الجـــــــــــــــــــمهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Télécommunications

Spécialité : Réseaux et Télécommunications

Par : MEKIDICHE Fatima Zahra & MEHADJI Esma

Sujet

Étude et implémentation des technologies MPLS & OpenFlow (SDN)

Soutenu publiquement, le 08 / 06 / 2023 , devant le jury composé de :

M.BAHRI Sidi Mohammed	Maitre de conférences B	Université de Tlemcen	Président
Mme. SLIMANE Zohra	Maitre de conférences A	Université de Tlemcen	Examinatrice
M.MERZOUGUI Rachid	Prof	Université de Tlemcen	Encadreur
M.MEHIAOUI Sid Ahmed	Ch.Div.Dév Réseau Core	Algérie Télécom	Co-Encadreur

Année universitaire : 2022 /2023

REMERCIEMENT

Louanges à **ALLAH**, qui nous a accordé la santé, la possibilité et la volonté d'entreprendre et de poursuivre nos études, et que le salut et la paix soit sur **Mohammed** le messager d'**ALLAH**

Nous remercions également, notre Encadrant, **M**. **MERZOUGUI Rachid**, d'avoir accepté de nous encadrer tout au long de notre travail, sa disponibilité et ses conseils précieux afin de présenter parfaitement notre projet de fin d'étude.

Nous exprimons aussi notre sincère gratitude pour **M**. **MEHIAOUI Sid Ahmed**, pour sa précieuse contribution en tant que Co-Encadreur et sa disponibilité pour discuter, répondre à nos questions et guider nos recherches dans notre travail.

Nous sommes reconnaissantes de l'honneur que nous ont fait **M**. **BAHRI Sidi Mohammed** en qualité de président du jury et **Mme**. **SLIMANE Zohra** en qualité d'examinatrice d'avoir porté un intérêt particulier pour notre travail et avoir prodigués leurs conseils et remarques précieux.

Nous tenons à exprimer notre chaleureux remerciement à nos parents et à nos familles. Leurs encouragements permanents et leur confiance inébranlable qui ont été d'une valeur inestimable pour nous.

Merci infiniment pour votre soutien.

بسم الله الرحمن الرحيم

الحمد لله الذي بفضله تتم الصالحات، الحمد لله الذي يسَّر البدايات وأكمل النهايات وأبلغنا الغايات، الحمد لله على الكمال - ولله الكمال وحده- وحسن الختام.

اللَّهم صَّل وسلم على خير الأنام، رجونا الكريم وقد وثقنا بصنعه وما كان من يرجو الكريم يخيب.

إهداء الى من كانت دعواتهم الصَّادقة سر نجاحنا، إلى آبائنا الأحبة وأمهاتنا الغاليات، إلى إخوتنا وأخواتِنا وكل فرد من أفراد عائلاتنا من قريب أو من بعيد كان يشجعنا ويدعمنا، إلى أساتذتنا الذين قدَّموا لنا علما نافعا طيلة مشوارنا الدراسي لعلَّه يكون شفيعا وصدقة جارية لنا ولهم.

وآخر دعوانا أن الحمد لله الذي وفقنا لهذا وَمَا تَوْفِيقِي إِلاَّ بِاللّهِ عَلَيْهِ تَوَكَّلْتُ وَإِلَيْهِ أُنِيب.

Table des matières

Liste des Tableaux

Chapitre I

Chapitre II

Chapitre III

Liste des abréviations

A
ACK: Acknowledgment
API: Application Programming Interfaces
AS: Autonomous System
AToM: Any Transport over MPLS
ATM: Asynchronous Transfer Mode
B
BGP: Border Gateway Protocol
BoS: Bit of Stack
C
CE: Customer Edge
CLI: Command Line Interface
CPU: Central Processing Unit
D
DR: Designated Router
E
EGP: Exterior Gateway Protocol
EIGRP: Enhanced Interior Gateway Routing Protocol
Exp: Experimental
F
FAI : Fournisseur d'Accès à Internet
FEC: Forwarding Equivalence Class
FIB: Forwarding Information Base
G
GUI: Graphical User Interface
H
HP: Hewlett-Packard
I
IBM: International Business Machines
ICMP: Internet Control Message Protocol
IDS: Intrusion Detection System
IGP: Interior Gateway Protocol
IPv4 /6 : internet Protocol Version 4/6
IP : Internet Protocol
ISO: International Organization for Standardization
IS-**I**S: Intermediate System to Intermediate System
IT: Information Technology
K
KVM: Kernel-based Virtual Machine
L
LAN: Local Area Network
LDP: Label Distribution Protocol
LER: Label Edge Route
LFIB: Label Forwarding Information Base
LIB: Label Information Base
LLDP: Link Layer Discovery Protocol
LSA: Link State Advertisement
LSP: Label Switching Path
LSR: Label Switch Router
L2 /3: Layer 2 /3
M
MAC: Media Access Control
MAN: Metropolitan Area Network
MIT: Massachusetts Institute of Technology
MP-BGP: Multi-Protocol Border Gateway Protocol
MPLS: Multi-Protocol Label Switching
N
NAT: Network address translation
O
OF: OpenFlow
ONF: Open Network Foundation
ONOS: Open Networking Operating System
OS: Operating System
OSI: Open System Interconnection
OSPF: Open Shortest Path First
OVS: Open Virtual Switch
P
PAN: Personal Area Network
PE: Provider Edge
Q
QoS: Quality of Service
R
RIP: Routing Information Protocol
RSVP: Resources Reservation Protocol
S
SDN: Software Defined Networking
T
TCP: Transport Control Protocol
TDP: Tag Distribution Protocol
TE: Traffic Engineering
TLS: Transport Layer Security
TTL: Time To Live
U
UDP: User Datagram Protocol
V
VBS: Virtual Base Station
VM: Virtual Machin
VPN: Virtual Private network
VRF: Virtual Routing and Forwarding
W
WAN: Wide Area Network

INTRODUCTION GÉNÉRALE

Introduction Générale

Les réseaux et les télécommunications se dressent comme les fondations solides sur lesquelles repose notre société numérique. Ils jouent un rôle fondamental dans notre monde connecté d'aujourd'hui. Que ce soit pour échanger des informations, partager des ressources ou permettre la communication entre les individus et les appareils. Dans ce contexte, notre étude se concentre sur l'évaluation comparative de deux approches réseau clés : le MPLS et l'OpenFlow, une technologie émergente dans les réseaux définis par logiciel (SDN).

