REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'EINSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DE GÉNIE ÉLECTRIQUE ET D'INFORMATIQUE

DÉPARTEMENT D'ÉLECTROTECHNIQUE

Mémoire de Fin d'Etude

EN VUE DE L'OBTENTION DU DIPLOME DE MASTER 2 EN TELECOMMUNICATIONS

OPTION : RESEAUX ET TELECOMMUNICATIONS

Thème :

IMPLEMENTATION DU SDN DANS UNE STRUCTURE IP/MPLS

Soutenu le : 12/07/2018

Présenter par :

Mr : OULD LAMARA SAMI

Mr : TAKILT MOUSSA

Devant le jury composé de :

Encadreur M^{r} : M^{r}. LAHDIR	**MCA à l'UMMTO**
Président du jury M^{r} : Y. ATTAF	**MCB à l'UMMTO**
Examinateur M^{r} : M. SEHAD	**MCB à l'UMMTO**

2017-2018

Dédicace

Tiens tout d'abord à remercier le bon Dieu de m'avoir aidé à réaliser ce mémoire

Que je dédie à :

A la prunelle de mes yeux, celle qui m'a soutenu surtout avec son Dou3a jour et nuit pour qu'elle me voit au sommet et comme une étoile, à la source de la Douceur. *A toi ma très chère mère.*

A la personne qui a sacrifié sa vie pour moi, et qui a pris le défi pour mes études Et mas éclairé le chemin de ma réussite. *A toi mon très cher Père.*

A la mémoire de mes grands-pères (EL-HDJ SAÏD & EL-HADJ ALI) et ma grand-mère (MESSAOUDA)

A ma très chère Yema SASSI (TASSADIT).

A mes chers frères Qui sont toujours à mes côtés et leurs femmes et enfants.

A mes très chères sœurs qui ont sacrifié leur temps pour que je sois à l'aise dans Mes études, et aussi à leur maries.

A tous mes ami(e)s

A mon binôme SAMI et sa famille.

Dédicace

Tiens tout d'abord à remercier le bon Dieu de m'avoir aidé à réaliser ce mémoire

Que je dédie à :

A la prunelle de mes yeux, celle qui m'a soutenu surtout avec son Dou3a jour et nuit pour qu'elle me voit au sommet et comme une étoile, à la source de la Douceur. *A toi ma très chère mère.*

A la personne qui a sacrifié sa vie pour moi, et qui a pris le défi pour mes études Et mas éclairé le chemin de ma réussite. *A toi mon très cher Père.*

A mon très cher frère Yanis.

A ma chère sœur Qui est toujours à mes côtés et à son marie Hacene.

A la mémoire de mes grands-parant (Messaoud et Madjid)

A tous mes ami(e)s

A mon binôme Moussa et sa famille.

Tables des matières

Table des matières

Chapitre II : Le SDN et la virtualisation

Chapitre III : Les réseaux programmables

Chapitre IV : Conception et réalisation

Chapitre I :

Chapitre II :

Chapitre III :

Chapitre IV :

Liste des tableaux

A

API : Application Programming Inerfaces

ARP : Address Resolution Protocol

AS : Autonomous System

ATM : Asynchronous Transfer Mode

B

BGP : Border Gateway Protocol

BS : Base Station

C

CE : Customer Edge

CPU : Central Processing Unit

CSDN : Customs Secure Data Network

CSPF : Constrained Shortest Path First

D

DLCI : Data Link Connection Identifier Protocol

E

EGP : Exterior Gateway Protocole

EIGRP : Enhanced Interior Gateway Routing Protocol

F

FIB : Enhanced Interior Gateway Routing Protocol

FEC : Forwarding Equivalence Class

FR : Frame Relay

H

HDLC : High Level Data Link Control

HAN: Home Area Network

I

IGP : Interior Gateway Protocol

IS-IS : Intermediate System to Intermediate System

ICMP : Internet Control Message Protocol

IBM : International Business Machines

IPv4/6 : internet Protocol Version 4/6

IDS : Intrusion Detection System

IP : Internet Protocol

IPS: Intrusion Protection System

IaaS : Infrastructure as a Service

L

LAN : Local Area Network

LER : Label Edge Router

LDP : Label Distribution Protocol

LFIB : Label Forwarding Information Base

LIB : Label Information Base

LSP : Label Switching Path

LSR : Label Switch Router

LLDP : Link Layer Discovery Protocol

LSR : Label Switch Router

M

MAC : Media Access Control

MIT : Massachusetts Institute of Technology

MP-BGP : Multi-Protocol Border Gateway Protocol

MPLS : Multi-Protocol Label Switching

N

NAT : Network address translation

NIC : Network interface card

NV : Network Virtualisation

NVE : Network Virtual Enveronement

O

ONF : Open Network Foundation

OS : Operating System

OSI : Open System Interconnection

OSPF : Open Shortest Path First

P

PE : Provider Edge

PPP : Point to Point Protocol

Q

QoS : Quality of Sservice

R

RIP : Routing Information Protocol

RSVP : Resources Reservation Protocol

S`

SCMP : Simple/Secure Commerce Messaging Protocol

SDN : Software Defined Networking

SDWN : Software Defined Wireless Network

SP : Service Provider

SSL : Secure Socket Layer

T

TE : Traffic Engineering

TCP : Transport Control Protocol

TDP : Transport Control Protocol

TOS : Type of Service

TLS : Transport Layer Security

TTL : Time To Live

U

UDP : User Datagram Protocol

V

VBS :

VM : Virtual Machin

VN : Virtual Network

VNIC : Virtual Network Interface Card

VoD : Video on Demand

VPLS : Virtual Private Local Service

VPN : Virtual Private network

VPI/ VPC : Virtual Path Identifier/Virtual Channel Identifier

VSDN : Video Software Defined Network

VXLAN : Virtual Extensible Local Area Network

Introduction

Durant ces dernières années, les réseaux informatiques classiques ont été mis en rude épreuve notamment par le développement sans cesse de nouvelles et diverses applications qui sont déployées sur le réseau. Bien que le déploiement de ces applications est devenu chose très aisée et facile grâce à l'apparition du Cloud, l'infrastructure qui est censée accueillir ces applications quant à elle semble être figée et n'ayant subi aucune évolution ou modification dans le temps et s'adapte difficilement à ces nouveaux services multimédias, ce qui la rend sujette à de nombreuses critiques et reste perçue comme un frein qui bloque le lancement de nouveaux concepts.

En effet l'émergence de nouveaux services multimédias tels que la VoD, ToIP et la IoT nécessite une nouvelle politique de gestion réseau que l'architecture actuelle ne peut garantir, cette nouvelle politique prévoit l'amélioration des paramètres déjà existant tels que la fiabilité, la disponibilité, le débit et la réduction de la latence…etc, d'une manière à faciliter au mieux le déploiement de ces différents services susmentionnés.

C'est dans ce contexte et pour répondre à ces problématiques que la technologie des réseaux définis par logiciel SDN est apparu qui, en collaborant avec la virtualisation permettra d'adapter l'infrastructure à la nature dynamique des applications multimédias, car le SDN apporte le principe de la programmation réseau qui permet de centraliser la logique déterminant le comportement du réseau vis à vis d'une application.

De ce fait définir une politique de gestion d'un réseau revient à écrire un programme qui sera déployé au travers d'une entité appelée contrôleur, mettant ainsi fin aux erreurs et aux incohérences induites par la configuration manuelle de chaque équipement.

Dans ce document, on a voulu répondre à cette problématique qui est le déploiement des services dans un environnement IP/MPLS de l'opérateur OTA-DJEZZY et cela en étudiant la méthode utilisée actuellement qui présente certaines limites qui freinent le développement des offres de l'opérateur pour ces clients et essayer de présenter une solution qui repousse ces limites. Pour ce faire, notre étude sera structurée en quatre chapitres qui sont les suivants :

Chapitre I : Sera consacré à l'étude du réseau IP/MPLS qui présente l'infrastructure de OTA-DJEZZY dans laquelle sera implémentée la solution SDN.

Chapitre II : Ce chapitre comportera une étude globale sur le concept de la virtualisation et des généralités sur le SDN.

Chapitre III : Dans ce chapitre nous ferons une étude détaillée sur la technologie adoptée par le SDN par laquelle il permet de rendre le réseau programmable et de ce fait contrôler son comportement.

Chapitre IV : Ce dernier chapitre sera dédié à la description de l'application Segment-Routing et de ses modules dans le système ONOS ainsi que les differents testes de fonctionnements.

Chapitre I : Les réseaux IP/MPLS

I.1 Préambule:

Les communications au sein d'un réseau informatique reposent essentiellement sur le processus de routage et les protocoles de routage constituent la pierre angulaire de ce dernier, car sans ces protocoles aucun échange ni envisageable ou possible

Cependant, avec l'expansion que connaît internet dans le monde du fait de l'émergence de services convergents à engendrer des problèmes liés à la performance du réseau menaçant ainsi sa stabilité et sa rentabilité ce qui à mener les chercheurs à développer des solutions allant d'une simple amélioration à un changement radical dans les technologies déjà existantes.

C'est donc, sur ces protocoles de routage que les chercheurs se sont penchés pour trouver une solution, puisqu' ils ont un impact direct sur les performances du réseau.

I.2 Communication dans un réseau IP :

I.2.1 Éléments d'un réseau : Un réseau IP est principalement constitué des ces équipements suivants [1] :

Le concentrateur : Appelé aussi hub, c'est un équipement réseau qui fonctionne au niveau1 (couche physique) du modèle OSI, utilisé dans la topologie en étoile, il permet notamment de récupérer les données binaires parvenant sur un port et de les diffuser sur l'ensemble des ports.

Le commutateur : C'est un élément de niveau 2 (couche liaison de données) qui s'occupe de l'acheminement des trames au sein d'un même réseau. Contrairement au hub qui diffuse l'information sur tous les ports, le switch sait déterminer sur quel port unique il doit envoyer une trame et cela en exploitant les adresses MAC de destination assignée à chaque port du commutateur.

Le routeur : Un routeur est un élément intermédiaire de niveau 3 du modèle OSI qui assurent l'acheminement des paquets entre réseaux distants en se basant sur des tables dites de routage construites par les routeurs.

Le terminal : Désigne un ensemble de périphériques placés à l'extrémité des nœuds soit pour demander ou pour offrir des services via le réseau (ordinateur, serveur ou logiciel…etc.).

I.2.2 Structure du réseau d'opérateur :

Afin de simplifier l'administration du réseau et isoler rapidement les problèmes qui peuvent survenir, les réseaux d'opérateurs doivent être organisés suivant une hiérarchie comportant trois couches : [2]

— **Couche d'accès (layer access) :** C'est la couche qui se situe à la périphérie du réseau, elle regroupe un ensemble de commutateurs qui sont physiquement connectés à ceux de la couche directement supérieure pour offrir un accès au réseau.

— **Couche distribution (layer distribution) :** Elle établit une liaison entre la couche d'accès et la couche cœur, elle fournit les fonctions de router les données entre VLANs et délimiter les zones de broadcast.

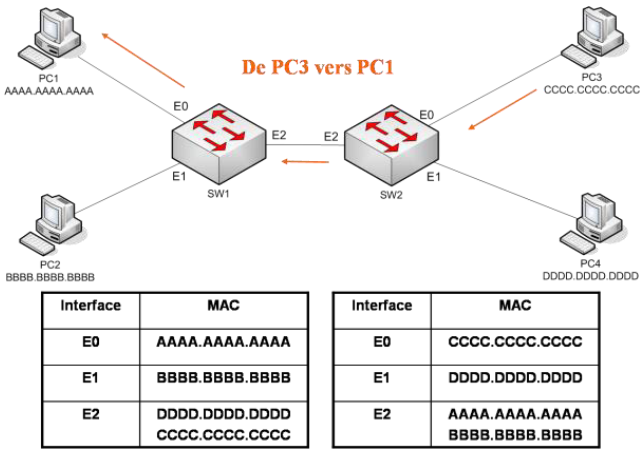
— **Couche cœur du réseau (layer backbone) :** C'est la couche fédératrice de tout le réseau et doit apporter une connexion à Internet en transférant les données le plus rapidement possible.

I.2.3 La commutation :

La commutation est la fonctionnalité qui permet d'acheminer les données d'un point A vers un point B dans un même réseau local (un LAN Ethernet par exemple). La commutation est effectuée par le commutateur en exploitant sa table MAC, cette dernière contient la correspondance entre l'adresse MAC d'un terminal et le port sur lequel il est connecté, cette table est remplie au fur et à mesure que les trames transitent par le switch. [3]

Fonctionnement : A la réception d'un paquet sur l'un de ses ports.

— Le commutateur désencapsule l'entête de la trame.
— Il vérifie si l'adresse MAC source et son port correspondant existent sur la table MAC.
Si la correspondance n'existe pas, il la crée.
Sinon il passe à l'étape suivante.
— En suite le commutateur cherche l'adresse MAC de destination dans sa table MAC.
Si l'adresse existe alors la trame est envoyée via le port correspondant.
Sinon la trame sera diffusée sur tous les ports à l'exception du port émetteur.



Interface	MAC
E0	AAAA.AAAA.AAAA
E1	BBBB.BBBB.BBBB
E2	DDDD.DDDD.DDDD CCCC.CCCC.CCCC

Interface	MAC
E0	CCCC.CCCC.CCCC
E1	DDDD.DDDD.DDDD
E2	AAAA.AAAA.AAAA BBBB.BBBB.BBBB

Figure I.1 : Exemple de commutation

Sur le commutateur, seul les adresses MAC sont sollicitées, les adresses IP ne sont pas exploitées autrement dit aucun routage n'est effectué.

I.2.4 Le routage :

Le routage est le mécanisme permettant d'établir des chemins entre une ou des sources vers une ou plusieurs destinations, cette fonction est remplie par le routeur, sur chaque interface du routeur un sous réseau y est connecté, cette interface représente aussi une passerelle pour les terminaux du même sous réseau, afin qu'ils puissent atteindre les destinations qui se trouvent sur d'autre sous réseaux.

Chaque routeur maintient une liste des sous réseaux avec en correspondance l'adresse IP du voisin qui permet d'atteindre le sous réseau, la liste en question est appelée la table de routage. [4]

I.2.5 La table de routage :

Une table de routage est un fichier de données dans la mémoire vive qui sert à stocker des informations concernant la route pour les réseaux connectés directement et les réseaux distants (tronçon suivant).

La table de routage contient des associations de réseau ou de tronçon suivant, menant à la destination finale.

Exemple :

Figure I.2 : Table de routage.

Champs	**Signification**
D	Indique comment la route à été découverte
10.1.1.0/24	Indique le réseau de destination
90	Indique la distance administrative (fiabilité) de la source ou de la route
2170112	Indique la distance à parcourir pour atteindre le réseau distant
209.165.200.226	Indique l'adresse IP du tronçon suivant pour atteindre le réseau distant
00 :00 :05	Indique le temps écoulé depuis la découverte de la route
S0/0/0	Indique l'interface de sortie du routeur permettant d'atteindre le réseau de destination

Tableau I.1 : Analyse d'une table de routage.

Fonctionnement :

Soit A et B deux machines se trouvant chacune dans un sous réseau différent et qui souhaitent communiquer :

— Le terminal A constate que son homologue B n'est pas sur son domaine de diffusion, alors il encapsule son paquet avec un entête de niveau 3 (adresse IP source de A et adresse IP de destination de B) ensuite avec un entête de niveau 2 qui cette fois contient l'adresse MAC source de A et l'adresse MAC de destination de B de la passerelle par default.

— Le commutateur achemine le paquet vers le routeur.

— A la réception du paquet le routeur décapsule l'entête IP de niveau 3 et en se basant sur l'adresse IP de B, il lance une recherche sur sa table de routage pour avoir l'adresse du prochain saut.

— Une fois la correspondance trouvée, le paquet est encapsulé dans un entête de niveau 3 (adresse IP source de A et adresse IP destination de B), ensuite un entête de couche 2 (adresse MAC source du routeur 1 et adresse MAC de destination du routeur 2).

— Les deux étapes se répètent jusqu'à atteindre le routeur relié au terminal B.

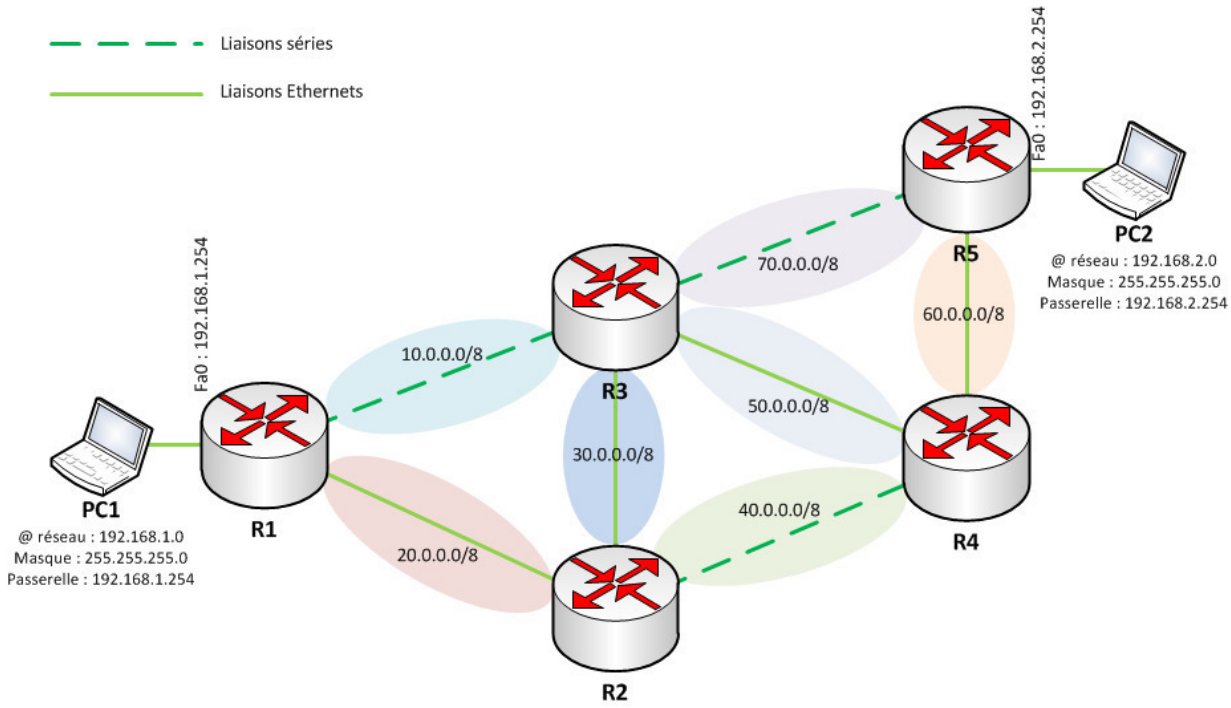


Figure I.3: Exemple de routage IP.

I.3 Type de routage : Un routeur peut apprendre des réseaux distants de deux manières distinctes :

I.3.1 Manuellement (Routage statique) : Un administrateur réseau peut configurer manuellement une route statique pour accéder à un réseau spécifique pour ce faire, l'administrateur réseau doit avoir une parfaite connaissance de la topologie de son réseau.

Ce type de routage apporte beaucoup d'inconvénient qu'il ne présente d'avantage, ce tableau en illustre quelques un :

Avantages	Inconvénients
• Les routes statiques ne sont pas annoncées sur le réseau, pour une meilleure sécurité. • Les routes statiques utilisent moins de bande passante que les protocoles de routage dynamique, aucun cycle de processeur n'est utilisé pour calculer et communiquer des routes. • Le chemin qu'une route statique utilise pour envoyer des données est connu.	• La configuration initiale et la maintenance prennent du temps. • la configuration présente des risques d'erreurs, tout particulièrement dans les grands réseaux. • l'intervention de l'administrateur est requise pour assurer la mise à jour des informations relatives aux routes. • n'évolue pas bien avec les réseaux en expansion et la maintenance devient fastidieuse.

Tableau I.2 : Avantages et inconvénients du routage statique.

I.3.2 Dynamiquement (routage dynamique) : Dans ce type de routage, c'est au routeur que revient la décision de déterminer la route par laquelle le paquet va transiter, afin d'arriver à destination. Pour cela une communication entre les équipements de couche 3 s'impose, cette communication consiste à l'échange et la mise à jour des tables de routage entre les différents nœuds du réseau, pour déterminer le chemin le plus adéquat pour l'acheminement des données, ce qui exclut toute intervention d'operateur humain.

I.4 Les protocoles de routage :

Ont pour rôle l'échange des informations de routes qui permettent la mise à jour dynamique des tables de routage, ces protocoles sont regroupés en deux catégories :

I.4.1 Les protocoles de routage internes (IGP) : Se sont des protocoles de routage interne utilisés dans systèmes autonomes (AS) pour permettre aux routeurs de communiquer entre eux, on distingue :

Routing Information Protocol (RIP) : Le RIP à été conçu pour travailler comme IGP dans un système autonome de taille moyenne, il appartient à la classe des protocoles à vecteur de distance autrement dit, c'est un protocole qui manipule des vecteurs de distance vers d'autres nœuds, la distance en question est le nombre de sauts permettant d'atteindre le nœud voisin.
Aujourd'hui, ce protocole est considéré comme désuet du fait qu'il est : [5]

— Limité dans le calcule du nombre de sauts (15 sauts au maximum).
— Utilise de la métrique fixe qui est susceptible d'introduire des instabilités dans le réseau.
— Susceptible de créer des boucles.
— Inapproprié pour être implémenté dans un environnement complexe.

Open Shortest Path First (OSPF) : C'est un protocole à état de lien qui est exécuté en interne dans un système autonome, le OSPF maintient dans sa base de données une vue complète de la topologie du système. De cette base de données, une table de routage est construite en calculant l'arborescence des chemins du système.

En outre, l'OSPF est sensible à la détection d'un quelconque changement topologique et cela en utilisant un minimum de trafic de protocole de routage, ce qui en fait un candidat idéal pour les réseaux évolutifs. [6]

I.4.2 Les protocoles de routage externes (EGP) : Ils forment un ensemble de protocoles qui permettent le routage inter-AS, le plus notable de ces protocoles est :

Border Gateway Protocol (BGP) : Le BGP est un protocole inter-AS, il est construit sur l'expérience acquise des EGP voisins, son principale objectif est d'échanger des informations de routage et d'accessibilité du réseau entre AS. La version quatre du protocole (BGP-4) est très utilisée sur internet et permet l'agrégation des routes, afin de limiter les tables de routage pour un meilleur acheminement des données. [7]

I.5 Pourquoi le MPLS ?

Les réseaux purement IP ne répondent plus aux critères de performances qui sont dictés par le développement des applications et des exigences des utilisateurs.

Car dans les réseaux distribués, chaque routeur fait suivre les paquets et effectue les trois étapes suivantes :

Étape 1 : Il désencapsule l'entête et la queue de bande de trame de couche 2 pour isoler le paquet de couche 3.

Étape 2 : Il examine l'adresse IP de destination du paquet IP pour trouver le meilleur chemin dans la table de routage.

Étape 3 : Si le routeur trouve un chemin vers la destination, il encapsule le paquet de couche 3 dans une nouvelle trame de couche 2 et transfère cette trame à l'interface de sortie.

Ces actions sont très gourmandes en temps CPU, ce qui augmente le temps de traitement des paquets dans le cœur d'un réseau, cette indépendance qui a été autre fois un avantage, est devenue aujourd'hui un inconvénient car à chaque saut le paquet va être désencapsulé et une recherche va se faire sur la table de routage d'où la nécessité de mettre en place le système de commutation des labels, autrement dit le MPLS.

Le protocole de routage MPLS *(Multi-Protocol Label Switching)* a été crée pour répondre aux limites des protocoles de routage purement IP, sa popularité il la doit à sa capacité à être implémenté sur une structure physique déjà existante, autrement dit, nul besoin d'une nouvelle infrastructure physique dédié, autre caractéristique est non la moindre et le transfert des paquets en utilisant une simple commutation de labels et non un routage IP.

I.6 Multi-protocol Label Switching (MPLS):

Le MPLS est un mécanisme de transfert de données basé sur la commutation de labels (étiquettes) qui sont insérés à l'entrée du réseau MPLS et retirés à sa sortie, cette insertion s'opère entre la couche de liaison de données (niveau 2) et le couche réseau (niveau 3), ce qui lui à valu d'être qualifié de protocole de niveau 2,5. [3]

.

