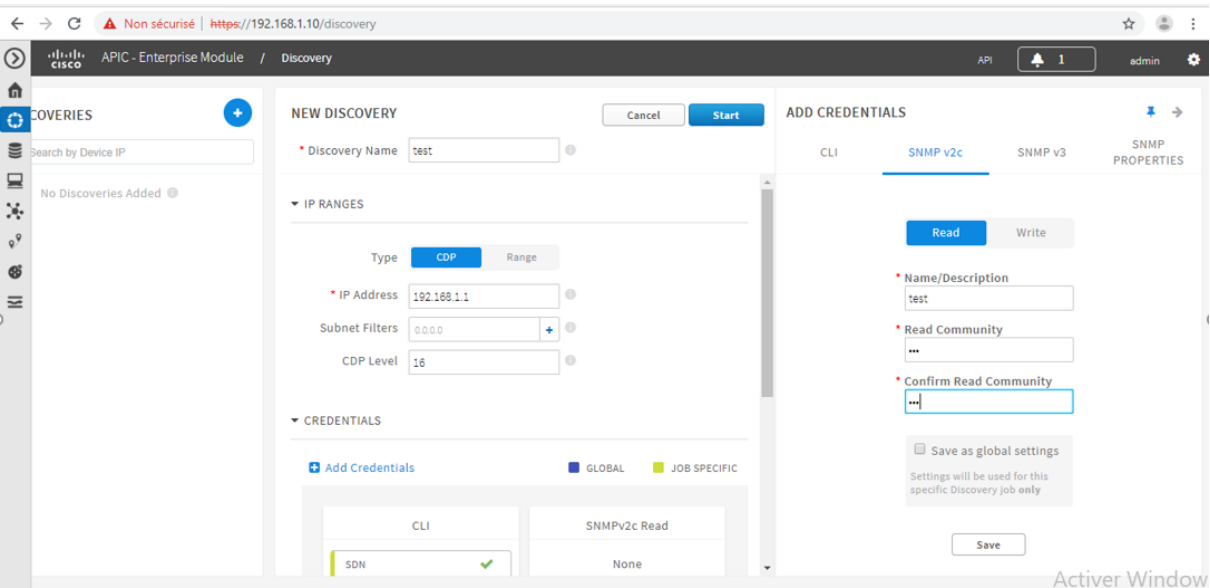


FIGURE IV.37. APIC-EM Discovery page.

- Clicking on SNMPv2c in the Add Credentials pane.
- Clicking “Read” in the SNMPv2c panel. Then, typing “public” in the Read Community and in “confirm Read Community” field and clicking “save”.

Then, clicking Start. The user interface displays an In-progress message to indicate that scanning is taking place. Results of the discovery is shown in figure IV.38.

The screenshot shows the 'DEVICES' section of the APIC-EM interface. It has a 'LIST' tab selected and a 'CHART' tab. Below the tabs, there's a legend for device status: SUCCESS (green checkmark), UNREACHABLE (grey circle), FAILURE (red X), NOT TRIED (grey circle), and UNAVAILABLE (grey question mark). Below the legend is a table with the following data:

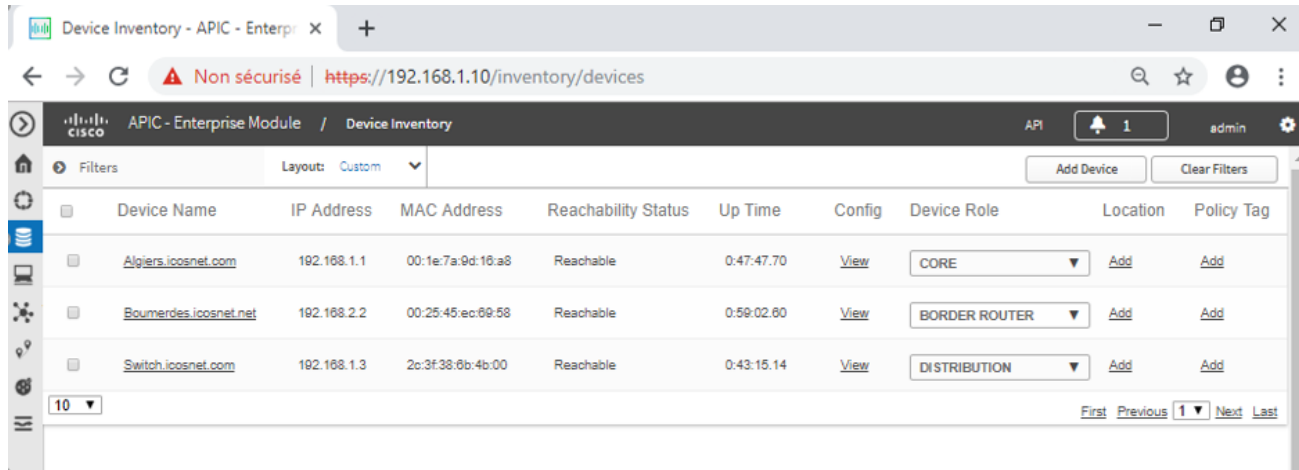
IP ADDRESS	DEVICE NAME	STATUS	ICMP	SNMP	CLI
192.168.3.2	Blida.icosnet.com	✓	✓	✓	✓
192.168.3.1	Algiers.icosnet.com	✓	✓	✓	✓
192.168.1.3	Switch.icosnet.com	✓	✓	✓	✓
192.168.2.2	Boumerdes.icosnet.net	✓	✓	✓	✓

At the bottom, there's a pagination bar showing '10 per page' and '4 Devices'.