La problématique qui sous-tend notre étude réside dans l'évaluation des deux approches, le MPLS et l'OpenFlow, et dans la compréhension de leurs impacts respectifs sur la gestion des réseaux et la satisfaction des besoins des utilisateurs. En effet, avec l'évolution rapide des exigences en matière de communication et de connectivité, il est crucial de déterminer quelle technologie offre les meilleures performances, la flexibilité et la gestion optimisée des flux de données.

L'objectif principal de notre recherche est de comprendre et de comparer les performances, les fonctionnalités et les avantages de ces deux technologies, ainsi qu'en explorant les perspectives offertes par l'adoption future du SDN.

Dans un premier chapitre, nous présenterons en détail la technologie MPLS. Nous examinerons son architecture, son fonctionnement et ses applications courantes. Nous mettrons également l'accent sur les performances du MPLS en termes de contrôle, de scalabilité, de débit et d'autres métriques clés, tout en soulignant ses points forts et ses limites.

Le deuxième chapitre sera consacré à l'architecture OpenFlow et au paradigme SDN. Nous décrirons les principes radicaux du SDN, en mettant en évidence les avantages potentiels de cette approche consiste à séparer l'intelligence du réseau du dispositif de commutation par paquets et à la centraliser logiquement dans un contrôleur. Nous discuterons des caractéristiques d'OpenFlow.

Dans la partie pratique en chapitre trois de notre étude, nous présenterons l'environnement de test que nous avons mis en place, ainsi que les outils utilisés pour évaluer les performances des deux technologies. Nous effectuerons des tests approfondis en mesurant des paramètres tels que le débit, la latence (RTT), la gigue (Jitter) et examinons comment s'effectue l’acheminement des paquets. Nous analyserons ensuite les résultats obtenus pour établir une comparaison détaillée entre le MPLS et l'OpenFlow.

En conclusion générale, nous résumerons les principales conclusions de notre travail et nous discuterons des perspectives d'avenir pour les réseaux dans le contexte spécifique.

CHAPITRE I

Technologie MPLS

I.1 Introduction

MPLS (Multi Protocol Label Switching) est une technique de commutation de réseau en cours de normalisation qui combine les fonctionnalités de routage IP de niveau 3 avec les mécanismes de commutation de niveau 2 utilisés dans des technologies telles que l'ATM ou le Frame Relay.

En plus de sa capacité à fournir une commutation rapide, le MPLS offre également des services avancés tels que les réseaux privés virtuels (VPN) et l'ingénierie de trafic (TE), qui ne peuvent pas être mis en place sur des infrastructures IP traditionnelles.

Dans ce chapitre, il est indispensable de bien comprendre quelque notion de base des réseaux IP pour mieux comprendre leur fonctionnement. Nous allons ensuite aborder la technologie MPLS, son mécanisme de fonctionnement ainsi que ses différents composants.

I.2 Notions de base d'un réseau IP

I.2.1 Définition

Un réseau est un outil de communication qui permet à des individus ou à des groupes d'échanger des informations et des services. Ces réseaux sont classés en fonction de leur couverture et de leur domaine d'application. Pour communiquer entre eux, les nœuds utilisent des protocoles **[1]**.

I.2.2 Classification réseau

L'usage courant distingue les réseaux selon divers critères. La classification traditionnelle basée sur le concept d'étendue géographique respecte un ensemble de contraintes que les concepteurs doivent prendre en compte lors de la création de réseaux **[2]**.

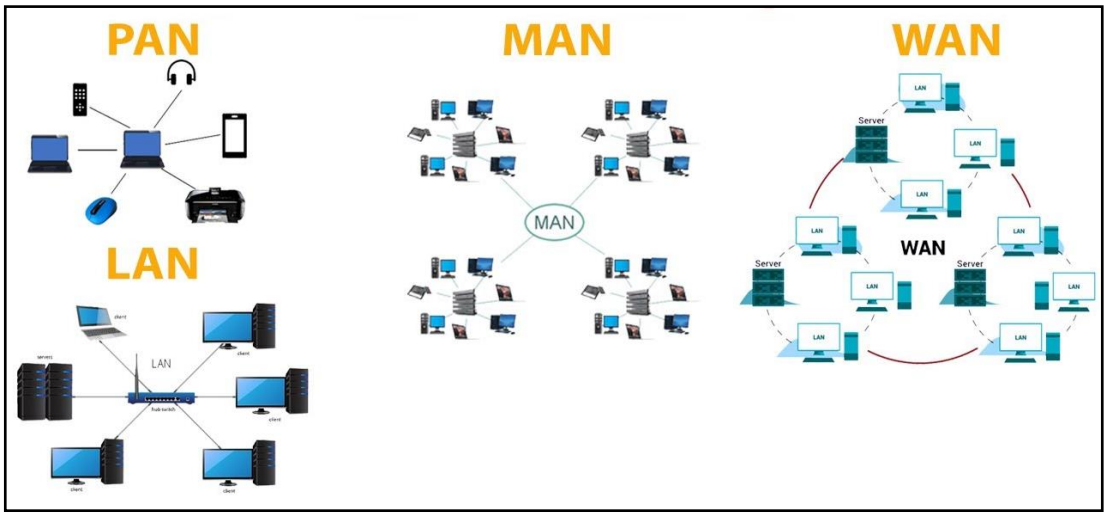


Figure.I.1 : Classification Réseaux

Les termes de la classification réseau sont présentés dans la Figure.I.1.