I.6.1 Eléments du MPLS : Dans les réseaux MPLS, Le transfert de données peut être effectué par des routeurs ou des commutateurs capables de faire la recherche d'étiquettes et leurs remplacements, mais ne sont pas capables d'analyser les en-têtes de la couche réseau.

Les principaux équipements utilisés dans une architecture MPLS sont LSR *(Label Switch Router)* et LER *(Label Edge Router).*

1. LSR (Label Switcher Router) : Le LSR est un équipement de type routeur, ou commutateur qui appartient à un domaine MPLS dont les fonctions sont :

— L'échange d'information de réseau.

— L'échange de label.

— L'acheminement des paquets.

2. LER (Label Edge Router) : LER est un LSR qui fait l'interface entre un domaine MPLS et le monde extérieur. En général, une partie de ses interfaces supportent le protocole MPLS et l'autre un protocole de type IP. Les deux types de LER qui existent sont :

- Ingress LER : Est un routeur qui gère le trafic qui entre dans un réseau MPLS.
- Egress LER : Est un routeur qui gère le trafic qui sort d'un réseau MPLS.

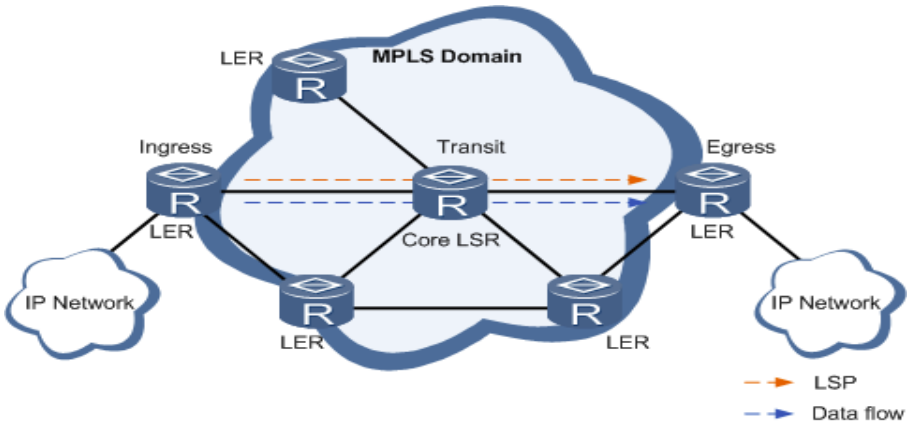


Figure I.4 : Structure du réseau MPLS.

I.6.2 Terminologie :

- **LSP (Label Switch Path) :** Est un chemin unidirectionnel de bout en bout qui traverse le LSR dans un domaine MPLS sans aucune modification du trafic.

- **Label :** Est une étiquette qui se rajoute à un paquet par un LER lorsqu'il rentre dans un domaine MPLS, c'est grâce à cette étiquette que le LSR peut commuter les paquets.

- **LIB (Label Information Base) :** Est une base d'informations qui sert à déterminer quel label utiliser pour atteindre un LSR.

- **LFIB (Label Forwarding Information Base) :** Est une table de commutation sur laquelle se base un LSR pour savoir vers quelle sortie commuter un paquet en entrée et avec quel label, elle contient les champs suivants : (Label-Opération-Prochain saut).

- **FIB (Forwarding Information Base) :** Est la table qui contient pour chaque ligne les champs (réseau-prochain saut-label), elle est utilisée par le LER pour affecter les labels aux paquets.

- **FEC (Forwarding Equivalence Classes) :** Est un ensemble d'adresse IP ayant le même préfixe d'adresse ou la même classe de service auxquelles le MPLS associe le même label.

I.6.3 Structure de données des labels :

Les routeurs du domaine MPLS construisent trois bases d'informations FIB, LFIB et LIB, ces bases d'informations sont construites selon plusieurs étapes :

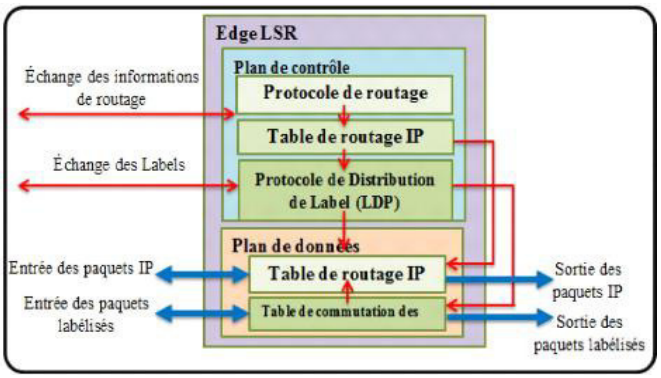


Figure I.5 : Structures de données des labels dans une architecture LSR

— Les protocoles de routage (OSPF, IS-IS ou EIGRP) construisent les tables de routages.

— Chaque LSR alloue indépendamment un label à chaque destination dans sa table de routage.

— Les labels alloués sont enregistrés dans la LIB.

— Ces labels et leurs prochains sauts ont enregistrés dans la table LFIB avec l'action à effectuer.

— Le LSR envoie les informations sur sa LIB à ces voisins.

— Chaque LSR enregistre les informations reçues dans sa LIB.

— Les informations reçues des prochains sauts sont enregistrées dans la FIB.

— Chaque LSR construit ses propres structures FIB, LFIB et LIB.

I.6.4 Fonctionnement de MPLS :

La mis en ouvre de MPLS repose sur la détermination de la caractéristique commune à un ensemble de paquets et dont dépondra de l'acheminement de ces derniers, cette notion de caractéristique commune est appelée FEC *(Forwarding Equivalence Class).* Une FEC est donc la représentation d'un ensemble de paquets qui sont traités de la même manière, qui suivent le même chemin au sein du réseau et ayant la même priorité.

Lorsqu'un paquet IP arrive à un Ingress LER, il sera associé à un FEC puis, exactement comme dans le cas d'un routage IP classique, un protocole de routage sera mis en œuvre pour déterminer le chemin jusqu'à l'Egress LER (voir Figure I.6), mais à la différence d'un routage IP classique cette opération ne se réalise qu'une seule fois, ensuite tout les

paquets qui appartiennent à un même FEC serons acheminés suivant ce chemin qu'on appellera (LSP).

Un LSP est le chemin établi au travers d'un ou plusieurs LSR pour rejoindre plusieurs LER au sein d'un réseau MPLS, quand un trafic est envoyé sur un LSP, il passe par un tunnel qui lui permettra d'atteindre la sortie du LSP sans altération ou modification. L'idée est de n'examiner aucun autre en-tête que l'en-tête de label.

Ainsi on a eu la séparation entre la fonction de routage et la fonction de commutation : le routage se fait uniquement pour le premier paquet, ensuit une commutation simple qui s'appliquera à tous les paquets appartenant au même FEC.

Pour que les LSR puissent commuter correctement les paquets il doit procéder comme suit :

— Le Ingress LER (device B) récupère le trafic des utilisateurs ayant la même adresse IP de destination (FEC) et leur attribue des labels avant de les envoyer vers l'interface de sortie (GE1/0/1) cette méthode appelé PUSH. Ensuite le paquet arrive au LSR, celui ci consulte le label du paquet et le cherche dans sa LFIB pour déterminer vers quelle sortie commuter le paquet.

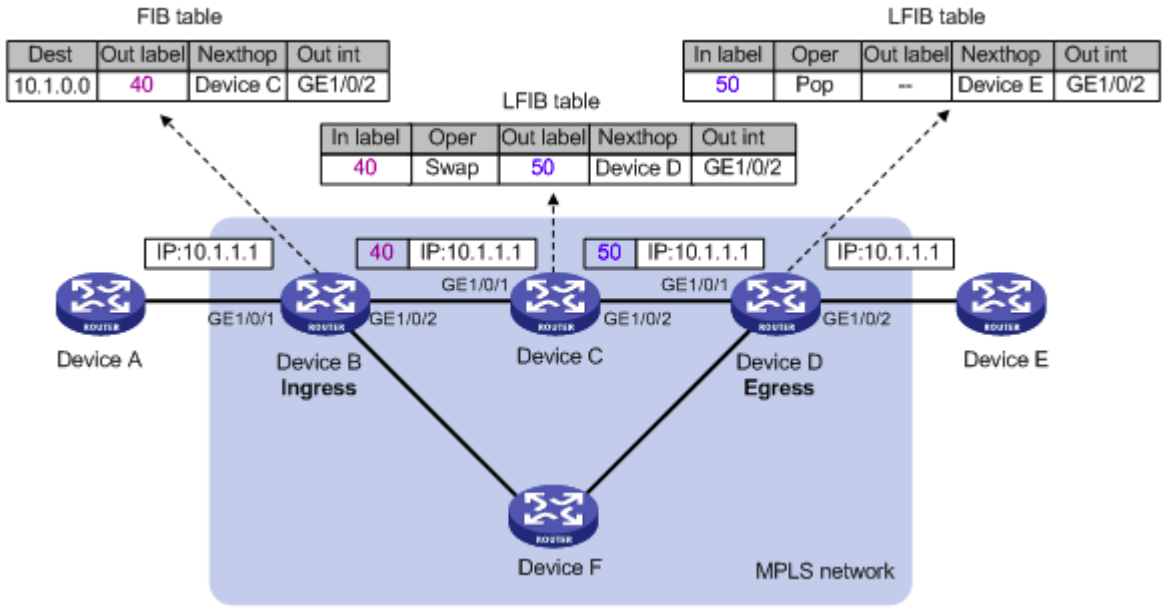


Figure I.6 : Exemple de commutation de labels dans un domaine MPLS.

— Les paquets arrivent au LSR (device C) ce dernier lui applique la méthode SWAP qui consiste à remplacer le label d'entrée (40) par le label de sortie (50) et consulte sa table LFIB pour déterminer vers quelle sortie les commuter, cette sortie n'est qu'autre que l'interface G1/0/2 qui relie le device C avec le device D.

— L'étape de commutation sera exécutée par tous les LSR du LSP jusqu'à atteindre le LER de sortie en l'occurrence le device D.

— A l'arrivée du trafic au device D, ce dernier consulte sa table LFIB et constate que l'opération appropriée est le POP qui consiste en la suppression du label est par conséquent le début d'un routage classique.

L'acheminement des paquets dans le domaine MPLS ne se fait donc pas à bas d'adresse IP, mais de labels (commutation de labels).

I.6.5 Architecture du MPLS : L'architecture logique du MPLS est conçue de telle manière à supporter plusieurs protocoles de niveau 2 (Ethernet ou ATM) ainsi que ceux de niveau 3 (OSPF, RIP et BGP…), cette architecture est constituée de deux plans séparés : [8]

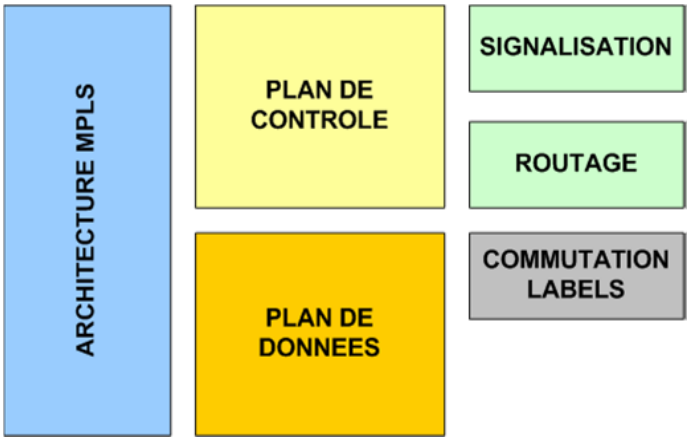


Figure I.7 : Architecture du MPLS.

- **Plan de contrôle :** Est composé par l'ensemble des protocoles de signalisation et de routage qui sont complémentaires au LDP. Le plan de contrôle est responsable de la construction et de la maintenance des tables d'acheminement (Forwarding Tables) et

permet la communication inter-nœuds (LSR) afin de disséminer les informations concernant le routage.

- **Plan de données :** Il est responsable de transporter les paquets labélisés qui sont commutés dans un domaine MPLS en se basant sur les " Forwarding Tables".

I.6.6 Format d'un label : Un label est un indice codé sur 32 bits inséré par le LER et identifie le chemin que le paquet doit suivre dès son entrée dans un nuage MPLS. Le label est directement encapsulé et transporté dans le paquet, pour être inséré entre l'entête de niveau 2 (adresses MAC) et celui de niveau 3 (adresses IP), chaque paquet devra suivre un voyage qui sera basé sur une commutation de labels dès qu'il reçoit son étiquette. [9]

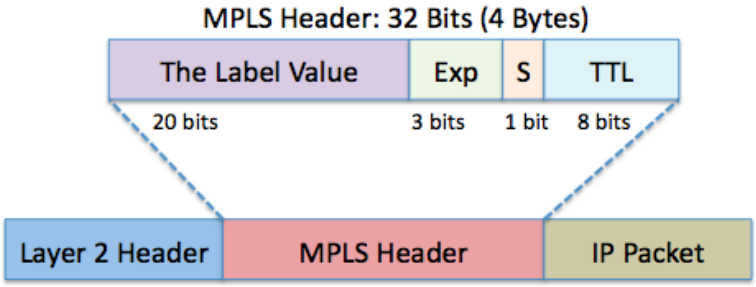


Figure I.8 : Format générique d'une étiquette MPLS.

LABEL : Sur 20 bits, est le code binaire du label qui va transiter dans le domaine MPLS, cette valeur numérique attribuée au label correspond au Forwarding Equivalence Class (FEC).

EXP : Sur 3 bits, est un champ expérimental, n'est pas normalisé et peut être utilisé pour gérer la qualité de service afin de coder le numéro de classe de service.

STACK : Sur 1 bit, sert à spécifier si le label est le dernier de la liste. Car, il est possible d'empiler les labels au sein d'un même paquet. Par conséquent, il faut indiquer au routeur s'il y a d'autres labels à lire. Dans le cas courant d'un seul label, le bit sera positionné à 1.

TTL : Sur 8 bits, comme pour l'IP, le TTL *(Time To Live)* représente la durée de vie d'un paquet dans un nuage MPLS, Le TTL permet de prévenir la formation de boucles dans le réseau.

Selon le type de protocole de niveau 2 véhiculés par le paquet, les labels sont implémentés différemment :

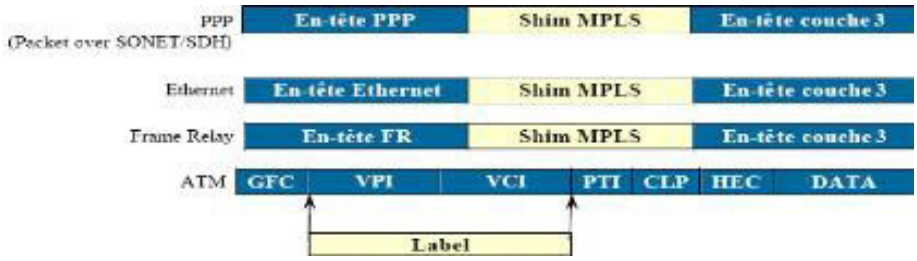


Figure I.9 : Encapsulation des labels dans différentes trames.

— Pour ATM, le label n'est pas inséré entre l'entête de niveau 2 et de niveau 3, mais directement dans le champ VPI/VCI de la trame ATM.

— Pour FRAME RELAY, le principe reste le même que dans ATM, le label est aussi inséré dans le champ DLCI de l'entête FR.

— Pour les protocoles Ethernet, HDLC (High Data Link Control) et PPP (Point To Point Protocole), et contrairement aux protocoles ATM et FR, le label occupe un champ à lui tout seul, qui est ajouté entre l'entête de niveau 2 et celui de niveau 3 des trames susmentionnées.

I.6.7 Les protocoles de signalisation : La signalisation au sein d'un environnement IP/MPLS fait référence à l'ensemble des échanges d'informations entre les LSR, donc un protocole de signalisation sert à informer d'autres LSR sur le routage et les étiquettes associées aux paquets entrant dans le domaine MPLS. Parmi les protocoles de signalisation on trouve :

— TDP/LDP (Tag/Label Description protocole) : Pour le mapping des adresses unicast**.**
— CR-LDP, RSVP-TE : Utilisés en trafic Engenering pour établir des LSP en fonction de critère de ressources.
— MP-BGP (Multi Protocoles Border Gateway Protocole) : Pour l'échange des routes VPN.

1.6.8 Distribution des labels :

La distribution des labels au niveau des routeurs fait partie du plan de contrôle et plus particulièrement de la signalisation, cette attribution des labels est réalisée par des protocoles spécialisés tels que le LDP *(Label Distribution Protocol)* ou le RSVP-TE *(Ressource Reservation Protocol with Traffic Engineering)*, les labels peuvent être distribués suivant deux approches : [10]

- **Statiquement :** Cette approche consiste en l'intervention de l'administrateur responsable du réseau, tous les labels seront affectés par celui ci d'une manière manuelle, cette approche n'est plus utilisée de nos jours, du fait qu'elle nécessite une intervention humaine à chaque fois qu'une modification ou une évolution est apportée au réseau, cela rend la tâche de l'administrateur réseau très fastidieuse à cause du volume des réseaux actuels, ce qui induit à des erreurs de configurations, sauf dans certain cas comme par exemple où il est nécessaire de forcer le trafic sur un certain LSP.

- **Dynamiquement :** Cette seconde approche est la plus utilisée actuellement, elle est basée sur l'exécution d'un algorithme au niveau du plan de contrôle sur chacun des routeurs du domaine MPLS, cet algorithme se charge de la création et l'affectation des labels et de se fait la création des LSP. L'utilisation de cette méthode dynamique est favorisée à cause de la flexibilité qu'offre cet algorithme, puisqu'il intègre automatiquement les modifications apportées sur le réseau et il reste relativement performent lors du passage à l'échelle, sans oublier sa capacité à réagir en cas de détection de pannes. Cette approche est bien sûr très appréciée par l'administrateur réseau car elle réduit considérablement leurs travails de configuration manuelle et ça leur offre beaucoup plus de temps pour réfléchir au développement et à l'optimisation du réseau.

Deux modes de distribution des labels peuvent coexister au sein d'un seul et unique domaine MPLS, ces modes sont les suivants :

- **Downstream on demande :** Traduit en français "vers l'aval à la demande ", c'est un mode qualifié de réactif, car dès qu'un routeur upstream (routeur en amont à la source

de données) aura découvert une nouvelle FEC, il va établir une connexion avec le routeur downstream (routeur en aval par apport à la source de données) pour lui demander un label correspondant à la FEC découverte.

Figure I.10 : Principe du Downstream on demande.

- **Unsolicited downstream :** Traduit en français "vers l'aval non sollicité ", ce mode est qualifié de proactif, car dés que le routeur en aval aura découvert une nouvelle FEC et mis à jour ses labels, il va en informer le routeur upstream pour que ce dernier met à jour sa table de commutation.

Figure I.11 : Principe du Downstream unsolicited.

Enfin deux politiques peuvent être utilisées pour la rétention des labels par les routeurs du réseau MPLS :

- **Conservative :** Le LSR conserve uniquement les labels du prochain saut valide qui sont associés à une FEC, si le prochain saut vient à changer, le LSR doit effectuer une LABEL REQUEST pour demander un nouveau label. L'avantage de cette méthode est qu'elle est économique en espace mémoire et très utilisée dans le mode Downstream on demande.

- **Libérale :** Par opposition à la première politique, celle-ci est très gourmande en espace mémoire du fait qu'elle conserve tous les labels annoncés par le routeur downstream, ce qui lui permet à un LSR d'effectuer un "fast rerout " si le prochain saut d'une FEC vient à changer.

I.6.9 Label distribution protocole : Le protocole LDP est un protocole de signalisation et plus précisément de distribution des labels, hérité du protocole propriétaire TDP (Tag Distribution Protocol), il a été à l'origine publié comme [RFC3036] en janvier 2001 par le groupe de travail MPLS de l'IETF.

C'est l'ensemble des procédures et des messages par les quels les LSRs établissent des LSPs à travers un réseau en transportant les mécanismes et les informations d'acheminement de couche réseau directement en mécanisme de commutation de couche de liaison de données Il est basé sur le protocole TCP pour la fiabilité sauf que les messages de découverte sont envoyés sur le UDP.

I.6.9.1 Les types de messages utilisés par LDP :

— **Messages de découvertes :** annoncent et maintiennent la présence d'un LSR dans le réseau.

— **Messages de session :** établissent, maintiennent et termines les sessions LDP.

— **Messages d'avertissement :** Créent, changent et suppriment des associassions entre FEC et Label.

— **Messages de notification :** pour fournir des informations supplémentaires ou signaler une erreur.

I.6.9.2 Distribution des Label par LDP :

Chaque LSP transporte une ou plusieurs classes de flux, cette classe peut être définie en fonction de plusieurs paramètres qui relève généralement de la QoS (Qualité de services).

Les paquets ayant la même classe sont associés au même FEC, mais dans le cas où la QoS n'est pas spécifiée et lorsque un LSP est partagé par plusieurs éléments de FEC, ce LSP se termine au nœud (ou avant) où les éléments de FEC ne peuvent plus partager le même chemin. Pour cela le LDP vient pour spécifier précisément les paquets pouvant être transposés sur chaque LSP, en fournissant une spécification de numéro de FEC dans chaque LSP.

Le protocole LDP supporte les deux modes de distribution des labels qui partagent la phase de découverte et d'établissement d'une session LDP/TDP :

— Un identifiant LDP (6 octets) est utilisé pour identifier un espace des Labels de LSR, auquel chaque routeur MPLS possède un identifiant unique sur 4 octets et les deux autres sont utilisés pour identifier un espace de Label spécifique dans LSR.

— Un LSR envoie périodiquement en multicast le message 'HELLO' en UDP sur le port 646 qui contiennent l'ID du LSR, à la réception les LSR établissent une relation de voisinage répondent par le même message 'HELLO' pour confirmer la présence, c'est le déclanchement d'une session LDP.

— après l'établissement de voisinage une connexion fiable TCP est mise en œuvre entre les deux homologues.

C'est à partir de cette étape que le déroulement des deux modes diverge, ou le mode "downstream en demande", un simple message du LSR upstrem "LABEL REQUEST" est envoyé vers le LSR downstream, ce dernier lui répond avec un message "LABEL MAPPING" contenant les informations souhaitées, ensuite le LSR upstream décide de garder ou non le label en fonction de son mode de rétention de label.

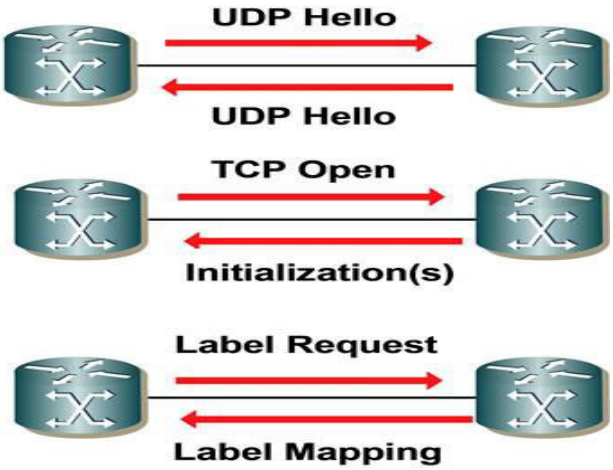


Figure I.12 : Session LDP en mode Dowenstream on demand.

Pour le mode "Unsolicited downstream" dés l'établissement d'une session LDP, le LSR diffuse en multicast son ‘’LABEL MAPPING’’ vers tout ses voisin (ceux avec qu'il possède une session LDP).

I.7 Applications du MPLS :

L'un des principaux atouts de l'MPLS est sa capacité à intégrer de nouvelles fonctionnalités de gestion de qualité de services et d'ingénierie du trafic pour répondre aux exigences des clients

I.7.1 Ingénierie du trafic (TE) : Désigne l'ensemble des techniques qui permettent d'optimiser l'utilisation des ressources d'un réseau, en prenant en compte la bande passante disponible sur un lien lors du routage, afin d'éviter la congestion. Cela est possible grâce à un algorithme particulier qui est le CSPF *(Constraint Shortest Path First),* car ce dernier permet de reconstruire la topologie du réseau selon la contrainte de la bande passante et de ce fait minimiser au maximum la congestion et optimiser le trafic. [11]
Cette algorithme à été implémenté par de nombreux protocoles déjà existants comme :

- De routage : Comme ISIS et OSPF pour évoluer vers ISIS-TE et OSPF-TE.
- De signalisation (réservation de bande passante sur un routeur) : RSVP pour évoluer vers RSVP-TE.