FIGURE IV.38. APIC-EM Discovery process results.

IV.1.5 Inventory

-After performing the discovery scan, Device Inventory icon is clicked to view the discovered devices information as figure IV.39 describes.



The screenshot shows the 'Device Inventory' page in the APIC-EM interface. The page displays a table of discovered devices with columns for Device Name, IP Address, MAC Address, Reachability Status, Up Time, Config, Device Role, Location, and Policy Tag. The table lists three devices: Algiers.icosnet.com, Boumerdes.icosnet.net, and Switch.icosnet.com. Each device has a 'View' link next to its Config column. The page also includes a search bar, filters, and a 'Layout' dropdown set to 'Custom'. The bottom of the table shows pagination controls for 10 items, with 'First', 'Previous', '1', 'Next', and 'Last' buttons.

Device Name	IP Address	MAC Address	Reachability Status	Up Time	Config	Device Role	Location	Policy Tag
Algiers.icosnet.com	192.168.1.1	00:1e:7a:9d:16:a8	Reachable	0:47:47.70	View	CORE	Add	Add
Boumerdes.icosnet.net	192.168.2.2	00:25:45:ec:88:58	Reachable	0:58:02.60	View	BORDER ROUTER	Add	Add
Switch.icosnet.com	192.168.1.3	2c:3f:38:6b:4b:00	Reachable	0:43:15.14	View	DISTRIBUTION	Add	Add

FIGURE IV.39. APIC-EM Inventory page.

IV.1.6 TOPOLOGY

By clicking on the topology icon on the left-hand side of the menu, the network topology is shown as in figure IV.40.

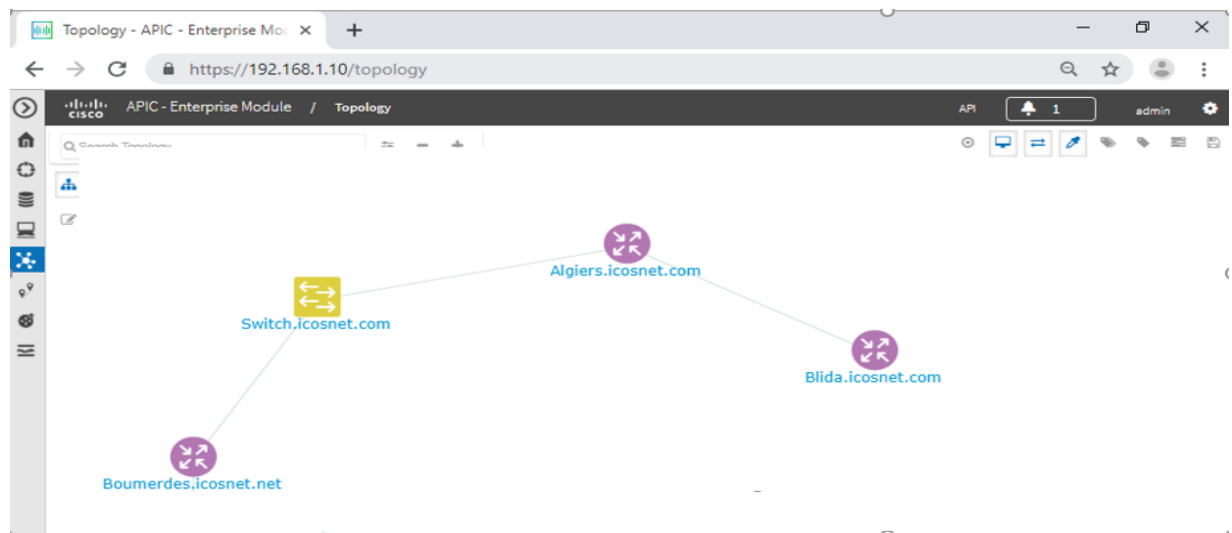


FIGURE IV.40. APIC-EM Topology page.

IV.1.7 Path Trace

Using Path Trace application of APIC-EM to determine the path that traffic takes from one network node to another

- Opening the Path Trace navigation-pane item and following the steps listed in section III.3.4 and described in figure IV.41. Selecting the source Algiers router with 192.168.1.1 port address to destination Boumerdes router port address 192.168.4.1.