❖ **Réseaux personnels** (**P**ersonal **A**rea **N**etwork : **PAN**)

Un réseau personnel (domestique), est une connexion d'appareils informatiques dans un espace d'environ 10 mètres autour de l'utilisateur. Ce type de réseau est généralement utilisé pour connecter des périphériques tels que des imprimantes, des téléphones portables. Les connexions à ces appareils peuvent être filaires ou sans fil.

❖ **Réseaux locaux** (**L**ocal **A**rea **N**etwork : **LAN**)

Les réseaux locaux s'étendent sur des dizaines à des centaines de mètres connectent des ordinateurs appartenant à la même organisation. Ces réseaux peuvent être basés sur différentes technologies (filaires ou wifi), Ethernet étant la plus répandue. La petite taille de ce type de réseau présente les avantages de délais de transmission courts, d'absence d'erreurs et d'une administration simple.

❖ **Réseaux métropolitains** (**M**etropolitan **A**rea **N**etwork : **MAN**)

Un réseau MAN, aussi appelé réseau fédérateur, assure la communication a une longue distance et connecte souvent plusieurs réseaux LAN à grande vitesse. Il peut être utilisé pour connecter différents bâtiments distants de plusieurs dizaines de kilomètres, via des connexions privées. Un MAN se compose de commutateurs et de routeurs connectés par des liaisons à haut débit (généralement des fibres optiques).

❖ **Réseaux étendue** (**W**ide **A**rea **N**etwork: **WAN**)

Un WAN est constitué de connexions issues de LANs ou de MANs qui peuvent transmettre des informations à des milliers de kilomètres à travers le monde via des routeurs et des liaisons nationales ou internationales à très haut débit appelés backbones. Les routeurs jouent un rôle important (contrôler le trafic) car la majeure partie du trafic WAN se situe dans les LAN qui le constituent.

I.2.3 Modèle OSI (Open System Interconnection)

Il s'agit d'un modèle de référence pour identifier les niveaux opérationnels des composants du réseau, défini par l'organisation ISO en 1984. Il se compose de sept couches (Figure.I.2). Pour qu'une couche envoie des commandes ou des données elle doit représenter les informations et les transmettre à toutes les couches inférieures. Chaque couche ajoute des en-têtes spécifiques (encapsulation). A l'arrivée, ces informations sont désencapsulés pour libérer la commande ou les données **[1]**.

- **Rôle de différentes couches**

Chaque couche réseau définie par le modèle a un rôle spécifique :

❖ **Couche physique**

Son rôle est la transmission bit à bit d'un signal électrique, électromagnétique ou lumineux qui encode des données numériques sur un support entre un émetteur et un récepteur.

❖ **Couche liaisons de données**

Cette couche convertit les données numériques en un signal et organise les bits de données en trames. Un en-tête est créé qui vous permet d'identifier les expéditeurs et les destinataires en fonction de leurs adresses physiques.

❖ **Couche réseau**

Ce niveau régit la sélection de la meilleure façon d'atteindre le destinataire en comparant les différents coûts calculés. Une adresse logique permet de faire référence à un composant de manière globale. Relativement à des protocoles, les blocs peuvent être appelés paquets, comme dans Internet Protocole (IP).

❖ **Couche transport**

C'est le cœur du modèle OSI. A ce niveau, divers mécanismes sont mis en œuvre pour établir le mode connecté afin qu'il assure la transmission des informations de manière fluide. Les protocoles les plus connu à ce niveau sont TCP et UDP.

❖ **Couche session**

C'est une couche qui gère au même le mode connecté et les connexions aux ressources partagées sur un réseau. Ce niveau manage les points de synchronisation et permet la reprise en cas d'incident en enregistrant les contextes et les sous-contextes.

❖ **Couche présentation**

Le rôle de cette couche est de mettre en place les données. Elle peut effectuer des opérations de compression, de chiffrement et de conversion de données.