I.7.2 Qualité de services (QoS) :

Tout comme le Traffic Engineering, la qualité de service est un élément crucial pour un réseau d'opérateur. En MPLS la qualité de services peut être fournie par deux approches :[12]

- **IntServ :**

  L'approche "integration services" s'appuie sur la réservation de ressources (RSVP) pour classer les flux un par un dans des files d'attente séparées, chaque flux se voit assigné une classe de service, ce processus compte beaucoup d'inconvénients :

  — Coûteux en ressources machine.

  — Supporte difficilement la montée en charge des utilisateurs.

— Il engendre des listes de classe de services très longues à maintenir par le routeur.
— Plus les flux seront nombreux plus il sera difficile au routeur d'assigner le flux à la classe de service qu'il lui correspond.

- **DiffServ :** Cette approche qui est la "différenciation de services" à pour principe de base de fournir une qualité de service à des agrégats de flux plutôt qu'à des flux de données individuelles, ceci est possible grâce au marquage des paquets IP. Si cette approche réduit considérablement le nombre de classes de services, elle ne reste néanmoins pas adaptée aux réseaux à grande densité de trafic, puisqu'elle attribue la même classe de services à des flux qui nécessitent des traitements différents de la part des routeurs.

I.8 Evolution du MPLS :

L'évolution du MPLS ne c'est pas arrêtée à proposer des reformes dans la qualité de service et dans le trafic réseau, le multi-protocole à voulu présenter de nouveaux concepts basés sur le tunneling pour offrir des services virtuels sur son infrastructure :

I.8.1 LSP Tunneling :

Le MPLS est un protocole qui offre la possibilité d'empiler plusieurs labels dans un seul et unique paquet. Ce principe nommé "Label Stacking" est utilisé pour créer des Tunnel LSP. Le LSP Tunneling est une composante importante de la technologie VPLS (Virtuel Privat Local Service) qui permet d'offrir aux clients des services virtuels tels que les VPN, le LSP Tunneling est souvent mis en œuvre pour agréger plusieurs LSP dans un seul, comme illustré dans la figure ci-dessous. [11]

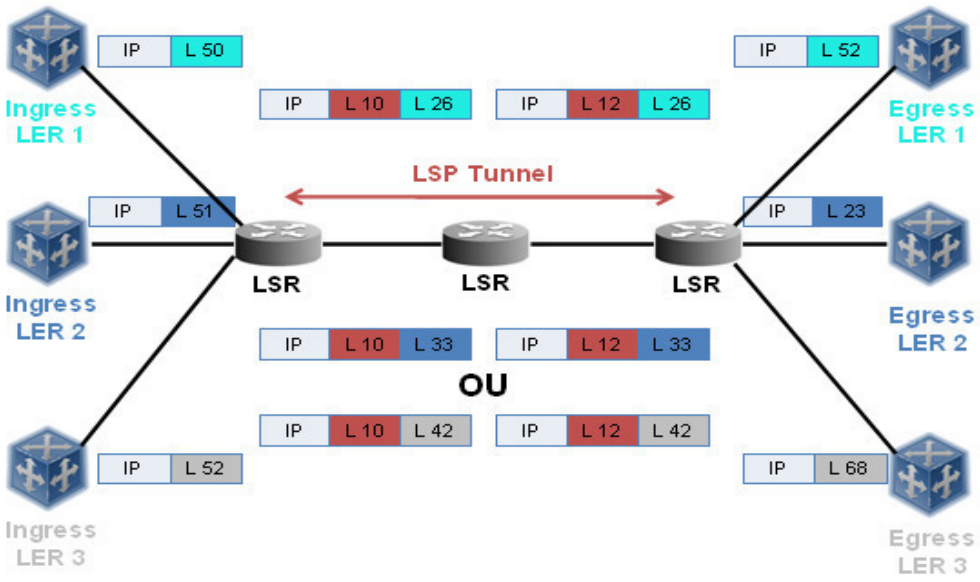


Figure I.13 : Processus du LSP tunneling.

On remarque que ces 3 LSP traversent les mêmes LSR dans le cœur du réseau. Pour agréger ces 3 LSP dans un seul au sein du backbone, un Tunnel LSP est mis en œuvre, et les différents labels associés à ce Tunnel LSP sont marqué en rouge. Ainsi, dans le cœur du réseau, le LSR du milieu ne connait qu'un seul LSP : le LSP rouge, qui encapsule tous les autres.

Dans un vrai réseau, on pourrait imaginer que plusieurs LSR du backbone soient traversé par le Tunnel LSP.

I.8.2 La technologie VPLS :

Le VPLS est une technologie très répondue dans l'MPLS et utilisée par un opérateur afin de proposer à ces clients des offres VPN de niveau 2, ce qui permet à un client de créer la structure logique d'un réseau local (LAN) entre des sites distants, où tous les services paraissent ainsi appartenir à un même LAN. Ces services s'appuient sur le protocole MP-BGP pour l'établissement des routes entre clients, le réseau fournisseur évite alors le besoin potentiel d'ouvrir plusieurs sessions LDP. [11]

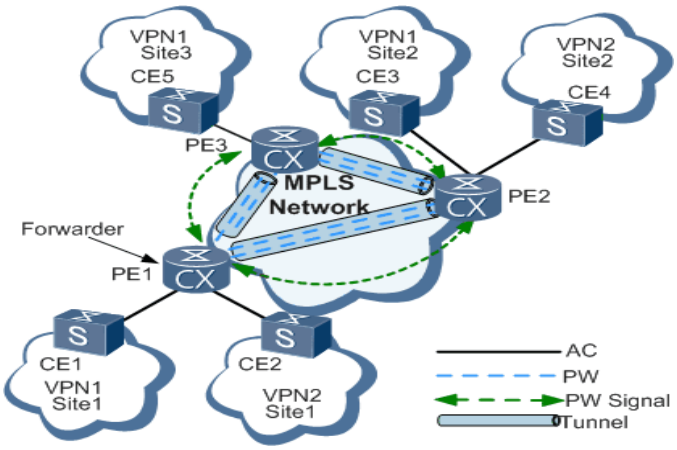


Figure I.14 : Service VPLS entre VPN.

L'offre de services VPLS repose essentiellement sur le LSP tunneling pour l'acheminement des trames entre tous les PE qui interconnectent les CE d'un réseau, ce qui permet de créer une topologie maillée au sein du réseau. La communication entre CEs est rendue possible grâce à la norme de commutation de couche 2, car le VPLS crée un commutateur Ethernet virtuel coté prestataire de services où chaque PE est considéré comme un port du switch virtuel. De ce fait, chaque PE possède une table MAC afin d'avoir une correspondance entre l'adresse MAC d'un CE et son PE de raccordement.

I.9 Discusions :

Les réseaux IP/MPLS ont un bel avenir devant eux puisqu'ils sont présents dans tous les réseaux d'opérateurs. En effet, ces réseaux permettent de répondre aux limites des protocoles de routage purement IP en mettant en place le principe de la commutation des labels, ces réseaux sont aussi une solution prometteuse parce qu'ils permettent d'intégrer très facilement de nouvelles technologies basées sur la virtualisation des services réseau telle que le VPLS, pour proposer des offres virtuels aux clients. La virtualisation constitue donc un concept incontournable pour révolutionner les réseaux d'opérateurs, c'est donc sur ce concept que se portera notre deuxième chapitre.

Chapitre II : La virtualisation et le SDN

II.1 Préambule :

Au cours des dernières années, le SDN (réseau définis par logiciel) et la virtualisation des réseaux, connaissent une attention toute particulière de la part des spécialistes réseaux. Mais cependant une certaine confusion règne à propos de ces technologies, en effet plusieurs fournisseurs proposent des solutions qui résolvent différents problèmes en se basant sur des architectures différentes, et chacun d'eux prétend avoir la solution ultime pour le SDN et la virtualisation des réseaux.

Le SDN et la virtualisation des réseaux vont avoir un impact bouleversant vont avoir un impact bouleversant dans le domaine du réseau, et imposeront une nouvelle vision de ce dernier, avec ces technologies un déploiement d'un réseau au sein d'une entreprise ne prendra que quelques minutes ou heurs contrairement à un déploiement classique qui exigera une intervention plus lourde et beaucoup de temps qui peut être évalué à des mois

Dans ce chapitre on va introduire le concept de la virtualisation, et on se focalisera sur la virtualisation des réseaux, en suite on entamera des généralités sur le SDN

II.2 C'est quoi la virtualisation :

La virtualisation consiste à créer une version virtuelle d'un dispositif ou d'une ressource comme un système d'exploitation, un serveur, un dispositif de stockage ou une ressource réseau. Nous pouvons donc considérer la virtualisation comme une abstraction physique des ressources informatique.

Chaque dispositif ou ressource virtuelle correspond donc à un dispositif ou ressource physique sur un système informatique physique, les machines virtuelles hébergées par un ordinateur hôte sont donc perçues par ce dernier comme des applications auxquelles il est nécessaire de dédier ou distribuer ces ressources. [13]

II.2.1 La machine virtuelle (VM):

Une machine virtuelle est un fichier informatique généralement appelé image, qui se comporte comme un ordinateur réel. D'autres parts la machine virtuelle est un conteneur de logiciels totalement isolé et doté d'un système d'exploitation et d'application propres. Chaque VM est une entité autonome. Complètement indépendante. L'installation de plusieurs VM sur un ordinateur permet d'exécuter différents systèmes d'exploitation et applications sur ce dernier. [14]

II.2.2 Virtual local area network (VLAN):

Les VLAN sont des réseaux locaux qui reposent sur des connexions logiques et non physiques, Ils permettent la segmentation, assouplissent et l'organisation du réseau.

Les VLAN offrent un moyen de regrouper des périphériques dans un LAN. Un groupe de périphériques dans un VLAN communiquent comme s'ils étaient reliés au même câble.

Les VLAN permettent à un administrateur de segmenter les réseaux en fonction de facteurs tels que la fonction, l'équipe de projet ou l'application, quelque soit l'emplacement physique de l'utilisateur ou du périphérique. Les périphériques d'un VLAN se comportent comme s'ils se trouvaient chacun sur leur propre réseau indépendant, même s'ils partagent une infrastructure commune avec d'autres VLAN. Pour assure une communication inter-VLAN un équipement de niveau 3 est toujours nécessaire. [15]

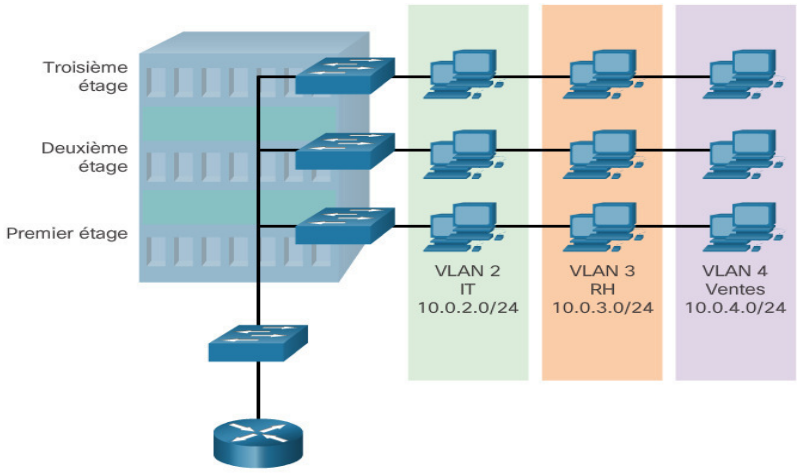


Figure II.1 : Exemple de VLAN.

II.2.3 Le réseau privé virtuel (VPN):

Le VPN est une technologie réseau permettant de construire un réseau privé (LAN) au sein d'une infrastructure publique telle que le réseau mondiale Internet. Les échanges entre les utilisateurs d'un même VPN seront totalement invisibles et confidentiels aux autres utilisateurs. Au moyen d'un VPN, un télétravailleur peut accéder à distance au réseau de sa société via internet, cela se fait à travers un tunnel sécurisé. Cette technologie est très utilisée sur les réseaux à commutation de labels MPLS. [16]

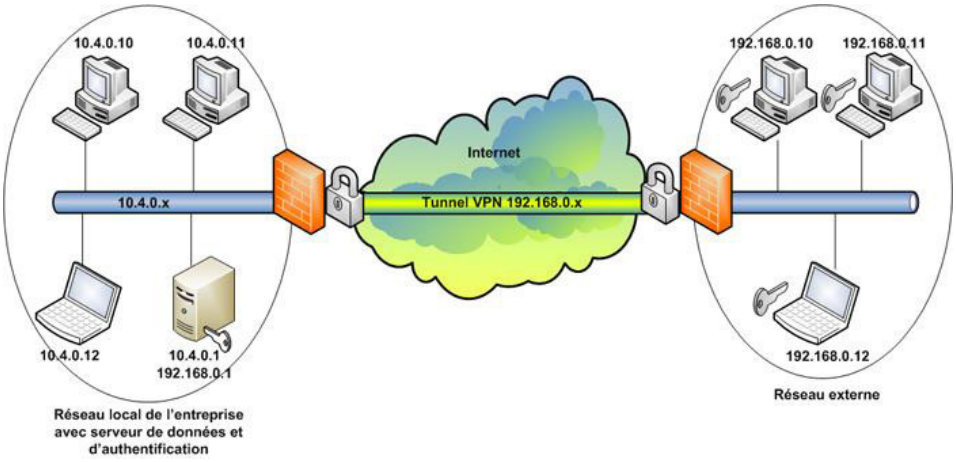


Figure II.2 : Exemple de VPN.

Ce concept comme le VLAN vient rejoindre l'esprit de la vitalisation au sein des réseaux mais qui ne répond pas aux exigences et tendances des moments, car elle nécessite toujours une intervention sur le matériel dans le cas d'une modification de la topologie et de la configuration.

II.3 La virtualisation réseau (NV) :

Elle est présentée comme l'abstraction d'un réseau vis-à-vis de l'équipement physique sous-jacent. La virtualisation de réseau permet à plusieurs réseaux virtuels de partager la même infrastructure physique, et chaque réseau virtuel peut avoir une topologie beaucoup plus simple (abstraite) que celle du réseau physique. [17]

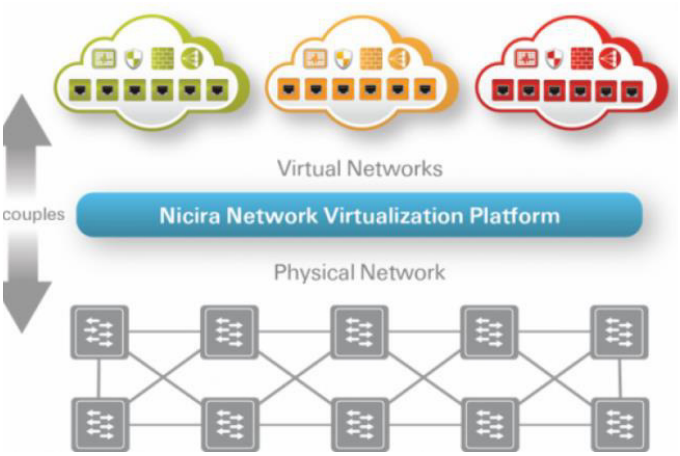


Figure II.3 : Virtualisation réseau selon Nicira.

Le résultat de la virtualisation du réseau est un réseau virtuel. Les réseaux virtuels sont classés en trois grandes catégories :

- **Réseau virtuel externe :** Les réseaux virtuels externes sont composés de plusieurs réseaux locaux administrés par le logiciel comme une entité unique. Les blocs de construction des réseaux virtuels externes standard sont le matériel de commutation et la technologie logicielle VLAN. Les réseaux virtuels externes comprennent par exemple les grands réseaux d'entreprise et les centres de données.

- **Réseau virtuel interne :** Se compose d'un système utilisant des machines virtuelles ou des zones dont les interfaces réseau sont configurées sur au moins une NIC physique. Ces interfaces réseau sont appelées cartes d'interface réseau virtuelles ou NIC virtuelles (VNIC). Ces conteneurs peuvent communiquer les uns avec les autres comme s'ils étaient sur le même réseau local, en devenant un réseau virtuel sur un seul hôte.

- **Réseau virtuel privé :** Est un type spécial de réseau virtuel interne. Les réseaux virtuels privés sont différents des réseaux privés virtuels (VPN). Un logiciel VPN crée une liaison point à point sécurisée entre deux systèmes d'extrémité. Le réseau virtuel privé est un réseau virtuel sur un système qui n'est pas accessible par les systèmes externes. L'isolation de ce réseau interne des autres systèmes externes est obtenue en configurant des VNIC sur des etherstubs.

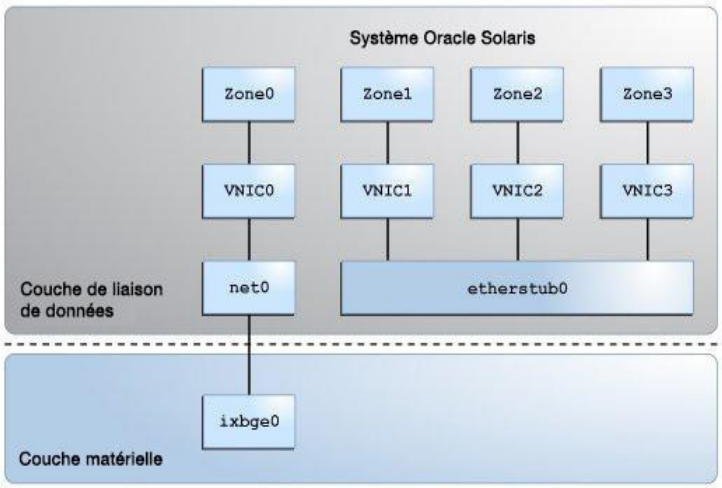


Figure II.4 : Illustration de réseau virtuel privé.

II.3.1 Les composants de la virtualisation réseau : La virtualisation du réseau doit comporter quatre éléments de base : [18]

- **Les VNIC :** Sont des périphériques réseau virtuels avec les mêmes interfaces de liaison de données qu'une NIC physique. Les VNIC sont configurées sur une liaison de données sous-jacente. Lorsque les VNIC sont configurées, elles se comportent comme des cartes d'interface réseau physiques. En outre, les ressources du système traitent les VNIC comme s'il s'agissait de cartes d'interface réseau physiques. Une VNIC génère automatiquement l'adresse MAC. Selon l'interface réseau en cours d'utilisation, on peut explicitement attribuer à une VNIC une adresse MAC autre que l'adresse par défaut.

- **Commutateur virtuel :** Lorsque une VNIC créée, un commutateur virtuel est créé automatiquement. Conformément à la conception Ethernet, les commutateurs virtuels fournissent aux zones une méthode de transmission de paquets. Le commutateur virtuel ouvre un chemin de données pour que les réseaux virtuels communiquent les uns avec les autres.

- **Etherstub :** Est un pseudo NIC Ethernet configurée au niveau de la couche de liaison de données de la pile réseau. Il permet de créer des VNIC sur des etherstubs plutôt que sur une NIC physique. Les etherstubs permettent de construire un réseau virtuel privé qui est isolé des autres réseaux virtuels sur le système et du réseau externe. Par exemple, des etherstubs peuvent servir à créer un environnement réseau dont l'accès est uniquement limité aux développeurs et de ne pas à votre tout le réseau**.**

- **Zone :** Une zone est un environnement de système d'exploitation virtualisé, créé au sein d'une instance unique du système d'exploitation. Les zones offrent un environnement isolé et protégé pour les applications en cours d'exécution.

NB : Les etherstubs et les VNIC ne sont qu'une partie des fonctionnalités de virtualisation. En affectant des VNIC ou des etherstubs pour une utilisation par des zones, cela permettra de créer un réseau au sein d'un système unique.

II.3.2 Fonctionnement : Sur la figure II.5, la carte d'interface réseau (NIC) est configurée avec trois VNIC. Chaque est assigné à une zone VNIC.

— Zone 1, Zone 2 et Zone 3 sont les trois les zones configurées pour l'utilisation dans le système. Les zones communiquent entre elles et avec le réseau externe à l'aide de leurs VNIC respectives.

— Trois VNIC se connectent à leur tour à la carte d'interface réseau physique sous-jacente via le commutateur virtuel. La fonction du commutateur virtuel équivaut à une connectivité fournie par un commutateur externe pour les systèmes connectés aux ports du commutateur.

Lorsque le réseau virtuel est configuré, une zone envoie le trafic vers un hôte externe de la même manière qu'un système sans réseau virtuel. Le trafic circule, par l'intermédiaire de la carte d'interface réseau virtuelle, de la zone vers le commutateur virtuel, puis vers l'interface physique, qui envoie les données au réseau.

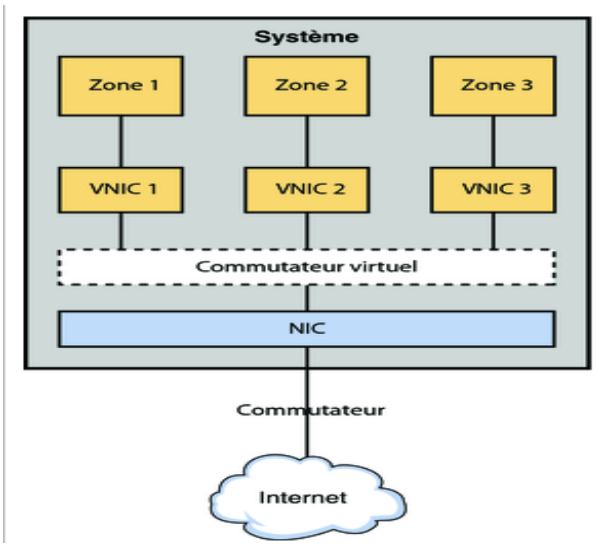


Figure II.5 : Configuration VNIC pour une interface unique NIC.

Les zones peuvent également échanger du trafic entre elles au sein du système si toutes les cartes VNIC configuré pour les zones font partie du même VLAN. Les paquets passent, par exemple, de Zone 1 par le biais de sa VNIC 1 dédiée. Le trafic passe ensuite à travers le commutateur virtuel vers VNIC 3. VNIC 3 passe ensuite le trafic à Zone 3. Le trafic ne quitte jamais le système, et ne viole donc pas les restrictions Ethernet.

II.3.3 Les différents types de virtualisation des réseaux d'opérateur :

Afin d'assurer une meilleure expérience utilisateur à faible coût, les futurs réseaux virtualisés nécessitent une architecture plus souple et plus élastique que celle des réseaux traditionnels, pour ce faire, il existe trois approches de virtualisation des réseaux d'opérateurs [19]:

- **La virtualisation locale :** Cette approche consiste à faire une subdivision de chaque BS en un ensemble de ressources afin de permettre la création de BS virtuelles (VBS). L'hyperviseur prend en charge l'allocation des ressources physiques et la synchronisation entre les différentes instances virtuelles installées sur la même VBS.

- **La virtualisation centralisée :** Consiste à centraliser tous les traitements (effectués sur le réseau) dans une seule entité centrale appelée orchestrateur réseau. Cette dernière doit maintenir à jour les informations concernant toute la topologie afin de déployer de manière optimale les réseaux virtuels.

- **La virtualisation hybride :** cette approche consiste à faire une combinaison optimale entre l'approche locale et l'approche centrale.

II.3.4 Les avantages de la VR :

- **Flexibilité :** La mise en place et le dimensionnement rapide du réseau peut répondre aux exigences de l'évolution des infrastructures.

- **Capacité multi-client :** Plusieurs réseaux logiques cohabitent indépendamment sur un même réseau physique, (un équipement physique peut être utilisé par les deux réseaux virtuels sans que les deux réseaux ne se voient), ceci a pour avantage de permettre le chevauchement des adresses IP.

- **Fiabilité :** L'aspect logique des réseaux virtuels leurs permet d'être sauvegardés et puis restaurés en cas d'erreurs ou perte de la configuration du réseau, avec une reprise très rapide en quelques instructions.

- **Agilité :** Nul besoin de modifier la topologie physique existante lors de modification de la topologie du réseau virtuel, l'ajout, la modification ou la suppression d'un routeur ne nécessitera pas une intervention sur l'infrastructure physique.

- **La migration** : La migration des éléments de réseau sont une exigence indispensable. Ce n'est qu'à travers leur migration sur les réseaux que les fournisseurs de services peuvent mettre en œuvre des protocoles personnalisés et déployer divers services.