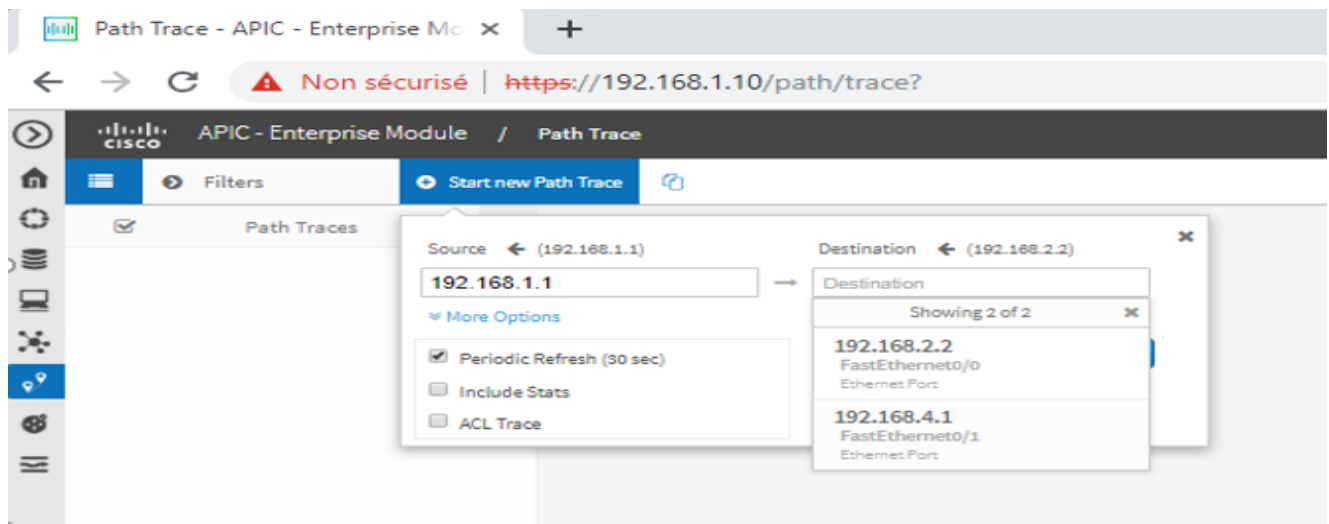


FIGURE IV.41. APIC-EM Path trace navigation-pane.

- After some time passes, the traffic route results from the source to the destination is shown in figure IV.42.

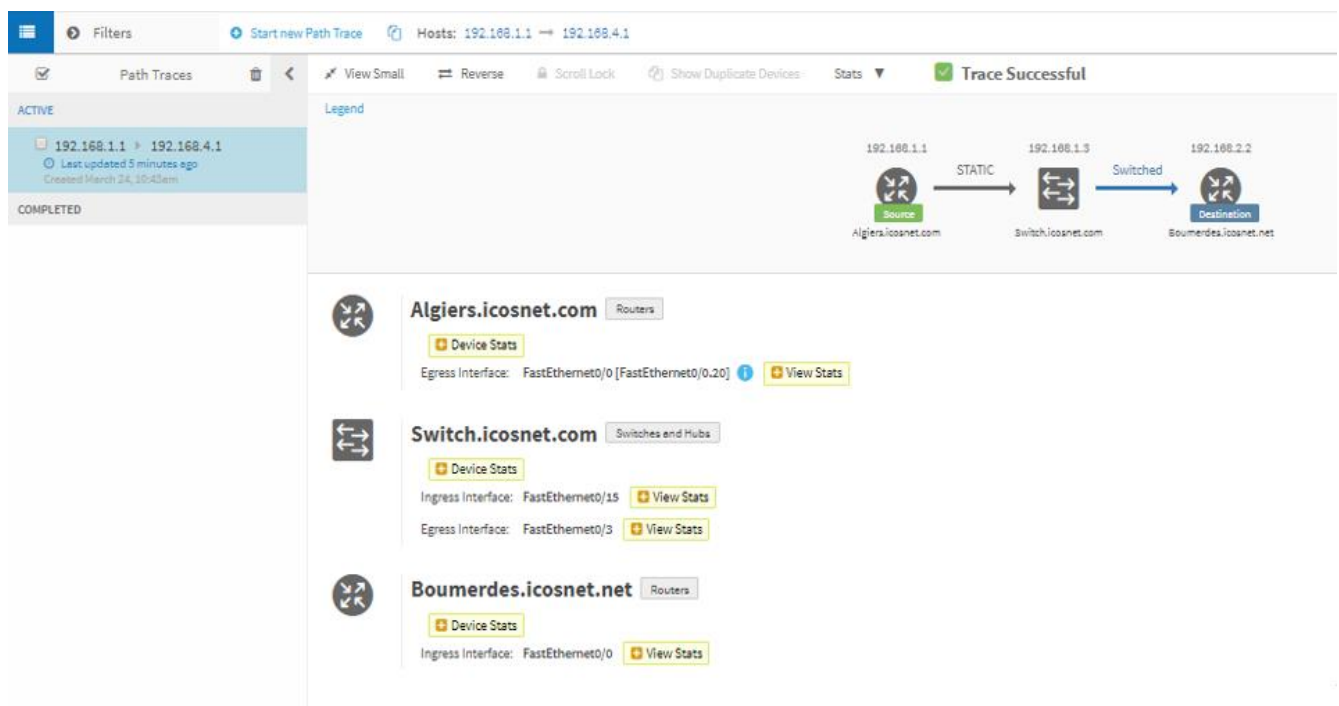


FIGURE IV.42. Path-trace results page.

IV.6 API Interfaces

This section is devoted to explain a REST API based application example, and its building steps. It is developed to test out one application of SDN features using Python and the Flask web framework.

Python simplified the task of creating a REST API. It is an object-oriented, high-level programming language with integrated frameworks primarily for web and application development [53]. One of its useful frameworks is FLASK, which is a web framework that provides functionality for building web applications, including managing HTTP requests and rendering templates [54].

The project API is named API interfaces. It is built up to get information about interfaces of a desired device in the network. This API reads from APIC-EM controller database, and exposes to user the obtained data in a GUI format based on flask-gentelella template.