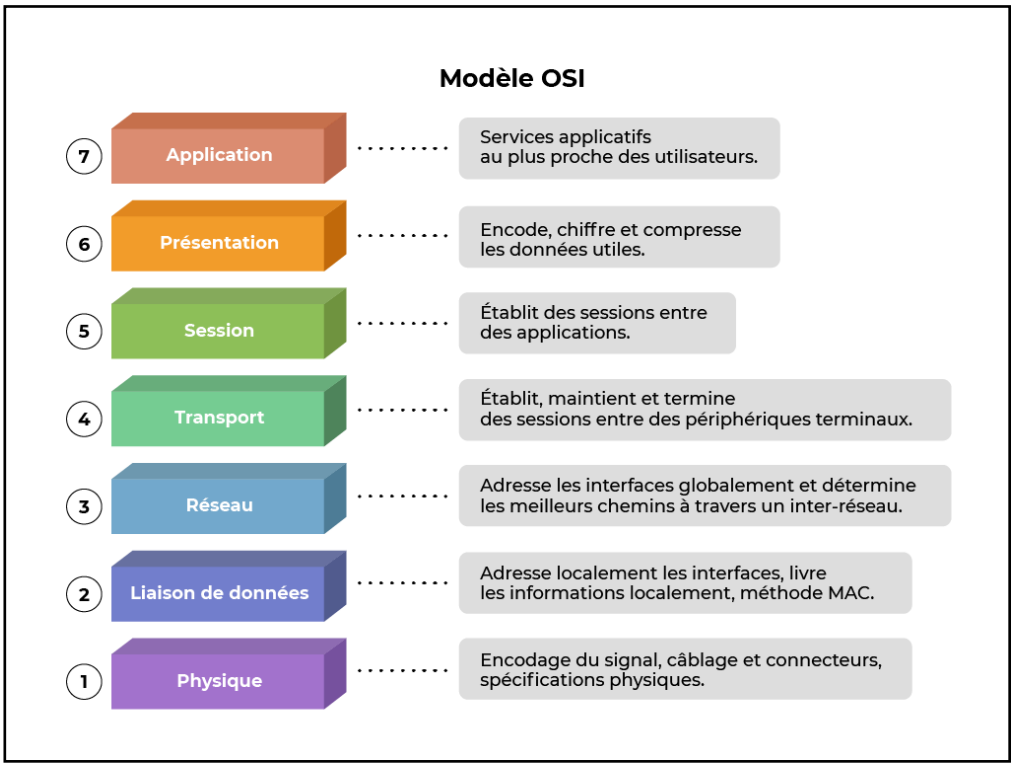


Figure.I.2 : Modèle OSI

❖ **Couche application**

Cette couche fournit une interface de communication avec l'utilisateur par l'intermédiaire d'un logiciel approprié. Elle gère aussi la communication entre les applications.

I.2.4 Modes de routage

Un routeur peut apprendre des réseaux distants de deux manières : **[3]**

I.2.4.1 Routage statique

Permet à l'administrateur de saisir manuellement les routes sur les routeurs et ainsi de choisir le meilleur chemin pour aller d'un réseau A à un réseau B.

I.2.4.2 Routage dynamique

C'est un mécanisme par lequel les routeurs communiquent entre eux et peuvent alors se configurer les uns des autres de manière totalement automatique. Ce mode est basé sur plusieurs protocoles qu'on va citer les plus fréquents et important dans le routage dynamique (Figure.I.3).

I.2.5 Protocol de routage

Un Protocol de routage est un ensemble de règles utilisées pour créer et mettre à jour dynamique des tables de routage, afin d'envoyer des données dans la bonne direction.
Il existe deux types de protocoles de routage :

I.2.5.1 IGP (Interior Gateway Protocols)

Protocole de routage intra-domaines (intérieur), il est appliqué à l'intérieur d'un domaine. Ils sont basés sur des algorithmes de vecteur de distance qui n'ont qu'une vue partielle du réseau, ou des algorithmes d'état de liens qui créent une vue de l'ensemble du domaine géré par chaque routeur.

- **RIP (Routing Information Protocol)**

Il s'agit d'un protocole de routage de taille moyenne à vecteur de distance, il utilise le nombre de sauts comme métrique pour la sélection de chemin (limité en 15 sauts au maximum), par défaut les mis à jour de table de routage sont envoyées toutes les 30 secondes.

- **EIGRP (Enhanced Interior Gateway Routing Protocol)**

C'est un protocole de routage à vecteur de distance améliorée et propriétaire développé par Cisco, à l'aide de l'algorithme DUAL (Diffused Update Algorithm). Il calcule le chemin le plus court, ainsi les mis à jour de table de routage sont diffusées par des modifications topologique.

- **OSPF (Open Shortest Path First)**

OSPF est un protocole à état de lien qui conserve une vue complète de la topologie du système dans sa base de données et à partir de cette dernière, il construit une table de routage en calculant l'arborescence des chemins du système en comparant le cout de chacun. De plus, les mises à jour du routage sont diffusées à mesure des modifications de topologie.

- **Principe de base d'OSPF :**

La diffusion de l'état des liens de chaque routeur à tous les autres routeurs peut générer beaucoup de trafic en arrière-plan sur le réseau. Un routeur, appelé *DR (Designated Router)* est alors désigné pour centraliser et redistribuer tous les états de liaison (LSA) sur tous les

routeurs. Pour réduire le trafic de contrôle, en cas de panne un routeur b-DR (backup-DR) est capable d'assurer les fonctions du DR, le domaine OSPF est divisé en zones (zones OSPF) et chaque zone fonctionne indépendamment selon l'algorithme de Dijkstra de plus court chemin. Une zone spécifique appelée la *zone backbone* (zone fédératrice) sert de lier entre toutes les zones.

- **IS-IS (Intermediate System -to- Intermediate System)**

IS-IS est un protocole normalisé d'état de lien qui a été développé pour être le protocole de routage définitif pour le modèle OSI. Il présente de nombreuses similitudes avec OSPF, sauf qu'il fonction sur la couche liaisons de données et cela permettra d'éviter des inconvénients basés sur protocole IP. IS-IS est également utilisé par les FAI en raison de son évolutivité. **[4]**

I.2.5.2 EGP (Exterior Gateway Protocol)

Protocole de routage inter-domaines (extérieur), celui-ci est utilisé pour la communication avec d'autres domaines, ce dernier fonctionne principalement sur les routeurs de bordure du réseau (Figure I.3).

- **BGP (Border Gateway Protocol)**

Il s'agit d'un protocole de routage extérieur à vecteur de distance. Son objectif principal est d'échanger des informations de routage et d'accessibilité du réseau entre les systèmes autonomes dans le réseau internet e-BGP (exterior *BGP*). Un réseau qui utilise BGP au sein d'un AS pour échanger et traverser des routes entre les routeurs périphériques du même AS est appelé i-BGP (Interior BGP) **[3]**.