II.4 Environnement de virtualisation de réseaux NVE :

Contrairement à l'Internet tout IP existant, l'environnement d'un réseau virtualisé est une collection de plusieurs architectures réseaux hétérogènes de différents fournisseurs de services. Chaque fournisseur de service loue des ressources d'un ou de plusieurs fournisseurs d'infrastructure pour créer des réseaux virtuels et donc pouvoir déployer des protocoles et services personnalisés à ces clients. En découplant le fournisseur de service du fournisseur d'infrastructure, la virtualisation introduit ainsi la flexibilité pour l'innovation et le changement. [20]

II.4.1 Modèle d'entreprise :

La principale distinction entre les réseaux à modèle de virtualisé et le modèle traditionnel est que dans le modèle virtualisé il y a là la présence de deux rôles joués par deux acteurs différents: fournisseurs d'infrastructure et les fournisseurs de services, par opposition aux réseaux traditionnels où le FAI est l'unique acteur et remplit les deux rôles susmentionnés

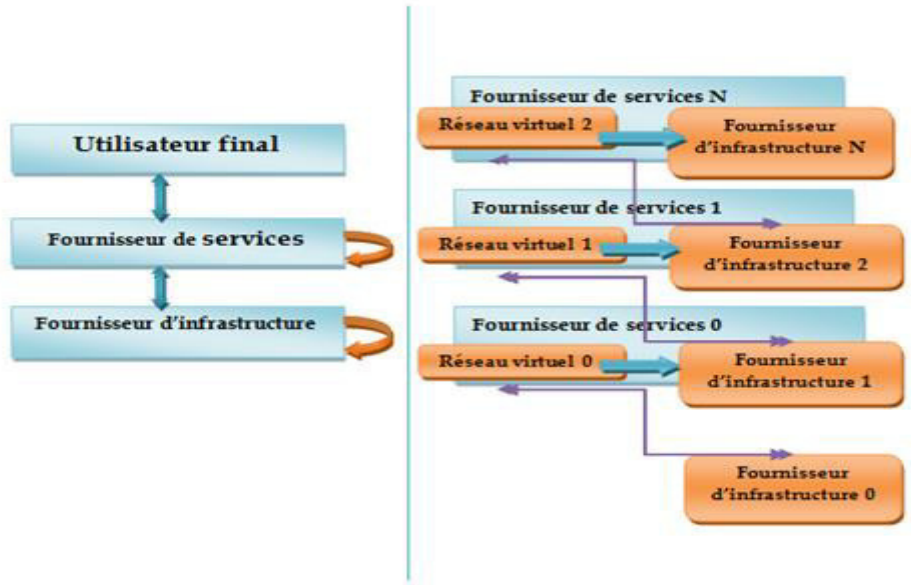


Figure II.6 : Modèle d'entreprise.

- **Les fournisseurs d'infrastructure (InP) :**

Ils déploient et gèrent réellement les ressources réseau physiques sous-jacentes. Ils offrent leurs ressources via des interfaces programmables à différents fournisseurs de services (SP). Les fournisseurs d'infrastructure se distinguent grâce à :

— la qualité des ressources qu'ils fournissent.

— la liberté ils délèguent à ces clients (SP) et les outils qu'ils leur fournissent pour exploiter cette liberté.

- **Les fournisseurs de services (SP) :**

Les fournisseurs de services louent des ressources auprès de plusieurs InP pour créer et déployer des réseaux virtuels en programmant des ressources réseau allouées pour offrir des services de bout en bout aux utilisateurs finaux.

Un fournisseur de services peut également :

— Fournir des services de réseau à d'autres les fournisseurs de services.

— Créer des réseaux virtuels enfants en partitionnant ses ressources et peut agir comme un InP virtuel par la location ces réseaux enfants à d'autres fournisseurs de services.

- **Utilisateur final :**

L'utilisateur final dans le modèle de virtualisation de réseau est similaire à celui de l'Internet existant, sauf que l'existence de plusieurs réseaux virtuels de SP concurrents lui fournit une plus large gamme de choix. Tout utilisateur final peut se connecter à plusieurs réseaux virtuels de différents SP pour différents services.

II.4.2 Architecture NVE :

Dans un environnement de virtualisation de réseau (NVE), l'entité de base est un réseau virtuel (VN). Un VN est un ensemble de nœuds virtuels connectés ensemble par le biais de liens virtuels pour former une topologie virtuelle, qui est essentiellement un sous-ensemble du réseau physique sous-jacent. Chaque nœud virtuel est hébergé sur un nœud physique particulier [20]

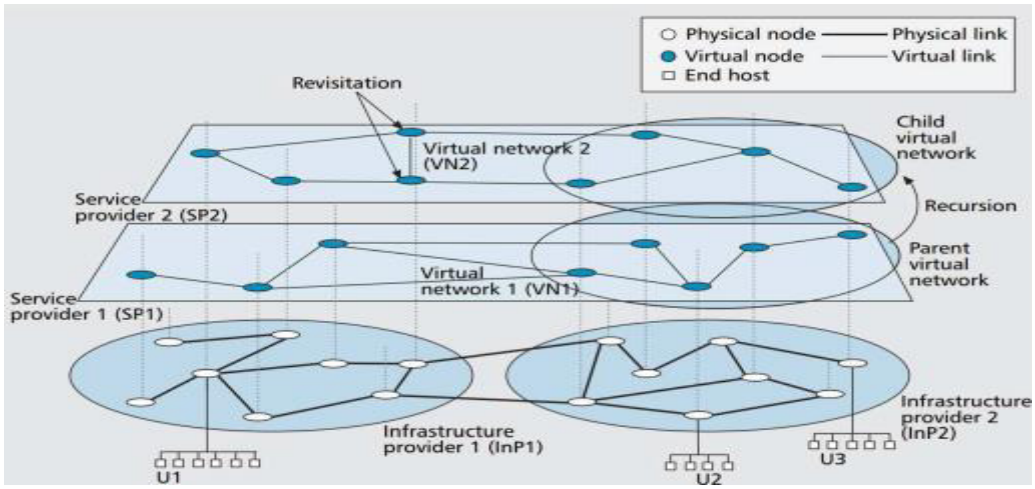


Figure II.7 : Architecture d'un environnement réseau virtualisé.

Chaque VN est exploité et géré par un seul SP, même si les ressources physiques sous-jacentes peuvent être agrégées à partir de plusieurs InP. La Figure II.7 représente deux

réseaux virtuels, VN1 et VN2 créés par les fournisseurs de services SP1 et SP2 respectivement :

— SP1 opère sur VN1 qui se situe au-dessus du réseau physique gérée par deux InP différents (InP1 et InP2) et fournit ainsi des services de bout en bout aux utilisateurs finaux U2 et U3.

— SP2, d'autre part, qui est déployé sur VN2 en combinant les ressources du fournisseur d'infrastructure InP1 avec un réseau enfant VN du service fournisseur SP1. Les utilisateurs finaux U1 et U3 sont connectés via VN2.

Le propriétaire d'un VN est libre de mettre en œuvre des services de bout en bout en déployant des formats de paquets personnalisés, des protocoles de routage, les mécanismes d'acheminement ainsi que des plans de contrôle et la gestion du VN. Comme mentionné précédemment, les utilisateurs finaux ont le choix d'opter pour n'importe quel VN. Par exemple, l'utilisateur final U3 est abonné à deux les réseaux virtuels VN1 et VN2 gérés par SP1 et SP2 respectivement.

II.4.3 Les principes architecturaux : Un environnement de virtualisation réseau (VNE) doit prendre en considération différentes contraintes :

- **La coexistence :** Signifie l'existence de plusieurs (SPs) sur la même machine physique sans que l'un impacte le bon fonctionnement de l'autre.

- **La récursivité :** Signifie que la relation parent-enfant de chaque réseau virtuel (VN) doit être bien définie afin de bien savoir le créateur de chaque VN. Comme nous pouvons le constater sur la Figure II.7, le SP1 a loué une portion de ses propres ressources au SP2. Dans ce cas-là, le SP1 a agi comme un InP afin d'attribuer le VN fils au SP2.

- **L'héritage :** Un VN enfant dans un VNE peut hériter les caractéristiques de son père (celui qui l'a créé). Comme illustré sur la Figure II.7, le VN2 du SP2 hérite automatiquement les caractéristiques du VN1 responsable de sa création.

- **La revisitassions :** Consiste à permettre à un nœud physique de déployer plusieurs nœuds logiques appartenant au même VN.

II.5 Passage vers le SDN [21]:

Les réseaux actuels (qui commencent à être de plus en plus appelés réseaux traditionnels) ont la particularité d'être statique et montrent leurs faiblesses lors des connexions entre les usagés, de ce fait tout changement de configuration exigera directement une intervention manuelle d'un administrateur sur l'équipement dans le but d'établir une connexion dotée d'un niveau approprié de QoS et de sécurité, cela peut prendre beaucoup de temps ainsi que d'énergie. Cette pression est devenue trop forte pour les entreprises et la situation n'est souvent plus tenable.

Face aux problèmes parvenus dans les réseaux traditionnels une nouvelle vague de charge de travail applicatif et de trafic de données apparaît comme une solution de création et de gestion des réseaux dynamiques qui est le SDN.

— Un nouveau concept d'architecture réseau est donc né en 2008, fruit de travail des équipes de recherche des universités de Brekley, Stanford et du MIT *(Massachusetts Institute of Technology).*

— En 2011, l'ONF *(Open Networking Fondation),* une organisation à but non lucratif dont l'objectif est la promotion du SDN, est créée par Deutsche Telekom, Facebook, Google, Microsoft, Verizon et Yahoo. De plus, elle compte parmi ses membres tous les grands constructeurs IT telles que Cisco, Juniper, HP, Dell, Broadcom, IBM et bien d'autres.

— Le SDN se base sur le principe des réseaux virtuels, et il permet la coexistence de plusieurs services sur une même et unique infrastructure physique d'une manière dynamique

En résumé les SDN ne sont pas simplement un travail de recherche universitaire, mais bien une nouvelle organisation des réseaux qui intéresse fortement tous les grands acteurs du réseau d'aujourd'hui

II.6 Présentation du SDN

II.6.1 Définition du SDN :

Le SDN signifie littéralement *(Software Defined Networking),* donc c'est une technologie permettant de définir une architecture réseau via des applications, dans le but d'accélérer le déploiement et de faciliter la gestion des réseaux.

Le SDN est une architecture émergente qui est dynamique, rentable, et adaptable à la nature dynamique des applications, cette architecture découple les fonctions de contrôle du réseau et de transfert (données) en virtualisant la couche réseau et en créant une couche logicielle de l'hyperviseur qui vient de se placer au-dessus des outils matériels à fin d'offrir une vision unique des équipement et de rendre le réseau directement programmable**.[22]**

Le SDN est donc reconnue aujourd'hui comme une architecture permettant d'ouvrir le réseau aux applications. Cela intègre les deux volets suivants :

— Permettre aux applications de programmer le réseau afin d'en accélérer le déploiement.

— Permettre au réseau de mieux identifier les applications transportées pour mieux les gérer (qualité de service, sécurité, ingénierie de trafic...).

II.6.2 Modèle de SDN: Le SDN englobe toutes les solutions permettant une programmation et automatisation du réseau et cela en se basant sur la virtualisation, afin de mieux interagir avec les applications. Diverses solutions coexistent, adaptées selon les besoins des utilisateurs.

Il existe actuellement sur le marché deux classes de modèles de SDN basés sur contrôleurs, tous les deux peuvent être utilisés selon le besoin et le contexte [23]:

II.6.2.1 Superposition :

Le modèle par superposition est basé sur le tunneling et l'encapsulation. L'un des avantages du modèle par superposition est qu'il est totalement indépendant vis-à-vis du réseau IP sous-jacent et qu'il peut de ce fait être mis en œuvre sans qu'aucune modification de ce

réseau ne soit nécessaire. Tout simplement parce que les commutateurs virtuels (l'un des composants de la solution) sont déployés à l'échelle du réseau et que des tunnels sont placés entre eux, et non sur les périphériques physiques.

Lors de son apparition, elle a été utilisée sans implémentation de contrôleur, elle reprenait en quelque sorte le fonctionnement des VPN, mais actuellement le modèle Overlay (superposition) inclus forcement un contrôleur Une solution de virtualisation de réseau se compose de plusieurs éléments. Notamment :

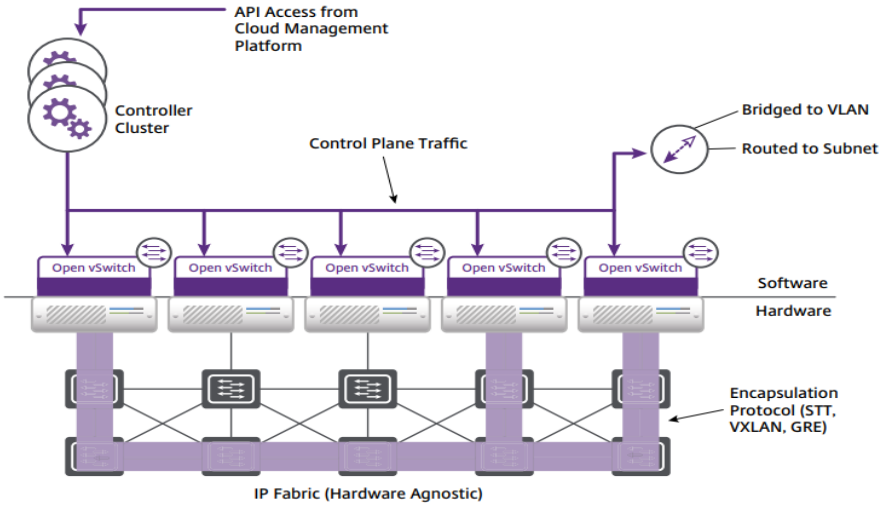


Figure II. 8 : Le modèle superposé.

— Le réseau physique sur lequel la solution de virtualisation de réseau s'exécute. Il est souvent appelé réseau sous-jacent.

— L'ensemble de réseaux virtuels composé de tunnels s'exécutant sur le réseau physique et leurs commutateurs virtuels correspondants. Il est souvent appelé réseau superposé.

— Les protocoles qui sont utilisés pour encapsuler le trafic en transit depuis le commutateur d'accès jusqu'au commutateur de sortie sont des protocoles d'encapsulation à titre d'exemple: VXLAN.

II.6.2.2 Infrastructurel :

Ce modèle est mis en œuvre via une application qui s'exécute sur un contrôleur SDN, qui communique avec les périphériques réseau sous-jacents. Dans le modèle infrastructurel, les réseaux virtuels sont basés sur des stratégies faisant correspondre les flux au réseau virtuel approprié à l'aide de l'en-tête du paquet. Le contrôleur SDN met en œuvre ces réseaux virtuels en configurant des tables de redirection sur des périphériques du réseau physique utilisant des protocoles communs (OpenFlow, par exemple).

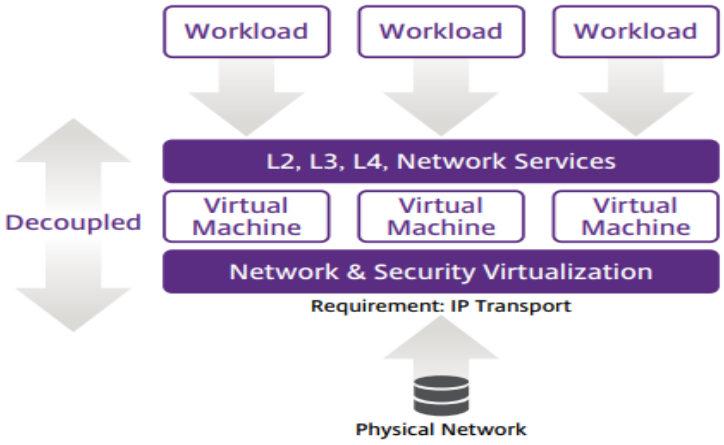


Figure II.9 : Le modèle infrastructurel.

Les avantages et les inconvénients de se modèle sont illustrés dans se tableau ci-dessous :

Les avantages	**Les inconvénients**
— Contrairement au modèle par superposition, il n'y a qu'un seul réseau à gérer, ce qui dans beaucoup de cas réduit les coûts et la complexité. — L'approche infrastructurelle ne s'appuyant pas sur le tunneling et l'encapsulation, la visibilité et la supervision du trafic applicatif s'en trouvent grandement simplifiées.	— Implique l'emploi de commutateurs et de routeurs prenant en charge un nouveau modèle de redirection, ce qui le rend plus adapté aux nouveaux déploiements de réseaux.

Tableau 2.1 : Les avantages et les inconvénients du modèle infrastructurel.

II.6.3 Architecture du SDN : L'ONF suggère un model de référence pour le SDN, ce dernier est constitué de 3 couches à commencer par la première qui est la couche d'infrastructure suivie par la couche de contrôle et enfin la couche d'application comme le montre la figure II.10. [24]

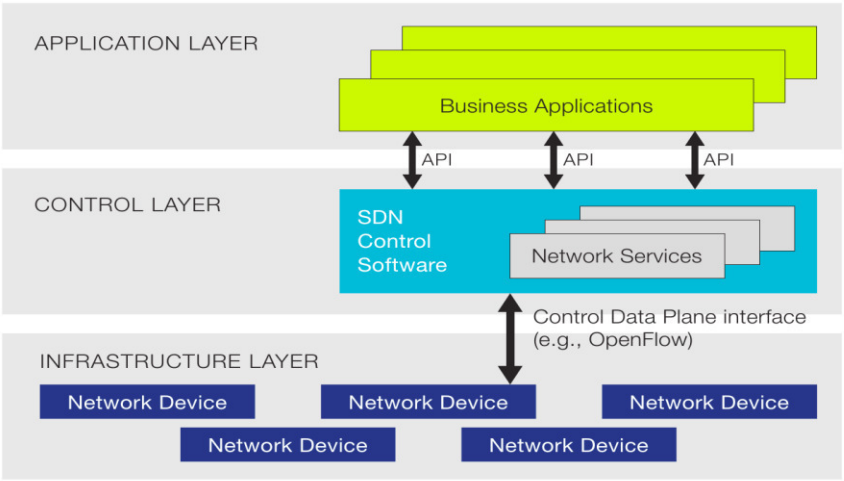


Figure II.10 : Architecture du SDN.

II.6.3.1 La couche infrastructure :

Cette couche est principalement constituée des équipements physiques/virtuels du réseau qui sont responsables de l'acheminement du trafic tel que les switchs et les routeurs qui sont interconnectés pour former un seul réseau. La fonction principale de ses équipements :

- La collecte des états du réseau (la topologie actuelle du réseau, statistiques du trafic sur le réseau, la densité d'utilisation du réseau) et les envoyer au contrôleur.
- Le traitement des paquets se basant sur les règles de gestion fournit par le contrôleur.
- les équipements physiques sont interconnectés à travers différents supports de transmission, comme les files de cuivre, fibre optique, les faisceaux hertzien.
- Acheminement, fragmentation et réassemblage des paquets.

II.6.3.2 La couche contrôle :

Est logicielle, basée sur le contrôleur SDN, permet de contrôler le plan de données (la couche infrastructure) en établissant des règles qu'il devra suivre. La couche contrôle relie

l'infrastructure du réseau avec la couche application à travers ses deux interfaces. L'interaction avec la couche physique se fait à travers une interface sud (The South-Bound interface) qui permet de :

- Importer les règles de transmission de paquets à partir du contrôleur pour désigner le comportement des équipements physiques de réseau.
- Envoyer l'état du réseau au contrôleur.
- Supporter le protocole OpenFlow qui se positionne comme une API sud agissant directement sur le plan de données.
- Etablissement des tables de routages.

L'interaction avec la couche application se fait à travers une interface vers le nord (north-bound interface), elle fournit des points d'accès aux services sous diverses formes par exemple : une interface de programmation d'applications (API). Dans le cas du SDN, ces applications ont accès aux informations reportées par le switch et peuvent ainsi prendre des décisions en définissant les règles de transfert de paquest aux switchs et tout ce la grâce aux API.

Les interfaces côté Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs pour synchroniser les états du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible.

II.6.3.3 La couche application :

La couche application concerne tout ce qui touche l'administration de l'équipement, il s'agit donc des applications SDN conçues pour répondre aux exigences d'une infrastructure, À travers la plate-forme programmable fournie par la couche contrôle, les applications SDN peuvent accéder et contrôler les dispositifs de commutation dans la couche infrastructure, et d'introduire de nouvelles fonctions et de services. Exemple d'applications SDN :

- application d'équilibrage de charge
- virtualisation du réseau.

II.7 Les avantages de SDN :

- Facilité de la conception et du contrôle du réseau, et surtout l'existence d'une structure de contrôle centralisée du trafic dans le réseau.

- La possibilité d'exécution d'une ou plusieurs règles de réseaux sur un équipement standard.

- L'orchestration et l'automatisation des réseaux de différent fournisseurs permet de provisionner des services réseaux, rapidement et à grande échelle, en réduisant le risque de l'erreur humain.

- La vue globale des réseaux par le contrôleur permet de remplacer les protocoles de routage distribués (OSPF, EIGRP, RIP…) par des mécanismes plus complexe.

- Faible coût à comparer avec les équipements réseaux actuels que nous achetons fermés qu'il est impossible de développer ou de fusionner avec d'autres produits, ainsi que de réduire le nombre d'ingénieurs, et gagner le temps de travail de plus de 50%.

- L'acheminement de services dynamiques virtuels définit par une politique des flux selon les exigences des entreprises

- ne pas être dépendant d'un seul fournisseur de produit (comme Cisco ou Juniper), ce qui permet l'existence d'une flexibilité dans le réseau, cela permettra à l'administrateur de faire des choix indépendants des meilleurs éléments au sein du réseau.

- L'existence des SDN va augmenter le taux d'innovation au niveau de l'infrastructure réseau, et cela conduira à l'apparition de nouvelles idées, par exemple, les développeurs peuvent tester des applications au sein du réseau sans affecter les performances du réseau ou sur d'autres services.

II.8 Applications du SDN : Les réseaux SDN ont une large variété d'applications dans les environnements réseaux, et permettent un contrôle personnalisé, une opportunité d'éliminer les équipements intermédiaires et ils simplifient également le développement et le déploiement de nouveaux services et protocoles réseaux.

II.8.1 Sécurité : Grâce à l'architecture centralisée de SDN, le contrôleur peut détecter les attaques rapidement et limiter leurs effets et cela avec un minimum de configuration tout en gardant une surcharge très faible sur le contrôleur parmi ses applications dans la sécurité on trouve:

— Inspecter tous les échanges de flux entre les switch et le contrôleur peut prévenir des attaques telle que l'usurpation d'adresse IP, car le contrôleur peut déployer des applications dédiées uniquement à la sécurité dans le but de la renforcer.

— Les composants IDS chargés de détecter les attaques peut reconnaitre les équipements non sécurisés pour informer le contrôleur SDN, de dernier procédera à l'isolation de l'équipement suspect, en définissant les règles appropriées via OpenFlow.

II.8.2 Multimédia et QoS : L'architecture Internet d'aujourd'hui repose sur l'envoi des paquets sans prendre en considération le type des paquets et la qualité de la transmission. Les applications multimédias comme le streaming vidéo, VoD, la vidéoconférence, la WebTV, etc, nécessitent des ressources réseau stables et tolèrent les erreurs et les retards de transmission. En se basant sur la vue centralisée du réseau offerte par SDN :

— On peut sélectionner selon le débit, des chemins différents pour les divers flux de trafic.
— Les auteurs du multimédia peuvent fournir de la vidéo sur SDN (VSDN) en utilisant la vue de l'ensemble du réseau que fournit l'architecture SDN qui détermine la trajectoire optimale pour tout service multimédia.

II.8.3 Réseaux sans fil : SDN a été aussi appliqué dans le domaine des réseaux sans fil, en particulier 5G, sous le nom de SDWN *(Software Defined Wireless Network)* mais aussi dans les réseaux cellulaire sous le nom CSDN *(cellular SDN)* et dans les réseaux domicile *(HAN)* .Ces différentes technologies ont été proposées comme solution d'intégration du SDN dans les réseaux sans fils afin :

- Faciliter la gestion et supporter le déploiement de nouvelles applications dans l'infrastructure réseau.
- Optimiser la consommation d'énergie, l'utilisation des ressources et la personnalisation des services aux utilisateurs notamment dans le CSDN *(cellular SDN)*.