IV.1.8 Requirements

- ✓ Installing Python.
- ✓ Installing PyCharm as cross platform integrated development environment (IDE).
- ✓ Installing Flask using the pip package manager for Python.
- ✓ A web browser.

IV.1.9 Steps of building API interfaces

STEP1: By applying the steps mentioned in section III.5.1 on the aim of API interfaces, the following identifications are obtained:

1. **Object Model:** The resource of the API is 'Device' and sub-resource is 'interfaces'.

2. **Model URLs:** Data is fetched from APIC-EM database according to a specific URL format giving in APIC-EM service API. So, the relationship between device and its interfaces has unique URL format by specifying the id of the device.
3. **Representations:** JSON format is used for the API interfaces.
4. **HTTP Methods:** The possible action in the case of API interfaces is only reading information from database. So, GET method is used to request data.

STEP2: Creating the backend API code in PyCharm editor following the flow chart workflow described in figure IV.43, and by using flask server at <http://127.0.0.1:5000/> the code is executed. When matching between the path provided and the writing in code, the resulted data are shown in the browser in JSON format.

STEP3: Cloning from the distributed version-control system GitHub a flask-gentelella template code (used as GUI). Then, integrating API interfaces backend code on it, and modifying the interface elements of GUI according to the way API data are shown clearly.

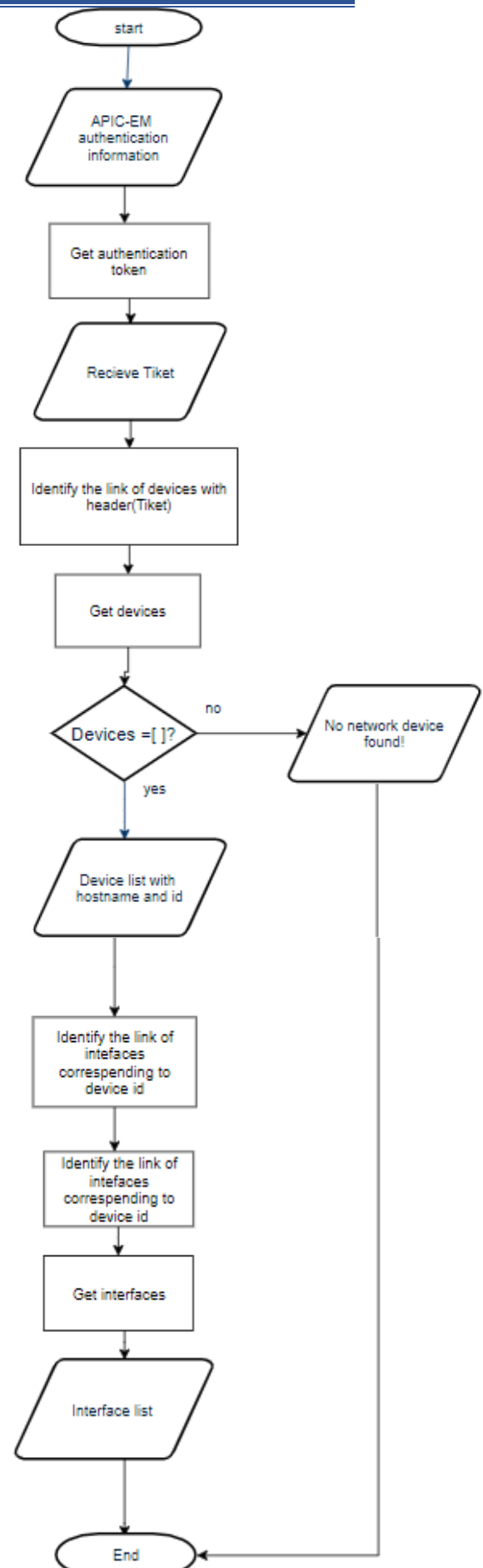


FIGURE IV.43. Workflow of API interfaces program.

IV.1.10 RESULTS

API interfaces is designed to read from any APIC-EM controller, this project was supposed to read from the controller of network shown in Figure IV.22. But, APIC-EM controller was not accessible. So, the API was tested on another APIC-EM network controller provided by cisco, which is sandbox APIC-EM (a Lab environment), and the results are according to its network topology shown in figure IV.44 [56].