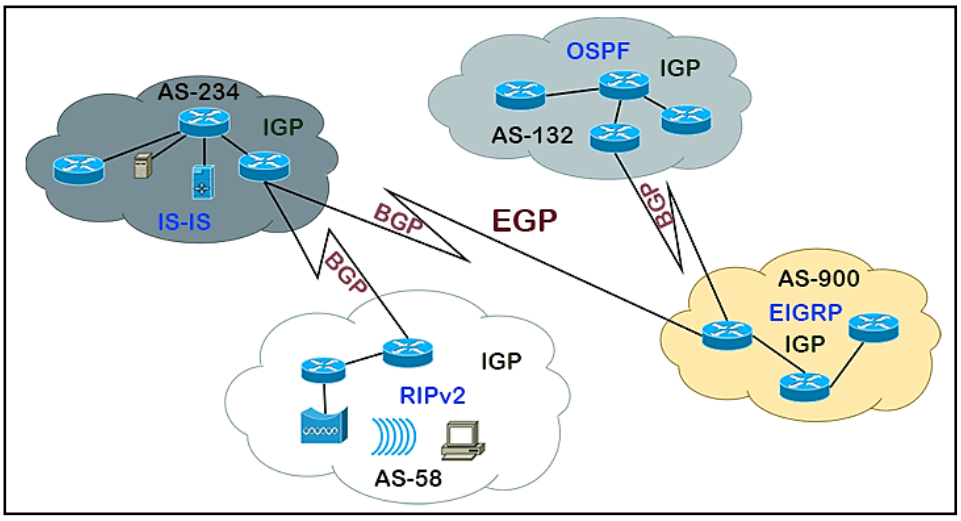


Figure.I.3 : Protocol de routage IGP /EGP

Dans le cadre de notre projet de fin d'étude nous proposons de simuler sous VMware Workstation Pro les deux architectures : MPLS et SDN afin d'effectuer une étude comparative entre les deux technologies. La partie suivante sera consacrée au détail du protocole MPLS.

I.3 Passage vers MPLS

Le processus de routage prend beaucoup de temps et consomme trop de ressources du routeur. Donc, il est nécessaire de trouver un moyen plus efficace d'acheminer les paquets. L'idée est venue de l'analyse de deux protocoles (Figure.I.4) ATM et IP, qui appartiennent aux couches 2 et 3 du modèle OSI.

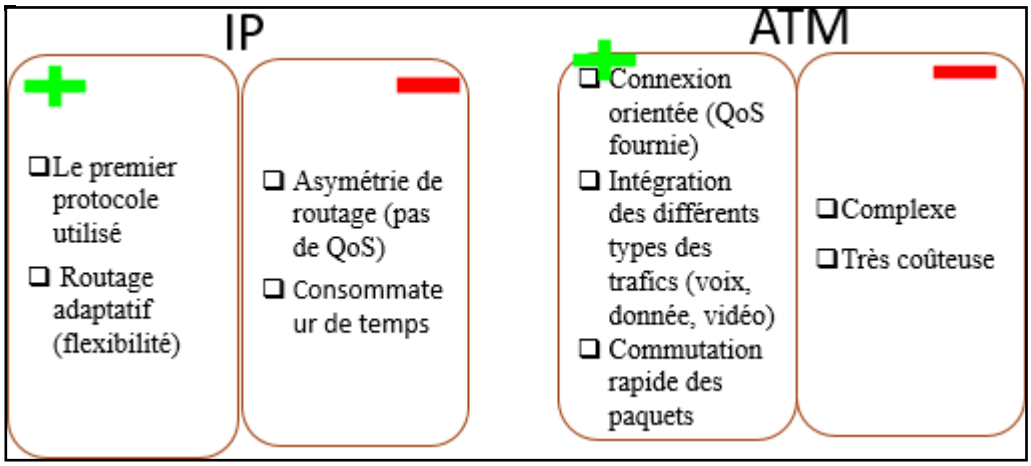


Figure.I.4 Analyses des deux protocoles IP et ATM

Ce qui a été développé est une nouvelle technologie appelée MPLS. En raison de profiter des avantages du routage et de la commutation IP pour répondre aux besoins de fiabilité et de disponibilité **[5]**.

I.3.1 Définition MPLS et sa place dans le modèle OSI

MPLS est une technologie d'acheminement asynchrone par paquets. En ce sens, elle est similaire à IP, mais avec un plan de transfert beaucoup plus léger et réduit considérablement la quantité d'états qui doivent être signalés et programmés au niveau des équipements, elle appartient à la couche 2.5 du modèle OSI, illustré dans la Figure.I.5 **[5]**.

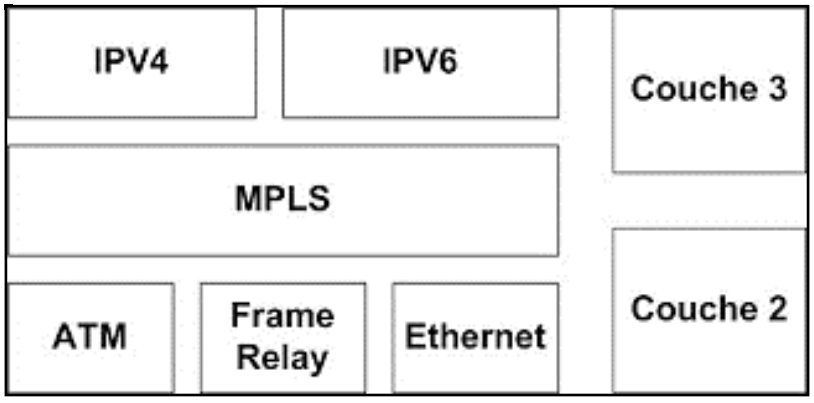


Figure.I.5 : Placement du MPLS dans le modèle OSI

- **Multi Protocoles :** Il est capable de supporter les différents protocoles de niveau inférieur d'OSI (ATM, Frame Relay, Ethernet, IPv4, IPv6 ...).
- **Label Switching (commutation par étiquette) :** Il commute les paquets de données en fonction des étiquettes.