II.8.4 Data center et Cloud computing : L'apparition de nouvelles technologies comme le cloud qui consiste à concentrer les ressources (stockage, calcul, etc) dans des centres de traitement de données et de les offrir comme plateforme ou service à l'utilisateur à augmenter la quantité de données manipulées et par conséquent l'explosion du nombre de centres de données**.** La gestion de ces ressources informatiques est le défi le plus important des réseaux Cloud. SDN est hautement considérée comme une des solutions les plus récentes, qui permet de configurer et de gérer facilement le Cloud et les centres de données et cela en fournissant un réseau virtualisé des centres de donneés dans le but de:

- Relier dynamiquement des machines virtuelles avec les serveurs de réseau.
- Déployer à la demande de nouveaux services comme le pare-feu et l'équilibrage de charge.
- Faciliter la configuration et la surveillance des réseaux virtuels en tout lieu.

II.9 Discussions :

S'il est vrai que le SDN s'appuie sur de nombreuses technologies habilitantes comme l'environnement de virtualisation pour créer des réseaux virtuels, il ne constitue pas en soi une technologie à proprement parler, mais bien une architecture. La virtualisation de réseau peut être considérée comme une application SDN. Le principal avantage d'une solution de virtualisation de réseau est qu'elle permet une prise en charge de la mobilité des machines virtuelles totalement indépendante du réseau physique et offre une interopérabilité quasi parfaite entre les FAI pour implémenter leurs différents services sur une unique infrastructure.

Mais le SDN présente de nombreux autres avantages potentiels, comme la programmabilité du réseau via des protocoles spécialisés et des équipements adaptés qui vont faire l'objet de notre étude dans le chapitre qui suit.

Chapitre III : Les réseaux programmables

III.1 Préambule :

Dans le fonctionnement actuel des réseaux IP, chaque équipement de réseau (routeur IP, commutateur Ethernet) exécute ses fonctions du plan de contrôle et du plan de données. SDN *(Software Defined Network)* propose de créer un point central qui gère le plan de contrôle, tandis que les commutateurs/routeurs physiques n'ont plus à prendre en charge que le plan de données.

Pour se faire, l'Open Networking Foundation (ONF) a mit en œuvre OpenFlow. Il s' agit d'un protocole standard utilisé par le contrôleur pour transmettre au commutateur des instructions qui permettent de programmer leur plan de données et d'obtenir des informations de ces commutateurs afin que le contrôleur puisse disposer d'une vue globale logique (abstraction) du réseau physique.

Cette vue est utilisée pour toutes les décisions que doit prendre le plan de contrôle (routage, filtrage de trafic, partage de charge, traduction d'adresse etc.). Le contrôleur SDN fournit une API aux « applications SDN » qui inclut cette vue globale. Ces applications implémentent, via le contrôleur, des services tels que le routage de flux, la mise en œuvre de politiques de QoS pour les flux, la sécurité de bout en bout des flux, le filtrage de flux, etc. Cette approche permet une mise en œuvre flexible et rapide de nouvelles applications réseau qui émule les « appliances » réseau.

Dans ce chapitre on introduira le protocole OpenFlow ainsi que les éléments du réseau OpenFlow et le principe de fonctionnement de ce dernier.

III.2 Le concept OpenFlow :

OpenFlow est le premier protocole de communication standard qui est définit entre les couches de contrôle et de données d'une architecture SDN. Il s agit d'une nouvelle approche réseau basée sur des commutateurs Ethernet, qui met en place un vaste champ d'expérimentation sur le réseau réel sans avoir à en perturber le fonctionnement ni en dépenser des coûts supplémentaires pour d'autres types d'infrastructures.

L'idée de base est la suivante : La technologie exploite le fait que la plupart des switchs et routeur Ethernet modernes contiennent des tables de flux qui fonctionnent à haut

débit pour implémenter des firewalls, NAT, QoS et pour collecter des statistiques. Ces tables de flux sont propres à chaque constructeur, mais on peut en identifier un ensemble de fonctionnalités qui sont implémentées sur la plupart des routeurs et des switchs afin d'être exploité par l'OpenFlow. [26]

III.3 Pourquoi OpenFlow?

OpenFlow garantit la séparation entre le plan de contrôle (responsable d'associer une décision au paquet) et le plan de données (responsable du transfert des paquets dans le réseau) ainsi que le traitement des paquets comme des flux de données, où un flux de paquets est défini comme une combinaison d'entêtes de couche 2, 3 et 4. Le plan de données doit être implémenté dans les switchs OpenFlow, ce qui allégera les switchs qui n'auront pour responsabilité que l'acheminement des paquets. Le plan de contrôle est assuré par une machine séparée (le contrôleur), qui réalise les traitements nécessaires pour déterminer les décisions à appliquer sur un flux de données. [27]

Cette séparation entre le plan de contrôle et celui des données permettra d'atteindre du très haut débit. Un autre avantage de cette séparation est que les composants de l'équipement responsable du plan de données seront optimisés pour le transfert des paquets ce qui évite d'alourdir le chemin de traitement avec des fonctionnalités qui relèvent du plan de contrôle.

III.4 Le protocole OpenFlow :

Il s'agit d'un protocole ouvert, ce standard est utilisé par le contrôleur pour transmettre et ajouter au commutateur des instructions. Ces dernières (appelées flow entries) sont des règles avec un modèle « pattern » (IP source ou destination, mac adresse, port TCP…) et une action correspondante (rejeter le paquet, transmettre sur port x, ajouter un entête VLAN …).

Ces règles permettent de programmer le plan de données et d'obtenir des informations de ces commutateurs afin que le contrôleur puisse disposer d'une vue globale logique (abstraction) du réseau physique. Cette vue est utilisée pour toutes les décisions que doit prendre le plan de contrôle (routage, filtrage de trafic, partage de charge, traduction d'adresse, etc.).

OpenFlow étant un standard ouvert, il peut donc être utilisé par tout contrôleur compatible OpenFlow pour communiquer avec tout équipement réseau compatible OpenFlow, indépendamment du constructeur. [28]

III.4.1 Les messages du protocole OpenFlow : Trois types de message peuvent apparaître entre le commutateur et le contrôleur. Ils sont : [29]

III.4.1.1 Les messages contrôleur->switch : Ils représentent la catégorie la plus importante de messages OpenFlow. Ils sont classés en cinq sous-catégories :

- **Configuration du switch :** Sont émis uniquement par le contrôleur vers le switch :
  - **SET_CONFIG:** Afin de positionner les paramètres de configuration du switch
  - **FEATURES_REQUEST/REPLAY:** Afin d'interroger le switch au sujet des fonctionnalités (optionnelles) qu'il supporte
  - **GET_CONFIG_REQUEST/ REPLAY:** Utilisée afin d'obtenir la configuration du switch.

- **Commandes venant du contrôleur :** Sont au nombre de trois **:**
  - **PACKET_OUT :** Afin d'émettre des paquets de données au switch pour leur acheminement via le plan de données.
  - **FLOW_MODE :** Pour modifier les entrées de flux existantes dans le switch
  - **PORT_MODE:** Utilisé pour modifier l'état d'un port OpenFlow.

- **Statistiques :** Des statistiques sont obtenues du switch par le contrôleur via la paire de message **STATS_REQUEST et STATS_REPLY.**

- **Configuration de files d'attente de port :** le contrôleur peut interroger le switch via **QUEUE_GET_CONFIG_REQUEST** acquitté par **QUEUE_GET_CONFIG_REPLY** pour apprendre quelle est la configuration des files d'attentes associées à un port ainsi fournir à ces paquets un niveau de QoS désiré.

- **Barrière : BARRIER_REQUEST :** Utilisé par le contrôleur pour s'assurer que tous les messages émis par le contrôleur et qui ont précédé cette requête ont été reçus et traités par le commutateur. La confirmation est retournée par le switch via la réponse **BARRRIER_REPLY.**

III.4.1.2 Les messages asymétriques : Emis indifféremment par le contrôleur ou le switch à travers le canal sécurisé sans avoir été sollicité par l'autre entité :

— **HELLO :** Emis afin de déterminer le numéro de version OpenFlow le plus élevé supporté par le switch et le contrôleur.

— **ECHO :** Sont émis pour s'assurer que la connexion est toujours en vie et afin de mesurer la latence et le débit de la connexion. Chaque message ECHO_REQUEST doit être acquitté par un message ECHO_REPLY.

III.4.1.3 Les messages asynchrones : Initiés par le switch et utilisés pour mettre à jour le contrôleur :

— **FLOW_REMOVED :** Pour informer le contrôleur qu'une entrée de flux a été supprimée de la table de flux.

— **PACKET_IN :** Utilisé par le switch pour passer les paquets de données au contrôleur pour leur prise en charge.

— **ERROR :** Pour notifier des erreurs au contrôleur.

— **PORT_STATUS :** Utiliser pour notifier un changement de port ou d'état du port.

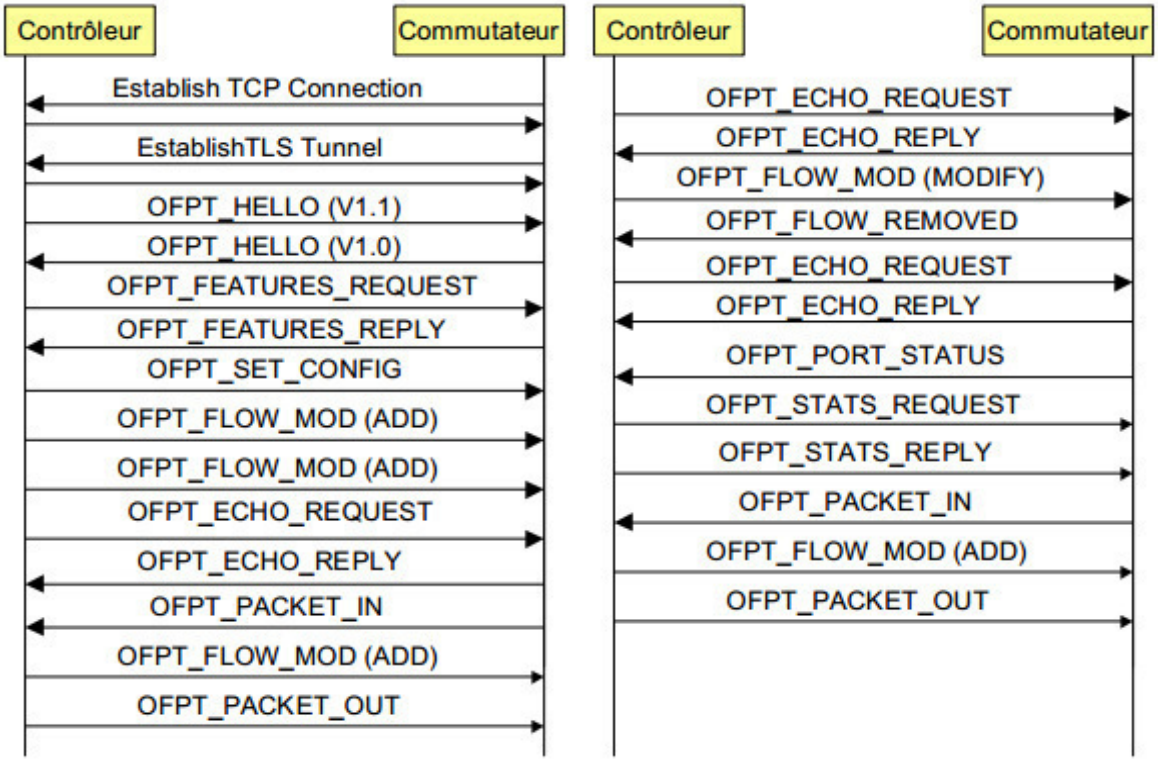


Figure III.1 : Echanges OpenFlow entre contrôleur et commutateur.

III.4.2 Etablissement d'une connexion contrôleur- switch OpenFlow :

Nous allons citer ci-dessous trois cas d'un établissement d'une connexion entre le commutateur OpenFlow et le contrôleur : [30]

- **Cas N°1 :**

Tout d'abord, il faut que le commutateur ait l'adresse IP de son contrôleur d'attache, car il peut y avoir plusieurs contrôleurs dans un seul réseau. Lors du démarrage du commutateur OpenFlow, ce dernier envoie un paquet « OFPT_HELLO » avec le numéro de version d'OpenFlow supportée. Le contrôleur vérifie la version d'OpenFlow supportée par le commutateur et lui répond par un message « OFPT_HELLO » en indiquant la version d'OpenFlow avec laquelle ils communiqueront. La connexion est ainsi établie.

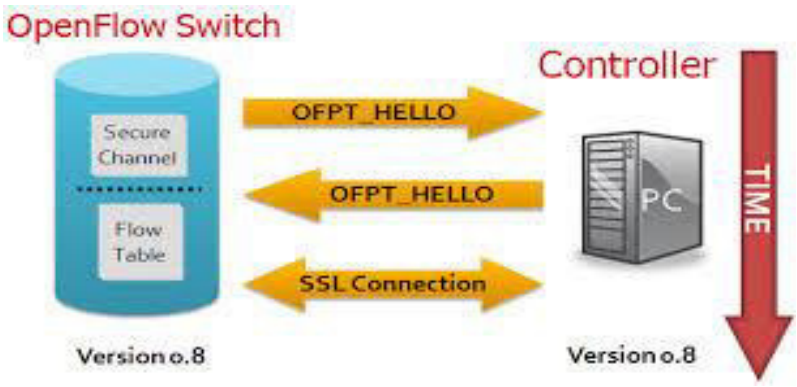


Figure III.2 : Connexion au contrôleur OpenFlow.

- **Cas N°2 :**

Comme dans le cas précédent le commutateur OpenFlow envoie un paquet « OFPT_HELLO » avec la version du protocole utilisé, le contrôleur s'aperçoit qu'il ne supporte pas la version OpenFlow du commutateur. Il lui retourne donc un paquet « OFPT_ERROR » en indiquant que c'est un problème de compatibilité, comme illustré dans la figure suivante.

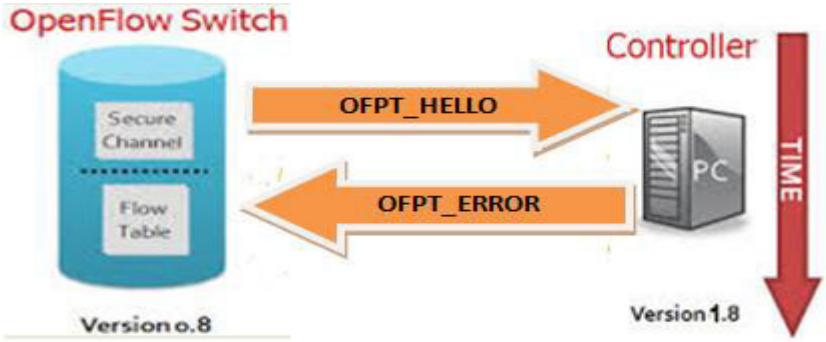


Figure III.3 : Échec de la connexion au contrôleur OpenFlow.

- **Cas N°3 :**

Comme dans les cas précédents le commutateur envoie un paquet « OFPT_HELLO » au contrôleur, si celui-ci ne répond pas il tente alors de joindre les éventuels autres contrôleurs qui lui ont été paramétrés. S'il ne parvient pas à les joindre, il se met alors en mode urgence

« EMERGENCY MODE ». Le commutateur utilise sa table de flux par défaut, si toutefois un paquet ne correspond à aucun enregistrement dans sa table il le supprime, comme le montre la figure ci-dessous :

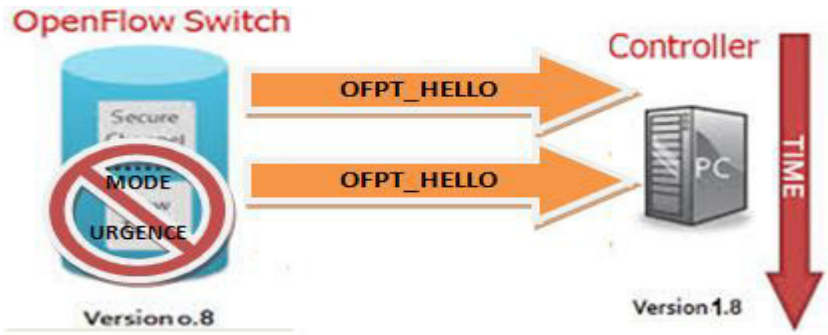


Figure III.4 : Switch OpenFlow en mode d'urgence.

III.5 Composant d'un réseau OpenFlow :

OpenFlow propose une nouvelle architecture bien adaptée aux environnements des réseaux virtuels, l'idée principale est de faire communiquer les deux plans qui ont été séparés,

le contrôleur quant à lui peut desservir l'ensemble du réseau par le biais d'OpenFlow en utilisant les liaisons du réseau.

La figure suivante détaille les éléments d'un réseau OpenFlow. Le réseau est formé essentiellement d'un ou plusieurs commutateurs OpenFlow, un ou plusieurs contrôleurs, et finalement le protocole OpenFlow définissant tous les messages échangés entre les contrôleurs et les commutateurs et permettant de standardiser la communication.

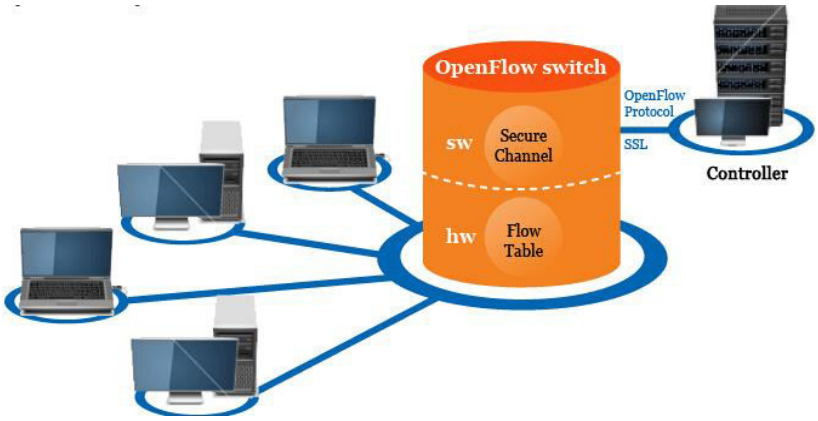


Figure III.5: Réseau OpenFlow.

III.5.1 Switch OpenFlow:

Les éléments de base du switch OpenFlow sont les tables OpenFlow. Ces tables vont assurer la classification des paquets en se basant sur plusieurs champs des entêtes niveau 2,3 ou 4 (Le Tableau III.1 illustre les champs pour la version 1.2 du protocole OpenFlow). On distingue deux types de switch OpenFlow : [31]

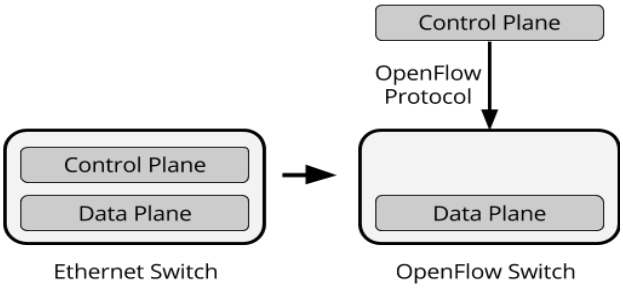


Figure III.6 : switch OpenFlow.

- **Switch OpenFlow-only :** Supporte uniquement les actions requises pour le fonctionnement du protocole OpenFlow.

- **Switch OpenFlow-enabled :** En plus des actions requises pour le protocole OpenFlow, ce switch supporte les actions d'un switch ordinaire.

La figure ci-dessous illustre les principales fonctions d'un commutateur OpenFlow.

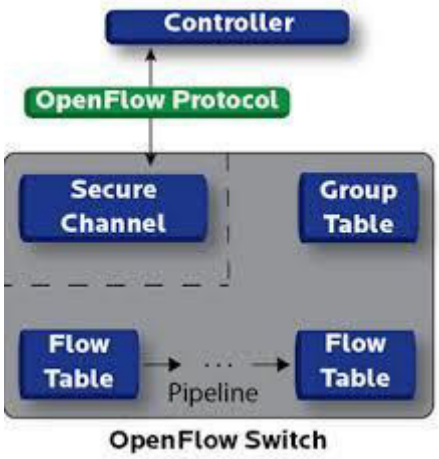


Figure III.7: Elements du switch OpenFlow.

III.5.1.1 Le canal sécurisé :

Est l'interface qui connecte chaque switch OpenFlow à un contrôleur. Cette interface permet au contrôleur de recevoir les messages du switch et de pouvoir la gérer à travers le réseau. Le canal doit être sécurisé afin d'assurer le bon déroulement des communications entre le switch et le contrôleur. Pour cela l'échange de message se fait au cours d'une session TCP établie via le port 6653 du serveur contrôleur ou à travers une connexion SSL/TLS *(Secure Sockets Layer / Transport Layer Security*). [31]

III.5.1.2 Table de flux (flow table) :

Un "Flow Table" est composé de plusieurs entrées de flux, chacune est structurée comme indiqué dans la figure III.6 : [32]

- **Champs de correspondance (Match Fild) :** Utilisés par le contrôleur lors de la recherche de l'entrée correspondante au paquet, cela consiste à vérifier le port d'entrée ou l'entête du paquet afin d'y appliquer une action x. L'ensemble des champs du match field sont listés ci-dessous :

Champ	N°couche	Application
Port d'entrée	2	Tous les paquets
Adresse MAC source	2	Tous les paquets sur le port activé
Adresse MAC destination	2	Tous les paquets sur le port activé
Type Ethernet	2	Tous les paquets sur le port activé
ID VLAN	2	Tous les paquets avec un tag VLAN
Priorité VLAN	2	Tous les paquets avec un tag VLAN
Etiquette MPLS	2,5	Tous les paquets avec un tag MPLS
Classe de trafic MPLS	2,5	Tous les paquets avec un tag MPLS
Adresse IPv4 source	3	Tous les paquets IPv4 et ARP
Adresse IPv4 destination	3	Tous les paquets IPv4 et ARP
Protocole IPv4	3	Tous les paquets IPv4/6 sur Ethernet et ARP
TOS bits IPv4	3	Tous les paquets IPv4
Port de transport source	4	Tous les paquets UDP, TCP, SCMP et ICMP
Port de transport destination	4	Tous les paquets UDP, TCP, SCMP et ICMP

Tableau III.1 : Champs de correspondance (Matching Filds) OpenFlow.

- **Compteurs (Counters) :** Servent essentiellement à garder des statistiques de gestion des flux pour ensuite décider si une entrée de flux est active ou non. Pour chaque table, chaque flux, chaque port, chaque file d'attente et chaque groupe de tables, des compteurs de statistiques sont maintenus.

- **Instructions :** Représentent l'ensemble des instructions OpenFlow qui servent à modifier le traitement (pipeline) que va subir le paquet. Les instructions supportées sont :

— **Apply-Actions :** Pour appliquer les actions sur le paquet immédiatement

— **Clear-Actions :** Pour supprimer une liste des actions du paquet.

— **Write-Actions :** Ajouter une liste d'actions au paquet.

— **Write-Metadata:** Ajouter des données utiles pour le séquencement entre les tables OpenFlow.

— **Goto-Table :** Indique que le paquet doit être acheminé vers une table d'indice supérieur pour subir un traitement pipeline.

Figure III.8: Entrée de table de flux.

III.5.1.3 Table de groupes :

Une Table de groupe se compose d'entrées de groupe. Ceci permet à OpenFlow d'envisager d'autres méthodes de transfert comme l'application d'un ensemble de transactions d'actions sur un type de Aux. La figure III.7 montre la structure d'une entrée de groupes : [32]

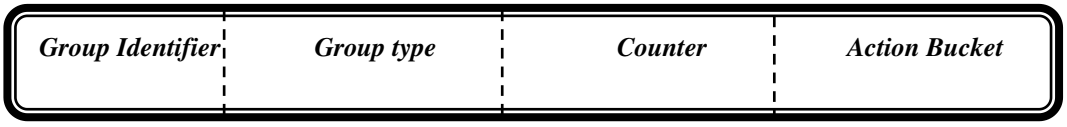


Figure III.9: Exemple de table de groupe.

- **Group Identifier :** Utilisé pour identifier d'une façon unique un groupe donné.
- **Group Type :** Sert essentiellement à déterminer la sémantique du groupe.
- **Counters :** Ensemble des compteurs qui vont être mise à jour une fois qu'un paquet est traité par le groupe.