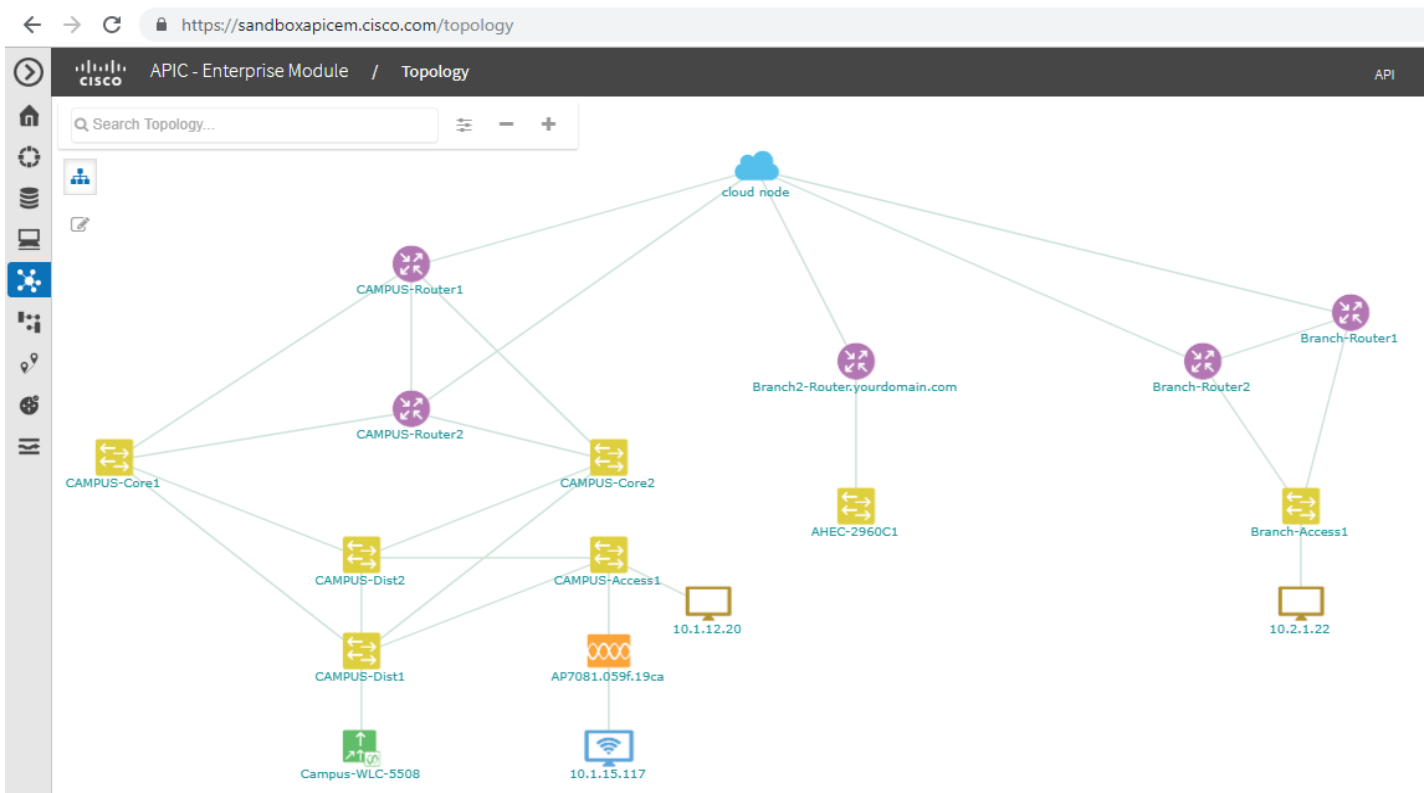


FIGURE IV.44. Sandbox APIC-EM network topology.

- ❖ The results shown by the end of this section consist of a welcoming page to the API interfaces users with different selectable devices presented in left side panel. Then, a page displaying information about interfaces for a selected device. For selecting another device, the ‘devices’ icon in the left should be clicked, then choosing one device from the devices menu that appears as shown in figure IV.49.

The interface information contributes the following parameters:

- **portName:** Interface name.
- **id:** A unique identifier comprised of numbers and letters that corresponds with a device.
- **Admin status:** Administrative status of the interface.

- **Mac Address:** MAC address of interface.
- **series:** Series of the device.
- **ifIndex:** Interface Index is unique identifying number for each device used to designate network traffic paths.
- **interfaceType:** Interface type as Physical or Virtual.
- **speed:** Speed of the interface.
- **ipv4Address:** IPv4 address assigned for interface.
- **duplex:** Interface duplex as AutoNegotiate or FullDuplex (Is there any).
- **serialNo:** Serial number of the device.
- **voiceVlan:** Vlan information of the interface.
- **mediaType:** Media Type of the interface (as RJ45 connector).
- **portType:** Port type as Ethernet Port / Ethernet SVI / Ethernet Sub Interface.
- **status:** Interface status as Down / Up.
- **description:** interface description.
- **lastUpdated:** Time when the device interface info last got updated.
- **nativeVlanId:** Vlan to receive untagged frames on trunk port.
- **vlanId:** Vlan ID of interface.
- **ipv4Mask:** Subnet mask for IPv4 address assigned for interface.
- **ospfSupport:** Flag for OSPF enabled / disabled.
- **portMode:** Port mode as access, trunk, routed.

❖ Welcoming page:

- The first appearing page when accessing API Interfaces application as figure IV.45.

-It shows in the left panel the API devices that can be accessed as in figure IV.46.