I.3.2 Terminologies du MPLS

L'architecture MPLS se compose de : **[5]**

I.3.2.1 LSR (Label Switch Router)

Un routeur à commutation de labels (LSR) est un routeur qui prend en charge MPLS. Il est capable de comprendre les étiquettes MPLS et de recevoir et transmettre un paquet étiqueté sur une liaison de données. Trois types de LSR existent dans un réseau MPLS :

- **Ingress LSRs :** Les LSR d'entrée reçoivent un paquet qui n'est pas encore étiqueté, insèrent une étiquette devant le paquet et l'envoient sur une liaison de données.
- **Egress LSRs** : Les LSR de sortie reçoivent des paquets étiquetés, retirent la et les envoient sur une liaison de données. Les LSR d'entrée et de sortie sont des LSR de bordure.
- **Intermediate LSRs :** Les LSR intermédiaires reçoivent un paquet étiqueté entrant, effectuent une opération sur celui-ci, commutent le paquet et l'envoient sur la bonne liaison de données.

N.B : L'Ingress LSR et l'Egress LSR sont également désignés par le terme "LER" (Label Edge Router).

I.3.2.2 LSP (Label Switchd Path)

Un chemin commuté par étiquette (LSP) est une séquence de LSR qui commute un paquet étiqueté à travers un réseau MPLS ou une partie d'un réseau MPLS.

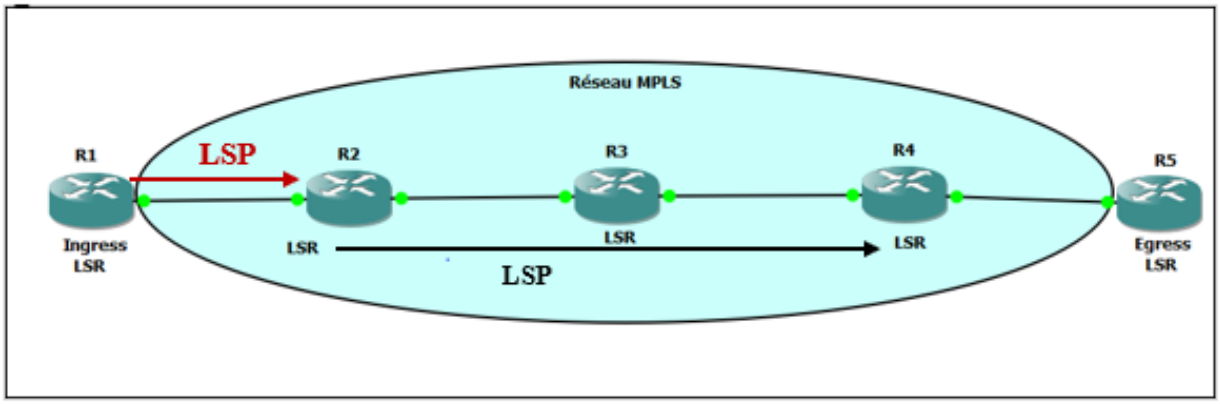


Figure.I.6 : Chemin LSP

En fait, le LSP est le chemin que prennent les paquets à travers le réseau MPLS ou une partie de celui-ci. Un LSP est unidirectionnel.

I.3.2.3 FEC (Forwarding Equivalence Class)

La Figure.I.7 illustre un exemple d'une classe d'équivalence :

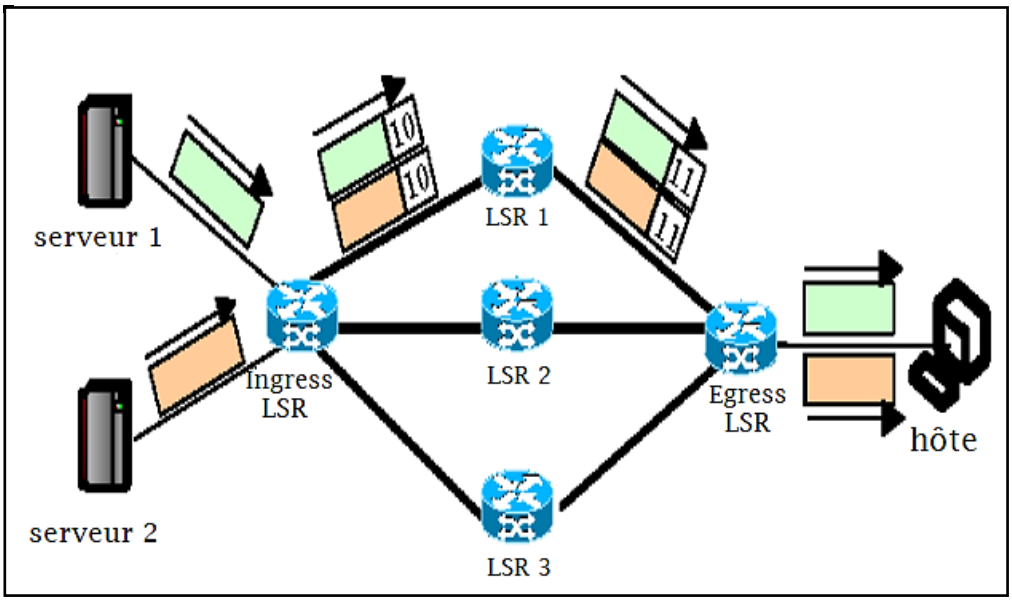


Figure.I.7 : Exemple d'une FEC

Une FEC est un groupe ou un flux de paquets qui reçoivent le même traitement d'acheminement dans tout le réseau MPLS qui ont la même priorité.