- **Action Buckets :** Contient une liste ordonnée de conteneurs d'actions, où chaque conteneur contient une série d'actions à exécuter et les paramètres associés.

III.5.2 Les contrôleurs SDN :

Un contrôleur est une application dans les SDN qui a pour mission de fournir une couche d'abstraction du réseau et de présenter ce dernier comme un système. Le contrôleur SDN permet une mise en réseau intelligente, il implémente rapidement un changement sur le réseau en traduisant une demande globale en une suite d'opération sur l'équipement, il s'appuie sur le protocole OpenFlow qui permet au contrôleur de communiquer aux commutateurs ou envoyer les paquets via des API, et aussi configurer les périphériques réseaux, choisir le chemin de réseau optimal et un meilleur débit pour le trafic des applications.

En effet le contrôleur joue le rôle d'un cerveau du réseau, il fait fonction de système d'exploitation pour ce dernier, il facilite la gestion d'une façon automatisée en offrant une vue centralisée de bout en bout du réseau et il le rend plus facile à intégrer et administrer les applications d'entreprises.

III.5.2.1 Types de contrôleurs :

Plusieurs contrôleurs ont été développés, dont la majorité sont open source et supportent le protocole OpenFlow. Ces contrôleurs diffèrent par leurs langages de programmation, la version d'OpenFlow supportée, les techniques utilisées comme les multitâches et les performances comme le débit. Le Tableau III.2 résume les caractéristiques des contrôleurs SDN les plus utilisés :

Nous présentons ci-dessous une liste non exhaustive des contrôleurs SDN : [33]

- **NOX et POX :** Sont les deux premiers contrôleurs OpenFlow. NOX est programmable en C++ tandis que POX utilise le langage Python

- **Maestro** : Utilise la technologie du multitâche pour effectuer le parallélisme au bas niveau, en gardant un modèle simple de programmation pour les développeurs

d'applications. Il atteint ses performances à travers la distribution des tâches du cœur aux threads disponibles et la minimisation de la mémoire consommée.

- **Beacon :** Est un contrôleur développé par Davide ERICSSON il est modulaire est basé sur java.

- **SNAC** : Utilise une application web pour gérer les règles du réseau. Un langage de définition des règles flexibles et des interfaces faciles à utiliser ont été intégrés pour configurer les équipements réseaux et contrôler leurs évènements.

- **RISE** : Conçu pour les expérimentations des réseaux à grande échelle, RISE est un contrôleur basé sur Trema . Ce dernier est un Framework programmé en Ruby et C. Trema fournit un environnement intégré de test et de débogage, incluant un ensemble d'outils pour le développement.

- **Ryu :** Est un contrôleur SDN capable de configurer les équipements réseaux en utilisant OpenFlow, NetConf et OF-config.

- **Floodlight :**

  Est un contrôleur de réseau défini par un logiciel, soutenu par la société Big Switch, il offre un point de gestion central pour les réseaux OpenFlow et il peut gérer des périphériques tel que openvswitch de manière transparente. Il prend également en charge une large gamme de commutateurs physiques OpenFlow, de sorte qu'il simplifie grandement la gestion du réseau. Il est sous licence Apache et écrit en java.

- **OpenDaylight :**

  Est un projet open source pris en charge par plusieurs fournisseurs de réseau. Opendaylight est une plateforme de contrôle SDN développée par java, il a été crée pour se pencher sur la problématique du contrôleur qui est un élément prépondérant d'une architecture SDN, l'objectif de ce consortium est de regrouper les efforts industriels et académiques pour développer un contrôleur réellement utilisable qui accepte de le déployer sur n'importe quelle plate-forme matérielle et système d'exploitation.

- **ONOS (Open Networking Operating System) :** C'est un système d'exploitation réseau ou contrôleur SDN récemment libéré qui se concentre sur les cas d'utilisation des fournisseurs des services, il est implémenté sur différentes couches fonctionnant comme un cluster il assure l'évolutivité, la haute disponibilité, la performance et des abstractions pour faciliter la création des applications et des services.

III.5.2.2 Comparaison entre contrôleurs les plus utilisées :

Le tableau suivant présente une étude comparative entre ces contrôleurs SDN :

Contrôleurs	NOX	POX	RYU	Floodlight	ODL	ONOS
Langage	C++	Python	Python	Java	Java	Java
Performances	Rapide	Lent	Lent	Rapide	Rapide	Rapide
Distribué	Non	Non	Oui	Oui	Oui	Oui
OpenFlow	1.0	1.0	1.0 et 1.3	1.0	1.0 et 1.3	Oui
Prix	Non	Non	Non	Non	Non	Non

Tableau III.2: Comparaison entre les contrôleurs SDN.

III.6 Fonctionnement du réseau OpenFlow : Nous pouvons résumer le fonctionnement du réseau OpenFlow par la figure suivante :

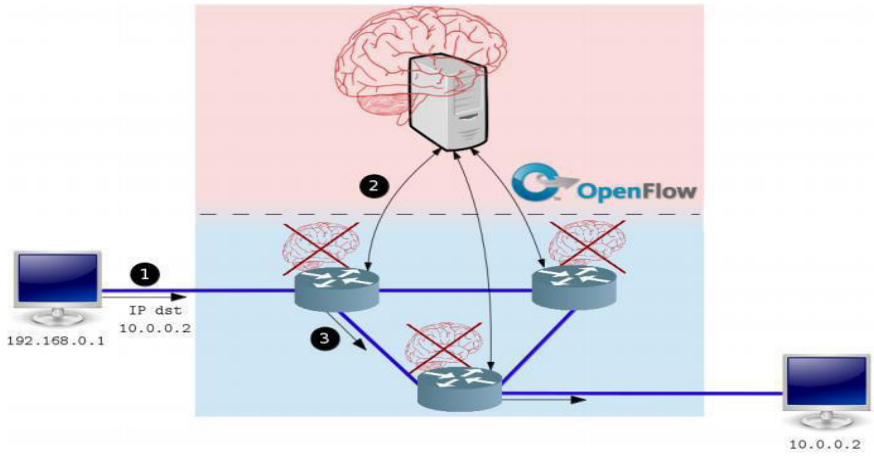


Figure III.10 : Schéma d'un simple réseau SDN.

Les switchs analysent les trames et déterminent leur interface de sortie, mais ce comportement découle des règles émises par le contrôleur. Le commutateur qui réceptionne la trame (1) signale l'évènement au contrôleur à travers un message de type « PACKET_IN », et reçoit en retour des règles au cours d'un échange par un message « PACKET_OUT » donnant l'instruction à suivre (2), afin de décider le transfert à travers quelle interface (3) vers le commutateur suivant approprié. Ce comportement est dit réactif ; un comportement proactif consisterait en la transmission des règles avant que le commutateur reçoit des trames. [34]

III.6.1 Processus de traitement des paquets dans un switch OpenFlow :

À chaque entrée de "Flow Table", est associé un ensemble d'actions (action set) à exécuter sur les paquets, avant de les envoyer vers un port de sortie. Les actions doivent être exécutées dans le même ordre dont lequel elles sont présentées dans la table.

Lorsqu'un paquet parvient dans l'interface d'entrée, l'ensemble d'opérations suivantes ont lieu : [32]

NB : Lors de la correspondance, l'entrée de flux ayant la priorité la plus élevée est traitée en premier

1. Les champs d'en-tête de paquet sont évalués par rapport à la table 0.
2. S'il n'y a pas de correspondance (y compris l'absence d'entrée de table manquante), le paquet est supprimé.
3. S'il n'y a pas de correspondance et qu'il existe une entrée de table manquante, alors exécutez l'action de suppression de table définie.

4. Si une correspondance a lieu alors,
    - Mettre à jour les compteurs.
    - Exécuter les instructions.
    - Transmis à une table plus loin dans le pipeline ou transféré hors d'un port de sortie.

Qu'est ce qu'une entrée de table-miss? L'entrée de table-manquante définit un ensemble d'actions qui sont effectuées dans le cas où aucune correspondance n'est trouvée pour un paquet. Les actions comprennent :

- Supprimer le paquet.
- Transférer le paquet sur toutes les interfaces.
- Transférer le paquet au contrôleur. Cela entraînera la création par le contrôleur de nouveaux flux pour ce trafic ou l'abandon

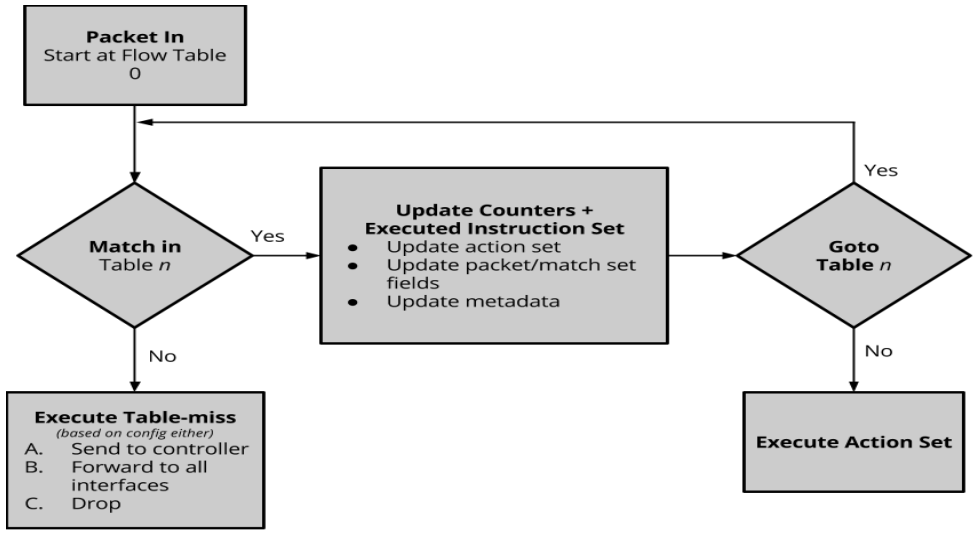


Figure III.11: Algorithme de traitement des paquets dans un réseau OpenFlow.

III.6.2 Le traitement pipeline : Chaque Switch contient plusieurs tables de Flux, et chaque table contient plusieurs entrées. Le traitement pipeline OpenFLow décrit et définit comment les paquets interagissent avec les Tables de Flux : [31]

1. Trouver la priorité la plus haute parmi les entrées de la table de flux
2. Appliquer les instructions :
   — Modifier le paquet et mettre à jour les champs de correspondance (appliquer les actions se trouvant dans les instructions.)
   — Mettre à jour l'ensemble des actions (effacer les actions et/ou les ecrire).
   — Mettre à jour les métadonnées.
3. Envoyer les données et l'ensemble des actions à la table suivante.

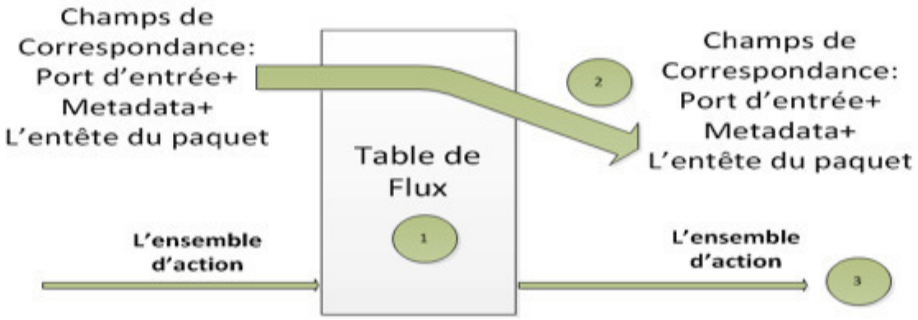


Figure III.12 : Traitement d'un paquet à travers le pipeline.

Les tables de Flux d'un Switch OpenFLow sont séquentiellement énumérées, commençant par 0.

Le traitement par Pipeline commence toujours avec la première table de Flux : on fait correspondre le paquet selon les entrées de la table de Flux 0. D'autres tables de Flux peuvent être utilisées selon le résultat obtenu lors de la première correspondance faite dans la première table de Flux.

Si le paquet correspond à une entrée dans la table de Flux, l'instruction en question sera exécutée. Les instructions dans les tables de Flux peuvent explicitement diriger le paquet vers une autre table de Flux utilisant l'instruction Goto.

Une entrée **x** dans une Table de Flux **n** peut enchainer le traitement du paquet en l'envoyant vers une autre table de Flux si seulement cette dernière dispose d'un numéro n supérieur à celui de la table où l'entrée x se trouve. Ainsi la dernière table du pipeline ne peut inclure l'instruction Goto. Si l'entrée dans la table de Flux ne redirige pas le paquet à une autre table de Flux, le traitement pipeline s'arrête. Arrivé à ce stade le paquet est traité avec l'action associée.

III.7 L'orchestrateur :

L'orchestration dans une architecture SDN est un processus de sélection optimal de ressources, avec une capacité de programmer les comportements automatisés dans un réseau, afin de coordonner les éléments matériels et logiciels de ce dernier. Le rôle principale d'un orchestrateur SDN est de manipuler les ressources d'un Datacenter tel que : les machines virtuelles, le réseau, le stockage et les applications. [35]

III.7.1 Types d'orchestrateurs : L'orchestration dans le SDN peut être classée en deux types: [35]

- **L'orchestration de services** : Pour la mise en place d'un VPN entre plusieurs sites d'une Enterprise ou la mise en place de pare-feu et des options de services IPS et IDS.

- **L'orchestration de l'infrastructure :** Utilise Openstack pour les serveurs, le stockage et les commutateurs

III.7.2 La communication inter-orchestrateurs :

Dans un modèle SDN, l'ensemble des contrôles de différentes ressources du réseau, le stockage et la manipulation des machines virtuelles sont gérées par un orchestrateur logiciel. En effet avec les extensions des fonctions réseau, le nombre important des applications du

domaine et le contrôle de ce dernier ne résident plus dans un seul orchestrateur, de ce fait les fournisseurs de services pour le SDN comprennent la réalisation d'une vision globale du réseau et mettent en œuvre une structure de contrôle distribué. Cette structure est bien que l'utilisation de plusieurs orchestrateurs gérant chacun un domaine de réseau et partageant la vision globale de ce dernier entre eux.

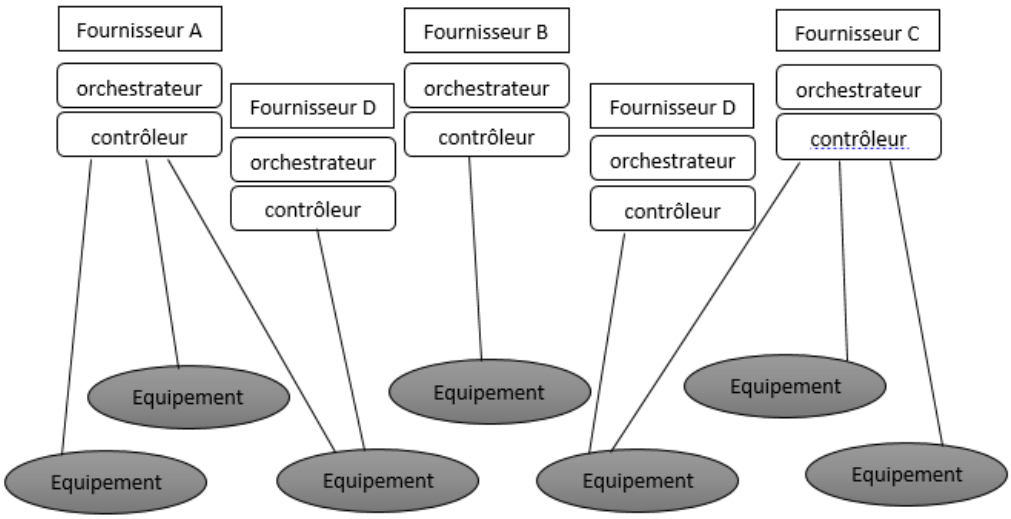


Figure III.13 : Les orchestrateurs de fournisseurs différents dans un modèle SDN.

La difficulté qui réside dans les systèmes d'orchestration est le problème de communication entre des systèmes de fournisseurs différents. Si on suppose que dans un réseau ou plusieurs systèmes d'orchestrations sont déployés, la communication entre ces dernier sera limitée c'est pour cela qu'il est préférable d'adopter la même solution de système d'orchestration dans un modelé SDN. La figure III.13 résume la communication entre les orchestrateurs des fourniers différents

III.7.3 La relation entre un contrôleur et un orchestrateur :

Dans un réseau qui utilise la technologie SDN la mission d'un contrôleur est de traduire les instructions qui sont imposées par l'orchestrateur en configuration appropriée sur

les ressources physiques et virtuelles. Donc un orchestrateur peut gérer plusieurs contrôleurs afin de délivrer des services comme cela montré sur la figure III.14

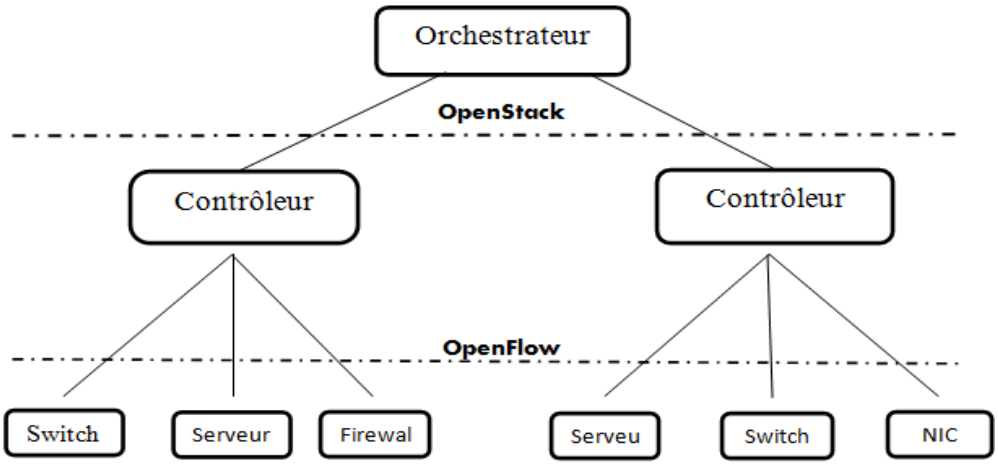


Figure III.14 : L'orchestrateur dans la technologie SDN.

Parmi les instructions que donne l'orchestrateur aux contrôleurs :

- Créer et relier une machine virtuelle à un réseau virtuel.
- Connecter une VM à un réseau externe.
- Appliquer des règles de sécurités pour un groupe de VM.
- Faire un chainage de service entre VM.

III.8 OpenStack :

OpenStack est un logiciel open source développé en python permettant de déployer les infrastructures de cloud computing de type IaaS *(Infrastructure as a Service)* sur une infrastructure existante. La technologie possède une architecture modulaire composée de

plusieurs projets corrélés qui permettent de contrôler des différentes ressources de machines virtuelles telle que la puissance de calcul et le stockage. [36]

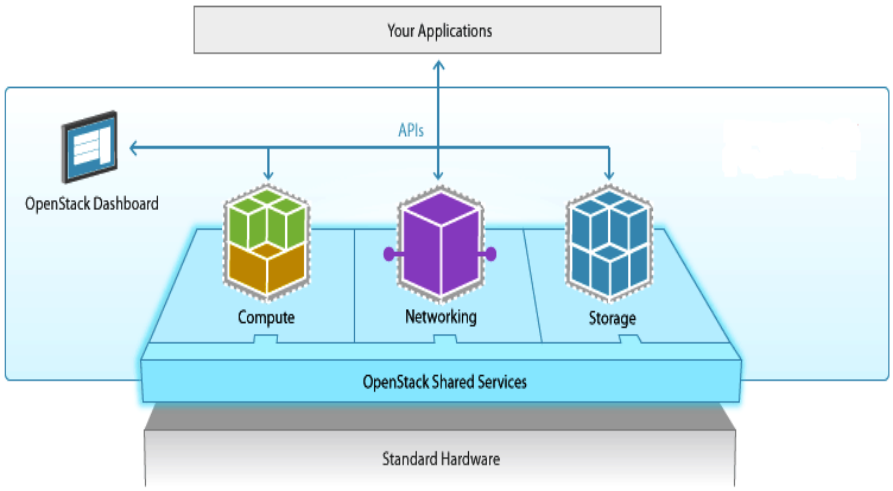


Figure III.15 : Déploiement d'OpenStack dans une infrastructure réseau.

L'ensemble de ces projets dressent des fonctionnalités complémentaires permettant de mettre en œuvre un cloud de type IaaS. Parmi les projets qui compose OpenStack :

1. **Nova :** Pour le calcul (instance virtuelle), création exposition des applications distribuées.
2. **Glance :** Pour la gestion des images d'OS à destination des instances virtuelles, exemple : Les images d'OS pour démarrer une instance virtuelle.
3. **Cinder :** pour le stockage sous forme de disque, il offre la possibilité d'ajouter des disques En cas d'insuffisance de la mémoire.
4. **Swift :** Pour le stockage orienté objet. C'est un système de stockage de données redondant et évolutif. Les fichiers sont écrits sur de multiples disques durs répartis sur plusieur serveurs dans un Datacenter.
5. **Quantum :** Pour la gestion des réseaux logiques (les Gateway, l'équilibrage de charge…).
6. **Keystone :** pour la gestion des identités, Il fournit un annuaire contenant la liste des services et la liste des utilisateurs d'Openstack ainsi que leurs rôles pour l'authentification aux services.

7. **Ceilometer :** Pour la capture et l'agrégation des métriques du system Par exemple : il permet de récolter le nombre d'instances lancé dans un projet et depuis combien de temps.
8. **Horizon :** Il s'agit d'une application web qui permet aux utilisateurs et aux administrateurs de gérer leurs nauges.

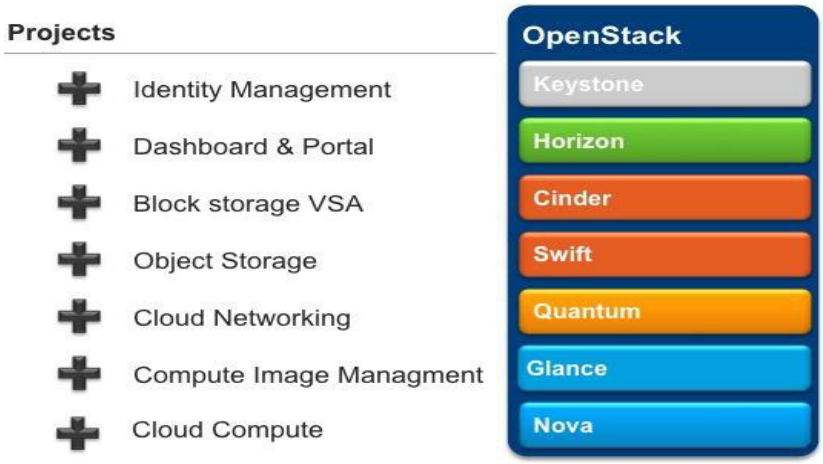


Figure III.16 : Fonctionnalités de quelques modules d'OpenStack.

III .9 Discussions :

Dans ce chapitre, nous avons décrit les concepts sur lesquels s'appuie le SDN ainsi que les différents éléments de ce dernier, comme le switch OpenFlow et son contrôleur, qui à pour objectif principale l'ajout des instructions ou règles qui permettent de programmer le plan de données, afin de disposer d'un réseau largement automatisé et optimisé pour un meilleur trafic réseau. Nous avons aussi fait une analyse comparative des contrôleurs SDN les plus utilisés et les versions d'OpenFlow qu'ils supportent.

Dans la suite, nous allons démontrer comment mettre en place un réseau SDN, en utilisant le contrôleur ONOS dans lequel l'application segment routing s'executera et qui permettra un meilleur contrôle du réseau.

Chapitre IV : Conception et réalisation

IV.1 Préambule :

Après avoir accompli toute une étude théorique sur le MPLS et le nouveau concept SDN, nous allons mettre en pratique cette combinaison sur une topologie réseau émulée.

Ce chapitre sera dédié à la description de l'application Segment-Routing qui remplace le domaine MPLS et qui présente des technologies améliorées par rapport à ce dernier et des modules qui sont intégrés dans le système ONOS, ainsi que les différents tests de fonctionnement.

IV.2 Segment-Routing :

C'est une nouvelle technologie qui veut répondre à plusieurs inconvénients d'IP/MPLS, en termes d'évolutivité, de simplicité et de facilité de fonctionnement. Segment-Routing peut être directement appliquée à l'architecture MPLS sans changement dans le plan de données.