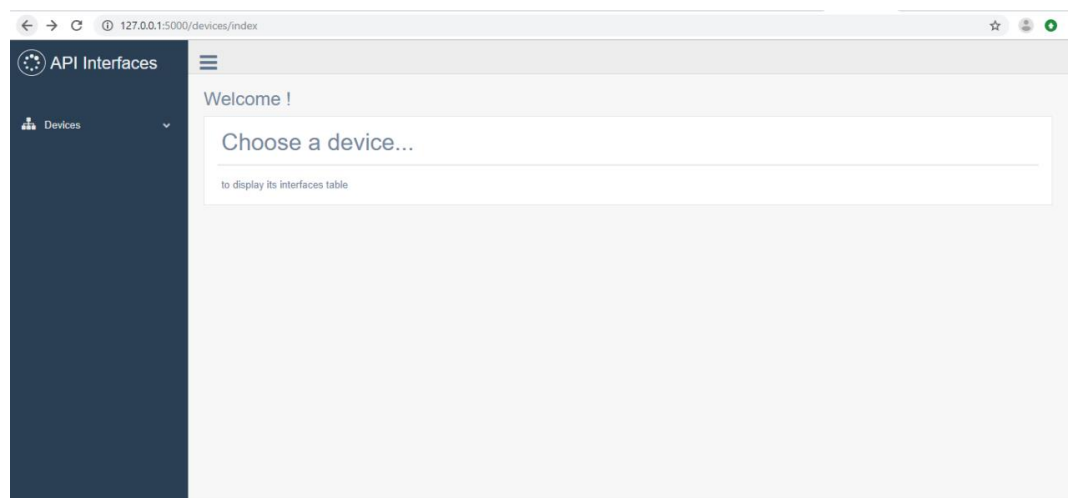


FIGURE IV.45. API Interfaces welcoming page.

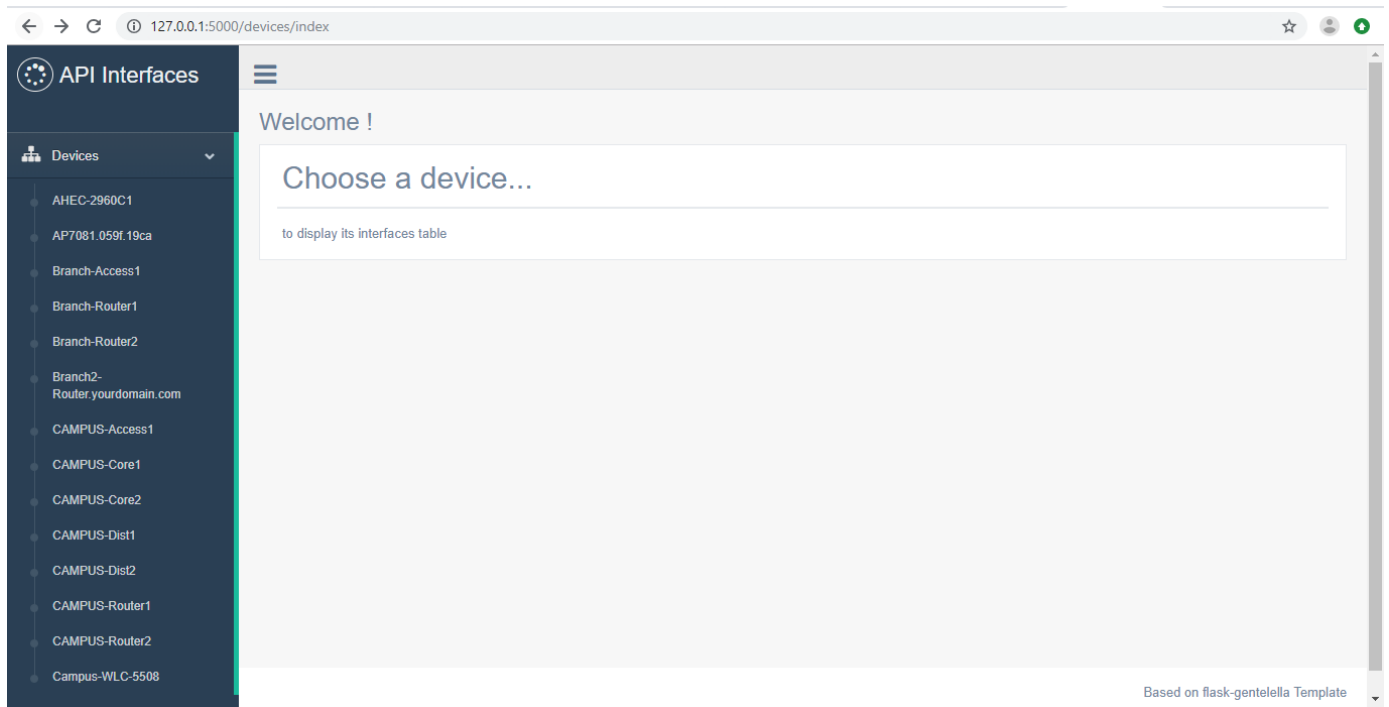


FIGURE IV.46. API Interfaces devices list.

❖ **Information of device Interfaces page:**

The page that shows the interfaces information of the selected device (CAMPUS-Router1) in column ribbles as figure IV.47 shows. Because the 22 information can't be displayed in one screen, scrolling right is done as shown in figure IV.48.



FIGURE IV.47. Interfaces information page of ‘CAMPUS-Router1’.

← → ↻ 127.0.0.1:5000/devices/9712ab62-6140-43fd-b1ee-1b07d1fb67d7 ☆ 1

Device: CAMPUS-Router1

Search for... Go!

Interfaces

Show 10 entries Search:

mediaType	portType	status	description	lastUpdated	nativeVlanId	vlanId	ifIndex	ipv4Mask	ospfSupport	portMode
RJ45	Ethernet Port	down		2019-05-18 11:19:35.931	None		5	None	false	None
RJ45	Ethernet Port	up		2019-05-18 11:19:35.931	None		1	255.255.255.248	false	routed
RJ45	Ethernet Port	up		2019-05-18 11:19:35.931	None		2	255.255.255.248	false	routed

FIGURE IV.48. Page of next interfaces information of ‘CAMPUS-Router1’.