I.3.3 Format du label MPLS

MPLS occupe 4 octets (32 bits) représenté dans la Figure.I.8 : **[5]**

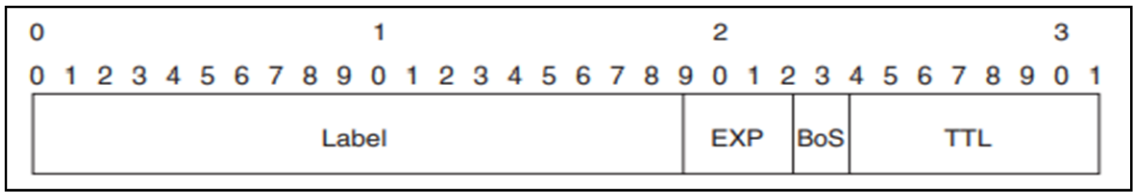


Figure.I.8 : Format de l'en-tête MPLS

❖ L'en-tête

- ✓ **Label** : Ce sont les 20 premiers bits correspondent à la valeur de l'étiquette.
- ✓ **EXP (Expérimental)** : Sont les trois bits expérimentaux (EXP). Ces bits sont utilisés uniquement pour la qualité de service (QoS).
- ✓ **BoS (Bit of Stack)** : Ce champ sert pour empiler les en-têtes MPLS. Lorsque ce bit est à 1, le bas de la pile est atteint, et l'en-tête de niveau 3 est placé juste après.
- ✓ **TTL (Time To Live)** : Ce TTL a la même fonction que le TTL présent dans l'en-tête IP. Il est simplement diminué de 1 à chaque saut, et sa principale fonction est d'éviter qu'un paquet soit bloqué dans une boucle de routage. Si le TTL de l'étiquette atteint 0, le paquet est rejeté.

❖ Opérations sur les labels

Les opérations sur les labels dans un LSR sont schématisées dans la Figure.I.9 :

- ✓ **Swap :** Échanger un label par le label suivant (next-hop label) dans un LSR du cœur du réseau.

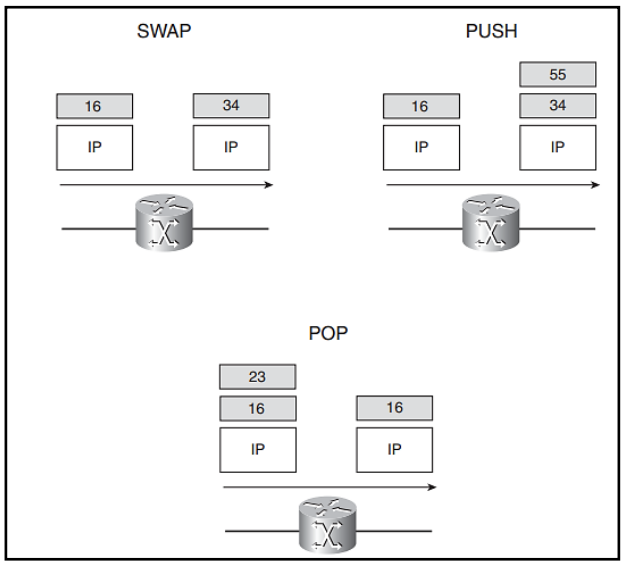


Figure.I.9 : Operations sur les labels

- ✓ **Push :** Insérer un label dans un LSR d'entrée.
- ✓ **Pop :** Supprimer un label dans un LSR de sortie.

I.3.4 Structure fonctionnelle d'un LSR

Pour la prise en charge de plusieurs protocoles et structure fonctionnelle. La technologie MPLS repose sur deux plans principaux, à savoir : le plan de contrôle et le plan de données **[5]**.

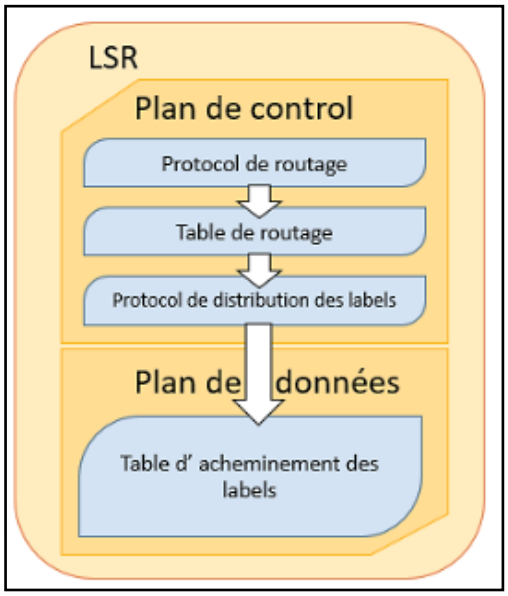


Figure.I.10 : Structure d'un LSR

I.3.4.1 Plan de contrôle (Control plane)

Il sera responsable de la gestion, de la maintenance et de la distribution des routes et étiquettes contenues dans chaque routeur du réseau MPLS. Ce plan de contrôle utilise des protocoles de routage classiques pour contrôler les informations de routage, comme OSPF, RIP ou IS-IS pour créer la topologie des nœuds du réseau MPLS, et des protocoles développés spécifiquement pour MPLS, comme le protocole de distribution d'étiquettes (LDP) que nous passerons en revue par la suite.

I.3.4.2 Plan de données (Data plane)

Il permet le contrôle d'envoi des informations en fonction de la commutation d'étiquette ou de l'adresse de destination. Il est totalement indépendant de la partie signalisation (Figure.I.10).

I.3.5 Tables du routeur LSR

- **RIB (Routing Information Base) :** Contient les informations nécessaires pour atteindre le réseau IP de destination, la table RIB réside dans le plan de contrôle du routeur LSR.
- **FIB (Forwarding Information Base) :** Elle est alimentée à partir des informations stockées dans la table RIB pour permet d'accélérer le transfert de données, c'est au niveau du plan de données du routeur LSR.