Segment-Routing est une méthode basée sur le routage par source. Où la source spécifie le chemin et l'encode dans l'en-tête du paquet en tant que liste de segments. **[37]**
Un segment : C'est une instruction que doit exécuter le paquet, il y'a deux types :
Nodal segment : C'est l'instruction qui dirige le paquet vers la destination.
Adjacency segment : C'est l'instruction qui oriente le paquet sur un lien spécifique vers un voisin.

IV.2.1 Les avantages de segment-routing :

- L'adoption de cette technologie ne nécessite pas forcément de nouveaux équipements, il suffit souvent de mieux exploiter le potentiel de l'infrastructure existante. Segment-Routing peut être utilisé pour diriger le trafic de n'importe quel chemin arbitraire dans le réseau.
- **Evolutif :** Segment-Routing ne nécessite aucune signalisation de chemin. Par conséquent, l'état par flux ne doit être maintenu qu'au niveau du nœud d'entrée du domaine SR, ce qui augmente la flexibilité du réseau tout en réduisant les coûts.
- **Simple :** Segment-Routing fournit un contrôle complet sur les chemins en combinant des instructions (segments). Il ne nécessite aucun protocole tel que LDP et RSVP TE.

- En l'adoptant, les opérateurs télécoms peuvent créer un réseau agile et efficace, mais aussi développer leurs activités en élaborant des produits novateurs et séduisants et en les introduisant rapidement sur le marché.

Le Segment-Routing est conçu pour les réseaux d'un plan de contrôle centralisé ce qui signifie qu'il est pris en charge par le SDN.

IV.2.2 Comparaison entre IP/MPLS et Segment-Routing :

IP/MPLS	Segment-Routing
▪ En MPLS la signalisation des labels et la réservation des ressources sont effectuées par l'implémentation des protocoles de signalisation LDP et RSVP-TE. ▪ MPLS n'utilise pas des labels globaux ce qui signifie que les routeurs adjacents utilisent des labels différents pour atteindre la destination. ▪ En MPLS l'état des tunnels est maintenu dans chaque nœud que le trafic traverse. ▪ Dans les réseaux MPLS les chemins sont déterminés strictement hop by hop ce qui signifie que ECMP n'est pas pris en charge.	▪ Dans le réseau SR il suffit d'avoir un IGP et une fois Segment-Routing est configuré l'IGP prendra les labels et les distribue dans le domaine IGP. ▪ Segment-Routing réduit l'échange des labels dans le réseau en utilisant des segments globaux. ▪ Dans Segment-Routing l'état est présent seulement au niveau du routeur de tête. ▪ Dans Segment-Routing l'équilibrage de charge entre deux chemins de coût égaux entre la source et la destination est intégré, cette propriété stabilise le réseau.

Tableau IV.1 : Comparaison entre les technologies IP/MPLS et Segment-Routing.

IV.3 La description de l'application Segment-Routing :

Segment-Routing est un nouveau concept introduit par l'IETF, il peut utiliser soit le plan de données MPLS existant, soit le plan de données IPv6.
Dans notre projet, nous avons choisi d'implémenter le plan de données MPLS.

Segment-Routing partage beaucoup d'avantages communs avec le SDN, dont :

- ✓ L'élimination de protocoles distribués complexes.
- ✓ L'utilisation du routage par source via les contrôleurs ou les routeurs de tête.

Dans notre solution, le plan de données reste exactement comme celui d'un routeur Segment-Routing tel que défini au début de ce chapitre. Cependant, le plan de contrôle, n'utilise pas un IGP distribué intégré dans les routeurs. Au lieu de cela, il utilise une application de routage « Segment Routing » développée sur le système ONOS. Cette application programme les routeurs de tête et les routeurs de cœur pour le routage par défaut, le routage basé sur des policies et le comportement du réseau en cas de défaillance, par l'envoi des instructions via le protocole OpenFlow.

IV.4 L'émulateur Mininet :

Mininet est un émulateur de réseau qui permet de créer une réseau virtuel composé d'hôtes, commutateurs, routeurs, contrôleurs et des liens entre ces équipements sur un ordinateur personnel via une ligne de commande, ou avec un Script Python. Mininet fournit à l'utilisateur la capacité de créer rapidement, d'interagir, de personnaliser et de partager un prototype de réseau défini par logiciel « SDN » pour émuler une topologie. **[29]**

IV.4.1 Création d'une topologie avec la ligne de commande :

Mininet fournit, en plus de la topologie « minimal », la topologie « single », « linear » et « tree ». Pour charger l'une de ces topologies, on utilise l'option «--topo», Par exemple:

```
$sudo mn --topo=single
```

«single» tout court donne la même topologie que minimal, c'est-à-dire créer un hôte relié avec un switch, mais on peut ajouter également comme argument un chiffre, qui indique le nombre d'hôtes à créer

```
$ sudo mn --topo=single,4
```

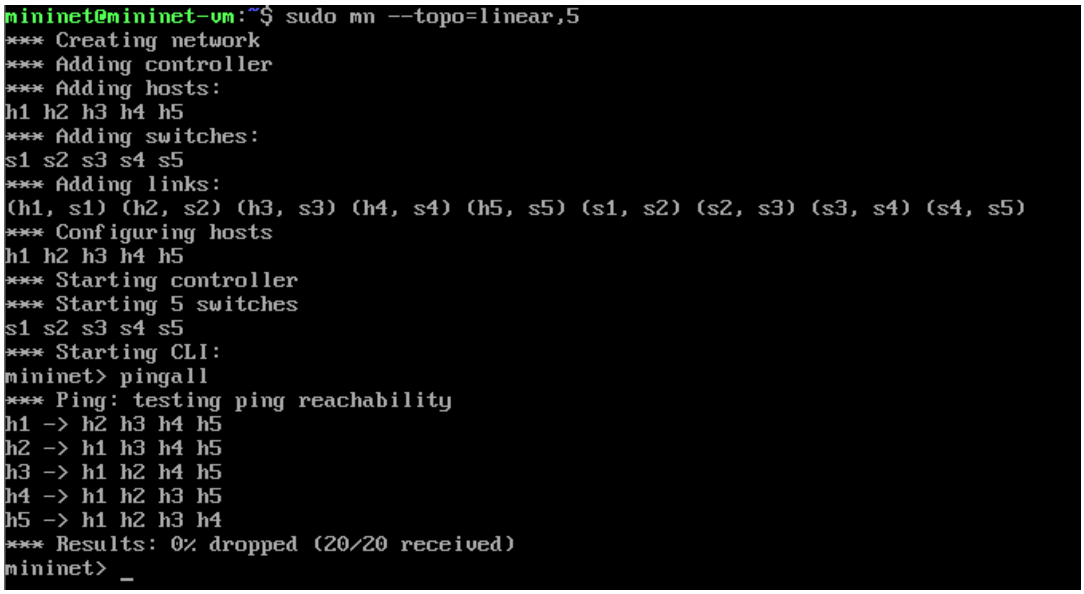


Figure IV.1 : Création d'une topologie avec Mininet.

Si nous souhaitons personnaliser une topologie déjà créée, nous appliquons une des commandes citées ci-dessous sans ligne de commande mais dans un script python par exemple :

- Ajouter un switch : self.addSwitch('S1')
- Ajouter un hôte : self.addHost(H1)
- Ajouter un lien : self.addLink(h1,s3)

Si on veut créer 2 hôtes h1, h2 et h3 les trois sont connectés à un switch s1.

```
class Test_Topo(Topo):
        def __init__(self):

            "Create P2P topology"
            # Initialize topology
            Topo.__init__(self)
            # Add hosts and switches
            h1 = self.addHost('h1')
            h2 = self.addHost('h2')
           h3=self.addHost('h3')
            s1 = self.addSwitch('s1')
            # Add links
           self.addLink(h1,s1)
```

```
                    self.addLink(h2,s1)

                    self.addLink(h3,s1)
        topos = { 'toptest': (lambda: Test_Topo()) }
```

Après, nous sauvegardons le code dans un fichier toptest.py, et nous exécutons la commande suivante:

```
$ sudo mn --custom /chemin/vers/toptest.py --topo toptest
```

NB : le contrôleur ONOS qu'on utilise dans notre simulation appuie sur des scripts python pour la création des topologies réseau, avec des commandes préprogrammées.

IV.4.2 Le commutateur CPqD :

C'est un composant essentiel dans notre émulation, c'est un commutateur virtuel que l'on utilise pour interconnecter des machines virtuelles provenant d'un hôte ou de machines distantes.

IV.5 Environnement d'expérimentation :

Dans cette section, nous fournissons une analyse des performances du contrôleur ONOS sur l'application segment-routing, nous décrivons un ensemble des tests de fonctionnement de cette dernière.

Afin de réaliser ces tests de fonctionnement sur le contrôleur ONOS, nous avons utilisé un laptop Apple, équipé d'un processeur i5 et qui utilise Mac-OS et Windows 7 comme systèmes d'exploitation, nous avons également installé dans cette machine l'outil VMware version Workstation Pro sous Windows pour importer et exécuter la machine virtuelle qui comporte :

- ✓ L'émulateur de réseau Mininet requis pour exécuter les commutateurs logiciels CPqD dans notre topologie.
- ✓ La version spécifique d'ONOS dans laquelle l'application Segment-Routing est implémentée.
- ✓ La CLI spécifique.
- ✓ La version spécifique du commutateur logiciel CPqD OF1.3 requis pour ce projet.
- ✓ Putty pour accéder à la VM via des autres terminales à travers le protocole SSH et le logiciel WinScp qui permet d'ouvrir et de modifier les fichiers installés sur la machine

concernant les VM et les différents scripts python qui permettent de créer plusieurs topologies.

IV.6 Création de la topologie :

Afin de créer notre topologie on doit ouvrir un autre terminal avec putty et on accède via le SSH à la machine virtuelle, puis on démarre l'émulateur mininet, celui-ci qui va créer la topologie propre au commutateur cpqd en exécutant le script python qui est dans le fichier suivant. **« sr_CPqD.full.py »**

```
login as: mininet
mininet@192.168.142.129's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Wed Jun 20 22:59:14 2018 from 192.168.142.1
mininet@mininet-vm:~$ cd mininet/
mininet@mininet-vm:~/mininet$ cd custom/
mininet@mininet-vm:~/mininet/custom$ sudo ./sr_cpqd_full.py
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s1) (h2, s6) (h3, s1) (h4, s7) (s1, s2) (s1, s3) (s2, s3) (s2, s5) (s2, s8)
 (s3, s4) (s4, s5) (s4, s6) (s5, s6) (s5, s8) (s7, s8)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8
```

Figure IV.2 : Création de la topologie.

Remarque : Le fichier «sr-cpqd-full.py» est le fichier de la création de notre topologie, il est crée en Python, et qui sera présenté en annexe B.

IV.6.1 Configuration de la topologie :

Au niveau du contrôleur il n'ya aucune configuration qui s'exécute donc on va importer un fichier de configuration crée en python qui est « sr-cpqd-full.conf » et qui sera présenté dans annexe B.

IV.6.2 La topologie :

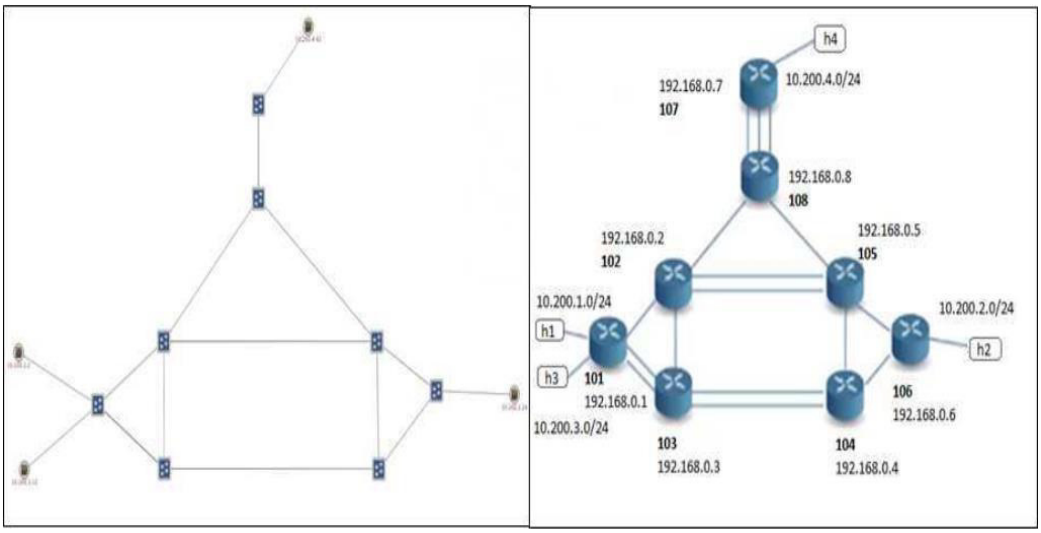


Figure IV.3 : La topologie.

La topologie est affichée par le contrôleur ONOS, Cette topologie contient :

- 8 commutateurs de type CPqD contrôlés par un contrôleur SDN.
- 4 hôtes.
- 20 liens qui relient les commutateurs et les hôtes entre eux.

IV.7 L'application segment routing dans ONOS :

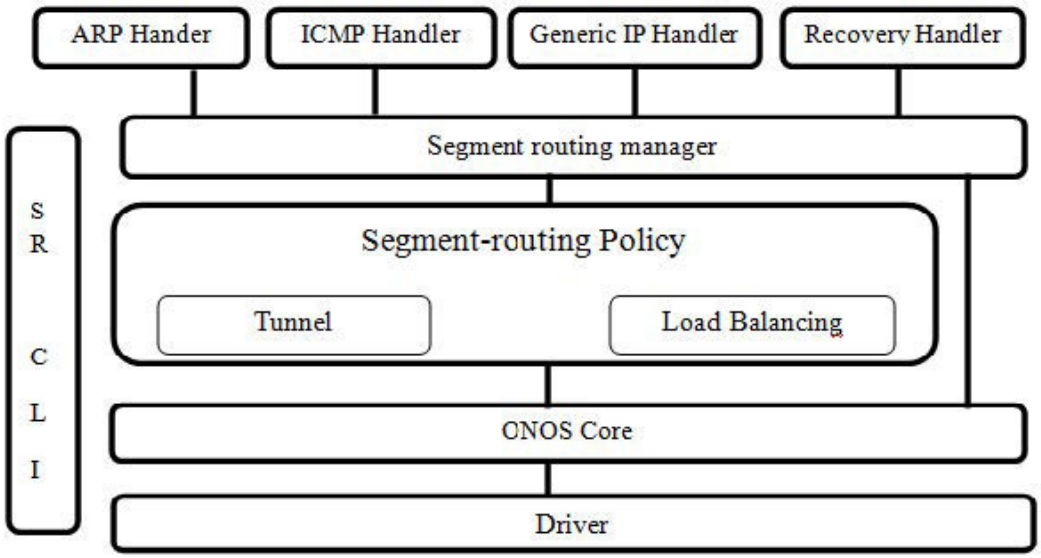


Figure IV.4 : Les modules de l'application Segment routing dans ONOS.

L'architecture ONOS est constituée de :

Driver : c'est un module qui chargé de donner les différents pipelines (TTP) aux commutateurs OpenFlow

ONOS Core : représente d'autre modules du contrôleur ONOS qui n'utilisent pas l'application segment-routing.

Segment-routing application :

- **Segment-routing manager :** Il calcule le chemin ECMP le plus court et remplit toutes les règles de routage dans les tables de routage des commutateurs. En outre, il transmet tous les paquets à des modules appropriés (handler) en fonction de type de paquet.
- **ARP Handler :** Il gère les requêtes ARP et les réponses à ces requêtes. Si la requête ARP concerne les routeurs gérés, il génère et envoie la réponse ARP aux hôtes correspondants.
- **ICMP Handler :** Il traite la requête ICMP. Il génère le paquet de réponse ICMP et l'envoie aux hôtes correspondants.
- **Generic IP Handler :** Il gère tous les paquets IP, si la destination du paquet IP est un hôte dans les sous-réseaux de routeur, il définit la règle de renvoi et envoie le paquet aux hôtes correspondants. Si l'adresse MAC de l'hôte n'est pas connue, il envoie une demande ARP au sous-réseau à l'aide d'un ARP handler.
- **Group Recovery Handler :** Ce module gère les pannes des éléments de réseau, en effectuant des opérations de modification, de remplacement ou de suppression.
- **Segment-routing policy :** Ce module crée la policie et définit la règle dans la table ACL du routeur. Si c'est une policie d'un tunnel, il crée le tunnel pour cette policie.

IV.8 Testes de fonctionnement :

Afin de bien assurer le bon fonctionnement des différents modules du contrôleur ONOS qu'on définit dans la page précédente, nous essayons de faire un ensemble des tests dans la VM qui représente ONOS et Segment-routing.

IV.8.1 Test 1 :

Pour voir que notre réseau fonctionne correctement et que les paquets peuvent être acheminés entre tous les équipements à travers toutes les routes de la topologie, nous avons

tracé des Pings vers plusieurs nœuds de réseau et nous avons constaté que le ping passe correctement.

Nous avons également lancé un ping continue entre la source « h1 » et la destination «h2 » pour savoir la manière avec laquelle le trafic traverse le réseau pour atteindre la destination.

```
mininet> h1 ping 10.200.2.24
PING 10.200.2.24 (10.200.2.24) 56(84) bytes of data.
64 bytes from 10.200.2.24: icmp_seq=1 ttl=61 time=9.72 ms
64 bytes from 10.200.2.24: icmp_seq=2 ttl=61 time=5.46 ms
64 bytes from 10.200.2.24: icmp_seq=4 ttl=61 time=3.61 ms
64 bytes from 10.200.2.24: icmp_seq=5 ttl=61 time=4.41 ms
64 bytes from 10.200.2.24: icmp_seq=6 ttl=61 time=7.33 ms
64 bytes from 10.200.2.24: icmp_seq=8 ttl=61 time=3.34 ms
64 bytes from 10.200.2.24: icmp_seq=9 ttl=61 time=10.3 ms
64 bytes from 10.200.2.24: icmp_seq=10 ttl=61 time=4.54 ms
^C
--- 10.200.2.24 ping statistics ---
11 packets transmitted, 8 received, 27% packet loss, time 10036ms
rtt min/avg/max/mdev = 3.344/6.102/10.380/2.562 ms
mininet>
```

Figure IV.5 : Ping réussi entre « h1 » et « h2 ».

Pour afficher des informations sur l'ensemble des switchs, host et lien, on doit ouvrir une autre terminale avec putty on accède à la Cli/ et à la Cli.py/ puis on va faire une des commandes suivantes : show switch, show link, show host.

Exemple :

```
mininet@mininet-vm: ~/spring-open-cli/cli
mininet-vm# sh switch
#  Switch DPID             Alias        Connected Since              Connected A
t    Type   Controller
--|----------------------|------------|--------------------------|-----------
----|------|----------
1  00:00:00:00:00:00:00:01 SFO-ER1      Tue Jun 26 15:12:42 PDT 2018 127.0.0.1:6
0761 packet mininet-vm
2  00:00:00:00:00:00:00:02 SFO-CR2      Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0507 packet mininet-vm
3  00:00:00:00:00:00:00:03 SFO-CR3      Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0502 packet mininet-vm
4  00:00:00:00:00:00:00:04 Dallas-CR4   Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0500 packet mininet-vm
5  00:00:00:00:00:00:00:05 Dallas-CR5   Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0505 packet mininet-vm
6  00:00:00:00:00:00:00:06 Dallas-ER6   Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0501 packet mininet-vm
7  00:00:00:00:00:00:00:07 NewYork-ER7  Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0506 packet mininet-vm
8  00:00:00:00:00:00:00:08 NewYork-CR8  Tue Jun 26 22:34:55 PDT 2018 127.0.0.1:6
0503 packet mininet-vm
9  00:00:00:00:00:00:00:09 NewYork-CR9  Tue Jun 26 15:12:44 PDT 2018 127.0.0.1:6
0769 packet mininet-vm
10 00:00:00:00:00:00:00:0a NewYork-ER10 Tue Jun 26 15:12:44 PDT 2018 127.0.0.1:6
```

Figure IV.6 : Affichage des switches sur la ligne commande.

Pour voir les instructions envoyées par le contrôleur SDN au commutateur de tête 101, et qu'elles doivent être appliquées au trafic entrant pour atteindre la destination « h2 », il faut il faut consulter la table IP du commutateur 101 en tapant la commande suivante :

```
Mininet-vm# Sh shwitch 00 :00 :00 :00 :00 :00 :00 :01 table IP
```

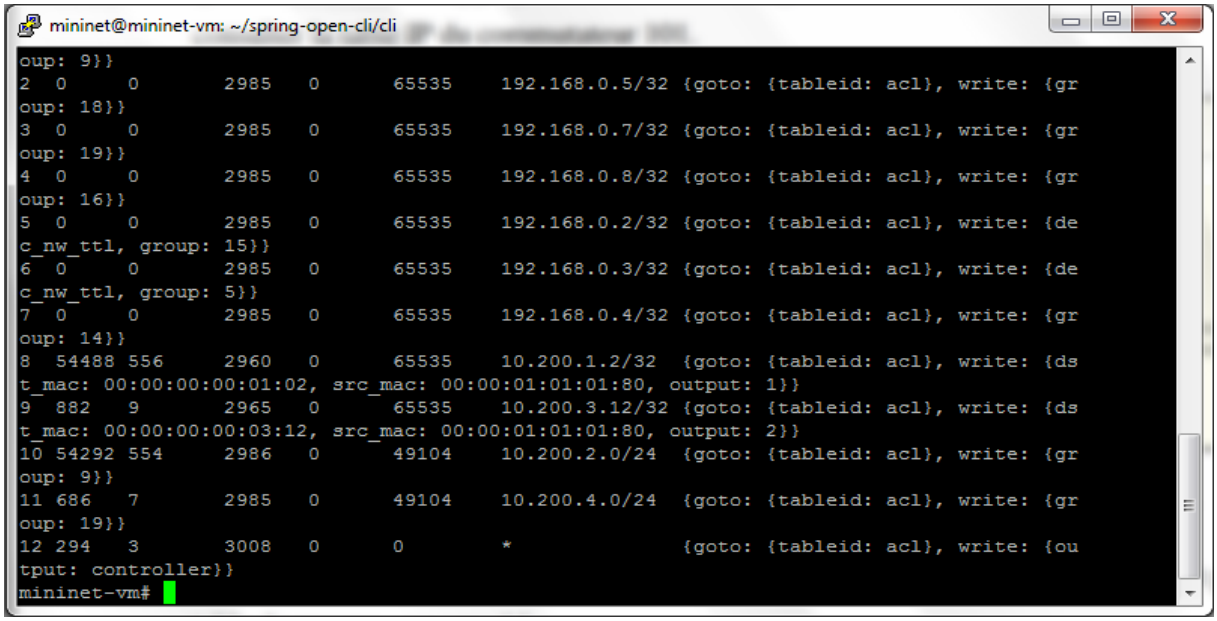


Figure IV.7: La table IP du commutateur 101.

D'après cette table, pour atteindre la destination 10.200.2.0/24, le trafic doit passer par la table de groupe « groupe 9 ».

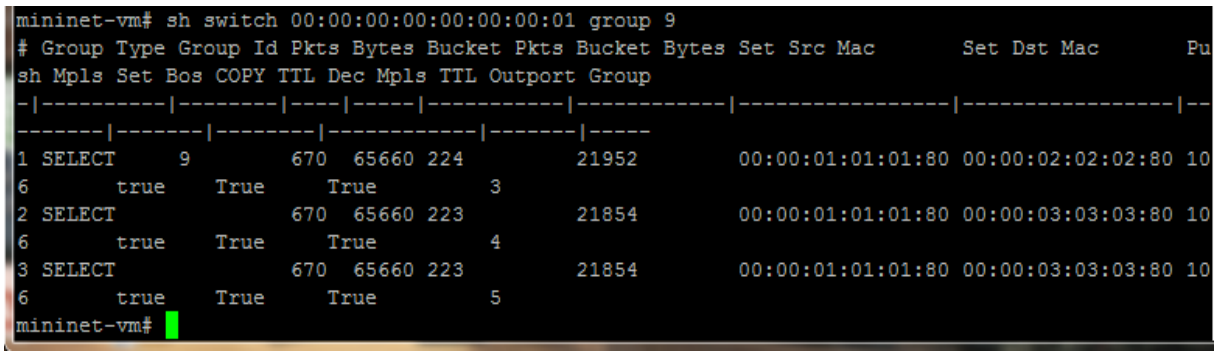


Figure IV.8 : Les instructions du groupe « 9 ».