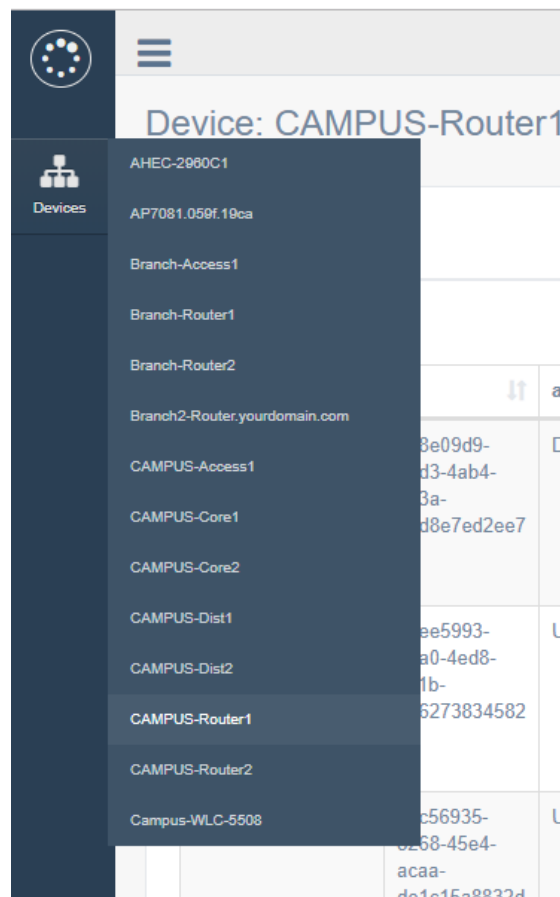


FIGURE IV.49. Devices menu to reselect a device.

Conclusion

The work presented in this chapter is a completely practical work, where professional devices were used. During the study of one of SDN approaches, VMware ESXI 6.5 was installed on the server (Not a necessary step when it is already preinstalled), APIC-EM application was installed in virtual machine on the server, and traditional network was built up as well its devices were configured with basic configuration.

APIC-EM application was deployed as SDN controller to manage the traditional network, and it has discovered all devices.

API Interfaces was built up and executed to read from APIC-EM. It has displayed information about device interfaces.

Thus, the SDN solution proposed in this project has satisfactory results.

Conclusion

This project is a result of the rising demands on networks and the different needs of newer networking and IT trends. This result is the movement from a traditional network into a Software Defined Networking one, that permits to control and manage networking devices, hence the entire network in an easy and more efficient manner. This is done by decoupling the control plane from the data plane and providing programmability for network application development.

SDN is considered as a promising solution to meet the demands of enhancing flexibility and customizability level, configuration simplicity, performance improvement, cost efficiency and security.

In this project, the question that was proposed at the beginning was answered by presenting the concept of SDN with its different functionalities and benefits. Moreover, the architectural components of SDN which are: infrastructure layer, control layer, application layer and the bounds APIs have been exposed and studied in details. Finally, the proper functioning of this solution was tested and verified, in addition with creating and developing a new API that extracts interface information of each device contained in the network from the controller.

This final year project consists mainly of the implementation of an approach of SDN network based on an existing traditional network of ICOSNET company, and its management through APIC-EM as an SDN controller, with the help of its built-in applications and the API based application that has been created.

To improve the project and to ensure a continuous evolution on Software defined networking field, it is essential to suggest a number of short, medium and long-term development perspectives, namely:

- Using SDN for managing home networks.
- Study of distributed SDN Controllers architecture.
- Implementing an SDN network using OpenFlow protocol.

References

- [1] Hassan Habibi Gharakheili; The Role of SDN in Broadband Network (2017, springer Singapore).
- [2] <https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/>, consulted (25/01/2019).
- [3] <https://www.lifewire.com/how-routers-work-816456>, consulted (25/01/2019).
- [4] https://www.cobranet.info/support/design/ethernet_overview, consulted (26/01/2019).
- [5] <https://study-ccna.com/types-of-ethernet-cabling/>, consulted (26/01/2019).
- [6] <https://www.techopedia.com/definition/2282/server>, consulted (24/01/2019).
- [7] https://www.vfu.bg/en/e-Learning/Computer-Networks/Introduction_Computer_Networking.pdf, consulted (30/01/2019).
- [8] https://www.g-w.com/pdf/sampchap/9781605253565_ch09.pdf, consulted (04/02/2019).
- [9] <https://searchservvirtualization.techtarget.com/definition/virtual-machine>, consulted (04/02/2019).
- [10] Guy Pujolle, Networks & Telecommunication_ Advanced Networks, Software Networks_ Virtualization, SDN, 5G, Security (2015, Wiley-ISTE).
- [11] William Stallings, Foundations of Modern Networking_ SDN, NFV, QoE, IoT, and Cloud (2015, Addison-Wesley Professional).
- [12] ONF Document Name: TR_SDN ARCH Overview 1.1, 2014; section: SDN Architecture Requirements and Scope.
- [13] “Software-defined networking: The new norm for networks,” Palo Alto, CA, USA, White Paper, Apr. 2012.
- [14] H. Song, Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane, 2013.
- [15] <https://www.opennetworking.org/sdn-definition/>, consulted (02/02/2019).
- [16] Paul Goransson, Chuck Black, How SDN works, software defined networks, 2014.
- [17] https://www.opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf, consulted (15/02/2019).
- [18] Rajesh Kumar, Software Defined Networking - a definitive guide, 2013.
- [19] Fei Hu - Network Innovation through OpenFlow and SDN_ Principles and Design (2014, CRC Press).

- [20] <https://www.sdxcentral.com/networking/sdn/definitions/southbound-interface-api/>, consulted (16/04/2019).
- [21] <http://netlab.boun.edu.tr/WiSe/doku.php/research:sdn>, section: Distributed Controller Mechanism for SDN-based Campus Networks, consulted (14/04/2019)
- [22] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.- J. Lee, P. Yalagandula, Automated and scalable qos control, 2010.
- [23] K. Teemu et al, Onix: a distributed control platform for large-scale production networks, OSDI'10, 2010.
- [24] T. Koponen et al., “Onix: A distributed control platform for large-scale production networks,” in Proc. 9th USENIX Conf. Oper.Syst. Design Implement., 2010.
- [25] <https://www.sdxcentral.com/networking/sdn/definitions/what-is-ryu-controller/>, consulted (14/04/2019).
- [26] <http://www.brianlinkletter.com/using-the-pox-sdn-controller/> , consulted (13/04/2019).
- [27] <http://yuba.stanford.edu/~derickso/docs/hotsdn15-erickson.pdf>, consulted(17/04/2019).
- [28] <http://www.projectfloodlight.org/floodlight/> , consulted (13/04/2019).
- [29] <https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-apic-em>, consulted (15/04/2019).
- [30] https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/application-policy-infrastructure-controller-enterprise-module/1-0-x/deploy-guide/b_apic-em_deploy_guide_v_1_0_0_x/b_apic-em_deploy_guide_v_1_0_0_x_chapter_01.html, consulted (20/04/2019).
- [31] <https://www.linkedin.com/pulse/cisco-apic-em-sdn-controller-first-impressions-peter-moorey>, consulted (13/04/2019).
- [32] <https://forum.networklessons.com/t/introduction-to-apic-em/1352>, consulted (13/04/2019).
- [33] http://www.izationeffect.com/wp-content/uploads/2017/04/LH2016-1060D1_AIM_Cisco_At-A-Glance_APIC-EM_07.pdf, consulted (15/04/2019).
- [34] <http://webcache.googleusercontent.com/search?q=cache:http://www.student.montefiore.ulg.ac.be/~agirmanr/src/tfe-sdn.pdf#17>, consulted (20/04/2019).
- [35] <https://searchnetworking.techtarget.com/definition/SDN-application-software-defined-networking-application>, consulted (22/04/2019).
- [36] E. Reinecke, “Mapping the future of software-defined networking,” 2014.

-
- [37] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, and N. Zeldovich, “Delegating network security with more information,” in *Proc. 1st ACM WREN*, 2009.
- [38] J. Fu, P. Sjödin, and G. Karlsson, “Intra-domain routing convergence with centralized control,” *Comput. Netw.*, vol. 53, 2009.
- [39] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in SDN-OpenFlow networks,” *Comput. Netw.*, vol. 71, 2014.
- [40] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Commun. Mag.*, vol. 51, 2013.
- [41] <https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/performance-routing-pfr/200281-Introduction-To-IWAN-And-PfRv3.html>, consulted (26/04/2019).
- [42] <https://developer.cisco.com/docs/network-plug-n-play/#!network-plug-and-play/introduction>, consulted (26/04/2019).
- [43] <https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Dec2017/APIC-EM-EasyQoS-DesignGuide-Dec2017.html>, consulted (26/04/2019).
- [44] https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/application-policy-infrastructure-controller-enterprise-module/1-4-x/path_trace/user-guide/b_Cisco_Path_Trace_User_Guide_1_4_0_x/b_Cisco_Path_Trace_Solution_Guide_1_4_0_x_chapter_01.html, consulted (26/04/2019).
- [45] <https://netfv.wordpress.com/2017/02/13/sdn-northbound-interfaces-nbi-and-southbound-interfaces-sbi/>, consulted (23/04/2019).
- [46] https://www.researchgate.net/publication/330306237_The_Northbound_APIs_of_Software_Defined_Networks, consulted (23/04/2019).
- [47] <https://www.mulesoft.com/resources/api/what-is-rest-api-design>, consulted (28/04/2019).
- [48] <https://www.restapitutorial.com/lessons/httpmethods.html>, consulted (28/04/2019).
- [49] https://www.academia.edu/29207256/THE_NORTHBOUND_APIs_OF_SOFTWARE_DEFINED_NETWORKS, consulted (26/04/2019).
- [50] <https://www.mulesoft.com/resources/api/types-of-apis>, consulted (26/04/2019).
- [51] <https://www.codementor.io/sagaragarwal94/building-a-basic-restful-api-in-python-58k02xsiq>, consulted (28/04/2019).
- [52] <https://restfulapi.net/rest-api-design-tutorial-with-example/>, consulted (02/05/2019).
- [53] <https://en.wikipedia.org/wiki/WebGUI>, consulted (02/05/2019).
-

[54] <https://www.pythonforbeginners.com/learn-python/what-is-python/>, consulted (05/05/2019).

[55] <https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask>, section: Creating a Basic Flask Application, consulted (05/05/2019).

[56] <https://sandboxapicem.cisco.com>, consulted (15/05/2019).