Le groupe 9 présente 3 Buckets de types SELECT, pour les trois ports sortants 3,4 et 5 ce qui signifie : envoyer le trafic en le partageant équitablement sur les trois ports. Le contrôleur a utilisé le module « Segment routing manager » qui prends en charge l'ECMP, donc le groupe 9 est un groupe d'ECMP qui partage la charge sur les trois liens.

IV.8 .2Test 2 :

Nous avons fait tomber un lien parmi les trois liens sortants du commutateur 101 en utilisant la commande suivante :

```
mininet> s1 ifconfig s1-eth4 down
mininet>
```

Figure IV.9: Lien défaillant.

Nous avons remarqué que le ping ne s'arrête pas, ce qui signifie que le transfert des paquets ne s'arrête pas, donc une convergence totale a été entrainé dans le réseau.

```
mininet> s1 ifconfig s1-eth4 down
mininet> h1 ping h2
PING 10.200.2.24 (10.200.2.24) 56(84) bytes of data.
64 bytes from 10.200.2.24: icmp_seq=1 ttl=61 time=3.47 ms
64 bytes from 10.200.2.24: icmp_seq=2 ttl=61 time=3.15 ms
64 bytes from 10.200.2.24: icmp_seq=3 ttl=61 time=4.48 ms
64 bytes from 10.200.2.24: icmp_seq=4 ttl=61 time=3.82 ms
64 bytes from 10.200.2.24: icmp_seq=5 ttl=61 time=2.74 ms
64 bytes from 10.200.2.24: icmp_seq=6 ttl=61 time=2.96 ms
64 bytes from 10.200.2.24: icmp_seq=7 ttl=61 time=3.98 ms
64 bytes from 10.200.2.24: icmp_seq=8 ttl=61 time=4.26 ms
```

Figure IV.10 : Ping continue entre « h1 » et « h2 ».

Cette fois-ci le groupe 9 ne présente que 2 Buckets, ce qui signifie que le contrôleur SDN a retiré le lien qui a tombé en panne, et il a utilisé les deux liens restants pour le transfert de données.

IV.8.3 Test 3 :

Nous avons vu comment gérer le réseau Segment-Routing, et comment changer son comportement par défaut, nous avons créé un tunnel « t1 », et nous avons également appliqué une policie qui dirige des flux de trafic spécifiques dans ce tunnel.

```
mininet-vm# configure
mininet-vm(config)# tunnel t1
mininet-vm(config-tunnel)# node 101
mininet-vm(config-tunnel)# node 105
mininet-vm(config-tunnel)# node 106
mininet-vm(config-tunnel)#
```

Figure IV.11 : Création d'un tunnel « t1 »

A ce niveau il a y'a aucun paquet qui passe par ce tunnel, et le trafic se partage toujours sur les trois liens pour atteindre la destination. Pour diriger des flux de trafic dans ce tunnel il faut créer une policie pour ce tunnel.

```
mininet-vm(config)# policy p2 policy-type tunnel-flow
mininet-vm(config-policy)# flow-entry ip 10.200.1.0/24
mininet-vm(config-policy)# flow-entry ip 10.200.1.0/24 10.200.2.0/24
mininet-vm(config-policy)# tunnel t2
mininet-vm(config-policy)# priority 2000
mininet-vm(config-policy)# ex
mininet-vm(config)#
```

Figure IV.12: Création d'une policie « p1 »

Résultat : le trafic passe seulement par le tunnel « t1 » comme montre la figure suivante

```
27 INDIRECT   21       0    0     0          0                                          105
     false   True                 3
28 SELECT     22       0    0     0          0          00:00:01:01:01:80 00:00:02:02:02:80 106
     true    True     True               21
mininet-vm(config)#
```

Figure IV.13 : Trafic passe par « t1 »

IV.8.4 Test 4 :

Nous avons vu comment un administrateur réseau peut effectuer du trafic engineering en créant des tunnels Segment-Routing strictement définis. En utilisant Adjacency SIDs qui sont localement significatifs pour sélectionner une des interfaces sortantes du routeur et forcer le trafic de sortir à travers cette interface. Alors nous avons commencé par la création d'un tunnel en combinant les AdjacencySIDs et les Nodes-SIDs pour forcer le trafic à passer par un lien spécifique d'un commutateur.

```
mininet-vm(config)# tunnel t2
mininet-vm(config-tunnel)# node 101
mininet-vm(config-tunnel)# node 102
mininet-vm(config-tunnel)# node 103 adjacency
103001  103002  103003  103004  103005
mininet-vm(config-tunnel)# node 103 adjacency 103005
mininet-vm(config-tunnel)# node 104
mininet-vm(config-tunnel)# node 106
mininet-vm(config-tunnel)# exit
mininet-vm(config)#
```

Figure IV.14: Création d'un tunnel « t2 ».

Cette fois-ci nous avons effectué une priorité plus élevée pour forcer le trafic de prendre le tunnel « t2 » au lieu du tunnel « t1 ».

Ainsi nous avons créé une policie « p2 » qui dirige les flux de trafic dans le tunnel « t2 ».

```
mininet-vm(config)# policy p2 policy-type tunnel-flow
mininet-vm(config-policy)# flow-entry ip 10.200.1.0/24
mininet-vm(config-policy)# flow-entry ip 10.200.1.0/24 10.200.2.0/24
mininet-vm(config-policy)# tunnel t2
mininet-vm(config-policy)# priority 2000
mininet-vm(config-policy)# ex
mininet-vm(config)#
```

Figure IV.15 : Création d'une policie « p2 ».

Résultat : Une policie de trafic engineering s'est appliquée au trafic et ce dernier a pris exactement le chemin défini avec Adjacency-SID.

```
27 INDIRECT   21      0    0     0          0                                          105      false
   True                3
28 SELECT     22      0    0     0          0          00:00:01:01:01:80 00:00:02:02:02:80 106      true
   True     True                21
29 INDIRECT   23      0    0     0          0                                          103      false
   True                3
30 INDIRECT   24      0    0     0          0                                          103005   false
   True                          23
31 SELECT     25      0    0     0          0          00:00:01:01:01:80 00:00:02:02:02:80 106      true
   True     True                24
mininet-vm(config)#
```

Figure IV.16: Trafic pass par « t2 ».

IV.9 Discussion :

Au cours de ce chapitre nous avons attient le but de ce projet, qui est l'étude pratique d'un contrôle centralisé pour un réseau segment- routing qui est une solution logiciel intelligente. Cette dernière est une application qui remplace tout un domaine MPLS et qui permet de régler les problèmes de capacité matérielle tout en simplifiant l'infrastructure.

Nous avons commencé par la création de la topologie et puis nous avons fait un ensemble des tests de fonctionnement, dans lesquels nous avons utilisé un contrôleur ONOS pour tester le comportement par défaut du réseau. Par la suit nous avons montré la convergence automatique du contrôleur et sa capacité de trouver des solutions en cas d'échec exemple : lorsque on a fait tomber un lien en panne.

En fin nous avons effectué un changement de comportement par défaut du réseau par la création des tunnels.

Conclusion

Les réseaux d'operateurs téléphoniques *(IP/MPLS)* pensaient avoir atteint une certaine stabilité. L'arrivé des architectures SDN (*Software-Defined Networking*) modifient le paysage de l'industrie des réseaux et remet en cause un grand nombre de principes qui semblaient bien établis. Par opposition aux réseaux traditionnels, le paradigme *SDN* prône une séparation des fonctions de contrôle et de celles de transfert de données ce qui permet de déployer des nouveaux services de manière beaucoup plus rapide et avec des coûts significativement réduits. Il changerait complètement l'écosystème des infrastructures Télécoms dans les années à venir.

Les travaux menés durant ce projet se positionnent dans le cadre de la découverte des réseaux SDN et leurs concepts afin d'optimiser le fonctionnement des réseaux IP/MPLS et cela en mettant l'accent sur le protocole OpenFlow, ainsi qu'un des composants clés de cette technologie qui est le contrôleur. Dans notre cas nous avons choisi un des contrôleurs qui ont séduit les leaders du monde des télécommunications, il s'agit d'ONOS. Ce contrôleur présente plusieurs applications parmi eux nous nous sommes intéressés à l'application segment-routing qui ce veut être une amélioration du MPLS en répondant à plusieurs inconvénients dans ce dernier en terme d'évolutivité, de simplicité et de facilité de fonctionnement.

Les résultats de notre projet étaient t satisfaisants, et nous pouvons dire que nous avons atteint nos objectifs qui sont :

- Surpasser les défauts des réseaux traditionnels.
- Aborder une technologie qui facilite les réseaux IP/MPLS et permet l'adoption de SDN.

Références bibliographiques

[1] André VAUCAMPS, "protocole et concept de routage", CISCO.

[2] F. Nolot, 2009 « Modélisation hiérarchique du réseau ».Académie Cisco, Reims.

[3] S. BELLALI, « Conception et déploiement d'une solution SDN au sein d'un réseau IP/MPLS » université USTHB, 2016.

[4] Cisco Networking Acadimy, « principes de routage » CCNA 2 Version 6 Chapitre 1.

[5] G. Malikin, « RIP Version 2 », Request for Comments : 2435, 1998.

[6] J. Moy, « OSPF Version 2 », Request for Comments: 2328. 1998.

[7] Y. Rekhter et all « A Border Gateway Protocol 4 (BGP-4) », Request For Comments, 1771, 1995.

[8] Architecture du MPLS : http://igm.univ-mlv.fr/~dr/XPOSE2006/marot/architecture.html.

[9] E. Rosen, G. Fedo rkow, Y. Rekhter, D. Tappan, D.Farinacci, T.Li, A. Conta, "MPLS Label Stack Encoding", Request for Comments : 3032, 2001.

[10] L. Andersson, I. Minei, B. thomas, « LDP Specification », Request for Comments: 5036, 2007.

[11] Y. Karkab, 2014, « les réseaux IP/MPLS » principe du MPLS.

[12] G. Pujolle, 2005 « Contrôle dans les réseaux IP » Lavoisier, Paris.

[13] K. Hess et A. Newman « La virtualisation en pratique », Pearson.

[14] Documentation VMware: http://www.vmware.com/ca/fr/virtualisation/how-it-works.html.

[15] Cisco Networking Acadimy « Les réseaux VLANs » CCNA 2 Version 6, Chapitre 6.

[16] L. Anderson, T. Madsen, « Provider provisioned virtual privat network (VPN) terminology », Request For Comments: 4026, 2005.

[17] https://docs.oracle.com/cd/E37927_01/html/E36562/gfkbw.html#scrolltoc.

[18] G.pujolle, 2014, « Gestion de la virtualisation réseau et des ressources réseau dans Oracle Solaris 11.2 »

[19] M. Azzakhmam, 2017 « Virtualisation des réseaux sans fil: stratégie optimisée de déploiement progressif », Montréal.

[20] N.M. Mosharaf et all, « Network Virtualization: State of the Art and Research Challenges». Université de Waterloo.

[21] S. Das, « An unified control architecture for packet and circuit network convergence », Stanford University , Stanford-USA, 2012.

[22] Livre blanc de ONF, « Software-Defined networking: The New Norm for Networks », 2012

[23] Livre blanc de CITRIX, « SDN 102: Le SDN et le rôle des services réseaux de mise à disposition d'application».

[24] Livre blanc de l'ONF, « SDN architecture », 2014.

[25] F. BENAMRANE « Etude des performances des applications du plan de contrôle des réseaux SDN » thèse doctorat, université de Mohammed, Rebat, 2017.

[26] A. Oumeddi, 2016 « Installation et mis en œuvre d'une solution SDN dans un système ONOS », INPTIC.

[27] Open Network Foundation**,** « OpenFlow Switch spécification », version 1.4.0, Octobre 2013.

[28] S. Znaty, **«** Le Protocole OpenFlow dans l'Architecture SDN **»,** Efort éditions, 2016.

[29] O. Ifakren, *«* Software Defined Network *»*, mémoire de PFE licence, Haute école du paysage d'ingénierie et d'architecture de Genève*,* 2016.

[30] V. Cloarec, « Le projet OpenFlow », mémoire de PFE master, Telecom Lille 1, 2013.

[31] Open Network Foundation, « OpenFlow Switch Specification », version 1.4.0, Octobre 2013.

[32] Qu'est-ce que c'est OpenFlow. http:fir3net.com/Networkig/protocole/What is OpenFlow.html.

[33] A. Merakchi, « Analyse de sécurité des contrôleurs SDN », mémoire de PFE Master,

Université d'Evry Val d'Essonne, 2015.

[34] M. Tury, « Les risques d'OpenFlow et du SDN », ANSSI, 2014.

[35] J. Metzler et A. Metzler « The 2015 guid to SDN and NFV » nuage network 2015.

[36] DocuStack, https://fr.wikipedia.org/wiki/OpenStack#Calcul.

[37] «Slideshare, CiscoPublicSector, segment-routing », Mar 9, 2015

Annexes

Annexe A :

1. Installation de la VM :

1.1. Installation du mininet :

Pour installer mininet, il faut utiliser la commande présentée dans la figure suivante :

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

Figure A.1 : Téléchargement de Mininet.

La commande git télécharge la version la plus récente du Mininet, comme montre la figure :

```
cd mininet
git tag  # list available versions
git checkout -b 2.2.1 2.2.1  # or whatever version you wish to install
cd ..
```

Figure A.2 : Version de Mininet.

Une fois nous avons accédé au dossier mininet, la commande illustrée dans la figure suivante permet d'installer tous les packages Mininet.

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

Figure A.3 : Installation du Mininet.

1.2. Installation d'ONOS :

Téléchargement du code source de la version d'ONOS dans laquelle l'application Segment routing est implémentée :

```
sudo apt-get install openjdk-7-jdk openjdk-7-doc openjdk-7-jre-lib
git clone https://gerrit.onosproject.org/spring-open
```

Figure A.4 : Téléchargement d'ONOS.

Compilation d'ONOS :

```
mvn clean
mvn compile
```

Figure A.5 : Compilation d'ONOS.

Pour lancer le contrôleur il faut exécuter la commande suivante :

```
mininet@mininet-vm:~/spring-open$ ./onos.sh start
```

Figure A.6 : lancement d'ONOS.

I.3 Installation de la CLI :

Téléchargement du code source de la CLI :

```
git clone https://gerrit.onosproject.org/spring-open-cli
```

Figure A.7 : Téléchargement de la Cli.

L'exécution du code :

```
cd spring-open-cli
./setup.sh
```

Figure A.8 : Exécution de la CLI.

A ce niveau-là, nous pouvons lancer la CLI :

```
$ git pull
$ source ./workspace/ve/bin/activate
(ve)$ make start-sdncon
(ve)$ cd cli/
(ve)$ ./cli.py
version200
default controller: 127.0.0.1:8000, SDN OS 1.0 - custom version
> enable
#
```

Figure A.9 : Lancement de la Cli.

La figure suivante présente le contrôleur ONOS affiché dans une console

Figure A.7 : Affichage d'ONOS.

Annexe B :

Le code source du fichier « sr-cpqd-full.py » :

```
#!/usr/bin/env python
'''
Script to connect 8 router & 4 host topo


'''

from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.node import UserSwitch, RemoteController, OVSSwitch
from mininet.topolib import TreeNet
from mininet.topo import SingleSwitchTopo
from mininet.net import Mininet
from functools import partial
from srTopo8 import SRTopo
from alterableNet import alterableCLI
from alterableNet import alterNet

def setDefaultRoute(node, ip, intf=None):
    """Modified node.setDefaultRoute that sets a default gateway IP.

    Example call:
    /sbin/route add -net 0.0.0.0 gw 1.1.1.1 eth0

    ip: string
    intf: interface string
    """
    if not intf:
        intf = node.defaultIntf()
    #node.cmd( 'ip route flush root 0/0' )
    #node.cmd( 'route add default %s' % intf )
    node.cmd('route add -net 0.0.0.0 gw %s %s' % (ip, intf))

if __name__ == '__main__':
    setLogLevel( 'info' )
    topo = SRTopo()
    # load the topology, use the cpqd1.3 switch and point to a controller
running
    # in the this VM
    net = alterNet(topo=topo, switch=UserSwitch,
controller=partial(RemoteController,ip='127.0.0.1'))

    #adding extra links between routers
    s1, s2, s3, s4, s5, s6, s7, s8 = net.switches
    net.addLink( s7, s8 )
    net.addLink( s7, s8 )
    net.addLink( s3, s1 )
    net.addLink( s2, s5 )
    net.addLink( s3, s4 )
```

```
net.start()

# end-host configuration
h1, h2, h3, h4 = net.hosts

# host h1 is attached to router s1
h1.setIP("10.200.1.2/24")
h1.setMAC("00:00:00:00:01:02")
setDefaultRoute(h1, "10.200.1.1")

# host h2 is attached to router s6
h2.setIP("10.200.2.24/24")
h2.setMAC("00:00:00:00:02:24")
setDefaultRoute(h2, "10.200.2.1")

# host h3 is attached to router s1
h3.setIP("10.200.3.12/24")
h3.setMAC("00:00:00:00:03:12")
setDefaultRoute(h3, "10.200.3.1")

# host h4 is attached to router s7
h4.setIP("10.200.4.42/24")
h4.setMAC("00:00:00:00:04:42")
setDefaultRoute(h4, "10.200.4.1")

alterableCLI(net)
net.stop(
```

Le code source du fichier « sr-cpqd-full.conf » :

```
{
  "comment": " Multilayer topology description and configuration",
  "restrictSwitches": true,
  "restrictLinks": true,

  "switchConfig":
             [
                 { "nodeDpid": "00:01:00:01:e8:8b:93:68", "name": "SFO-
ER101", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.1/32",
                             "routerMac": "00:01:e8:8b:93:6b",
                             "nodeSid": 101,
                             "isEdgeRouter": true,
                            "subnets": [
                                        { "portNo": 25, "subnetIp":
"10.200.1.1/24" },
                                        { "portNo": 29, "subnetIp":
"10.200.3.1/24" }
                                        ]
                             }

                 },
                 { "nodeDpid": "00:01:00:01:e8:8b:93:9b", "name": "SFO-
CR102", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.2/32",
                             "routerMac": "00:01:e8:8b:93:9e",
                             "nodeSid": 102,
                             "isEdgeRouter": false
                             }
                 }
                 ,
                 { "nodeDpid": "00:01:00:01:e8:8b:93:8c", "name": "SFO-
CR103", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.3/32",
                             "routerMac": "00:01:e8:8b:93:8f",
                             "nodeSid": 103,
                             "isEdgeRouter": false
                             }
                 }
                 ,
                 { "nodeDpid": "00:01:00:01:e8:8b:93:ad", "name": "DAL-
CR104", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.4/32",
                             "routerMac": "00:01:e8:8b:93:b0",
                             "nodeSid": 104,
                             "isEdgeRouter": false
```

```
                              }
                 }
                 ,
                 { "nodeDpid": "00:01:00:01:e8:8b:93:bc", "name": "DAL-
CR105", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.5/32",
                             "routerMac": "00:01:e8:8b:93:bf",
                             "nodeSid": 105,
                             "isEdgeRouter": false
                             }
                 }
                 ,
                 { "nodeDpid": "00:01:00:01:e8:8b:93:c2", "name": "DAL-
ER106", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.6/32",
                             "routerMac": "00:01:e8:8b:93:c5",
                             "nodeSid": 106,
                             "isEdgeRouter": true,
                            "subnets": [
                                        { "portNo": 31, "subnetIp":
"10.200.2.1/24" }
                                      ]
                             }
                 },

                 { "nodeDpid": "00:01:00:01:e8:8b:93:98", "name": "NYC-
ER107", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.7/32",
                             "routerMac": "00:01:e8:8b:93:9b",
                             "nodeSid": 107,
                             "isEdgeRouter": true,
                             "adjacencySids": [
                                              { "adjSid": 77777, "ports":
[ 14 ,8 ] }
                                              ],
                            "subnets": [
                                        { "portNo": 3, "subnetIp":
"10.200.4.1/24" }
                                      ]
                             }
                 },

                 { "nodeDpid": "00:01:00:01:e8:8b:27:e3", "name": "NYC-
CR108", "type": "Router_SR", "allowed": true,
                 "latitude": 80.80, "longitude": 90.10,
                 "params": { "routerIp": "192.168.0.8/32",
                             "routerMac": "00:01:e8:8b:27:e6",
                             "nodeSid": 108,
                             "isEdgeRouter": false
                             }
```

```
                }

              ],

  "linkConfig":[
               { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:68", "nodeDpid2":
"00:01:00:01:e8:8b:93:9b",
                 "params": { "port1": 4, "port2": 4 }
                 },

               { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:9b", "nodeDpid2":
"00:01:00:01:e8:8b:93:bc",
                 "params": { "port1": 11, "port2": 4 }
                 },

               { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:9b", "nodeDpid2":
"00:01:00:01:e8:8b:93:bc",
                 "params": { "port1": 8, "port2": 48 }
                 },

                 { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:c2", "nodeDpid2":
"00:01:00:01:e8:8b:93:bc",
                 "params": { "port1": 7, "port2": 5 }
                 },

                 { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:8c", "nodeDpid2":
"00:01:00:01:e8:8b:93:68",
                 "params": { "port1": 12, "port2": 3 }
                 },


                 { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:8c", "nodeDpid2":
"00:01:00:01:e8:8b:93:ad",
                 "params": { "port1": 48, "port2": 8 }
                 },

                 { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:8c", "nodeDpid2":
"00:01:00:01:e8:8b:93:ad",
                 "params": { "port1": 7, "port2": 4 }
                 },

                { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:ad", "nodeDpid2":
"00:01:00:01:e8:8b:93:bc",
                 "params": { "port1": 10, "port2": 10 }
                 },
```

```
              { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:9b", "nodeDpid2":
"00:01:00:01:e8:8b:93:8c",
                 "params": { "port1": 10, "port2": 4 }
                 },

                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:ad", "nodeDpid2":
"00:01:00:01:e8:8b:93:c2",
                 "params": { "port1": 5, "port2": 4 }
                 },


                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:9b", "nodeDpid2":
"00:01:00:01:e8:8b:27:e3",
                 "params": { "port1": 7, "port2": 5 }
                 },

                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:bc", "nodeDpid2":
"00:01:00:01:e8:8b:27:e3",
                 "params": { "port1": 11, "port2": 10 }
                 },

                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:27:e3", "nodeDpid2":
"00:01:00:01:e8:8b:93:98",
                 "params": { "port1": 2, "port2": 2 }
                 },

                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:27:e3", "nodeDpid2":
"00:01:00:01:e8:8b:93:98",
                 "params": { "port1": 14, "port2": 8 }
                 },

                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:27:e3", "nodeDpid2":
"00:01:00:01:e8:8b:93:98",
                 "params": { "port1": 18, "port2": 14 }
                 },

                  { "type": "pktLink", "allowed": true,
                 "nodeDpid1": "00:01:00:01:e8:8b:93:68", "nodeDpid2":
"00:01:00:01:e8:8b:93:8c",
                 "params": { "port1": 10, "port2": 22 }
                 }

                 ]
                 }
```

Résumé

Le SDN – Software-Defined Networking – est certainement le sujet chaud qui agite le monde du réseau depuis ces dernières années. Le SDN est reconnu aujourd'hui comme une architecture permettant d'ouvrir le réseau aux applications. On est donc loin de la définition rigide initiale dans laquelle il était seulement question de dissocier les plans de contrôle et de données. Openflow n'est donc qu'une composante du SDN et le marché semble aujourd'hui davantage réfléchir sur les solutions offrant réellement la programmation et la simplification des infrastructures. A ce niveau plusieurs modèles de SDN se développent en parallèle pour mieux répondre à des usages spécifiques.

Les travaux portent globalement sur la programmabilité intrinsèque des équipements (nouveaux composants programmables…), les contrôleurs SDN et orchestrateurs qui ont pour mission de fournir une couche d'abstraction du réseau, et la virtualisation des fonctions réseau qui vise à s'affranchir partiellement de la complexité du réseau physique sous-jacent. Le Software Defined Networking offrira de nouveaux usages. L'enjeu pour les administrateurs réseau est d'accompagner cette nouvelle étape afin de pouvoir tirer profit de ces nouvelles capacités.

Mots-clefs

SDN, API, Contrôleur, Openflow, Overlay, NFV,ONOS et Orchestration.