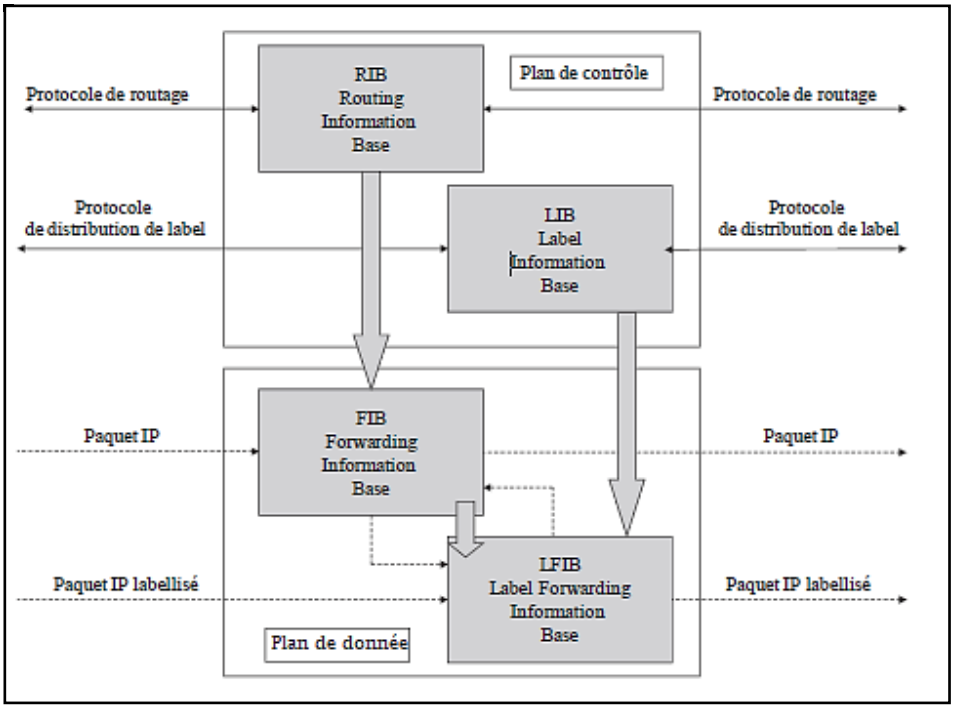


Figure.I.11 : Tables du routeur LSR

- **LIB (Label Information Base) :** La table LIB (Label Information Base) est alimentée par le protocole de distribution d'étiquettes LDP (Label Distribution Protocol). Contient les liens entre le label et la FEC. Autrement dit, content tous les chemins possibles pour

atteindre la destination. La table LIB réside dans le plan de contrôle du routeur LSR (Figure.I.11).

- **LFIB (Label Forwarding Information Base) :** Elle est alimentée à partir des tables FIB et LIB. Utilisé par les routeurs LSR Core contenant les labels du meilleur prochain saut pour effectuer la commutation d'étiquettes **[6]**.

I.3.6 Distribution de labels

L'utilisation d'un protocole distinct pour la distribution des étiquettes présente l'avantage d'être indépendante du protocole de routage **[7]**.

Plusieurs variétés de protocoles distribuent les étiquettes :

- TDP (Tag Distribution Protocol) : C'est le premier protocole de distribution de labels développé par Cisco ;
- LDP (Labels Distribution Protocol) : Le protocole le plus utilisé pour la distribution des labels ;
- RSVP (Ressource Reservation Protocol) : Est utilisé sauf pour MPLS TE (Traffic Engineering).

Parmi les protocoles décrits précédemment, nous nous focalisons beaucoup plus sur le protocole LDP :

Le protocole LDP établit des méthodes de communication entre les "LSR" pour leur permettre de partager les tables de commutation des labels et des flux de leurs voisins. Les labels utilisés sont de type "saut par saut", c'est-à-dire qu'ils ne sont mis à jour que pour le prochain saut. La découverte des "LSR" voisins se fait via le protocole UDP, qui envoie des ACK (Acknowledgment) pour l'accusé de réception. Une fois que deux nœuds sont découverts, ils établissent une connexion TCP pour assurer le transport fiable des données.

Les messages échangés lors d'une session LDP (Figure.I.12) entre deux routeurs LSR sont de plusieurs types, à savoir :

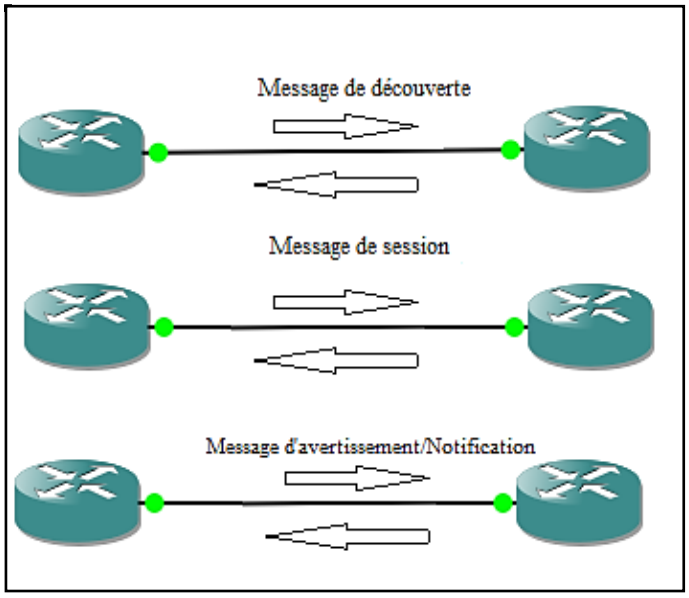


Figure.I.12 : Etablissement d'une connexion LDP.

- *Messages de découverte* : Ils permettent la recherche et le maintien de la connexion avec un LSR sur le réseau.
- *Messages de session* : Ils servent à établir, maintenir et mettre fin aux sessions LDP.
- *Messages d'avertissement* : Ils sont utilisés pour créer, modifier et supprimer les correspondances entre FEC et labels.
- *Messages de notification* : Ils sont destinés à signaler les erreurs.

I.4 Fonctionnement MPLS

Le MPLS vise à simplifier le routage des paquets IP en remplaçant les opérations complexes par des opérations plus simples (Figure.I.13). Lorsqu'un paquet pénètre dans le réseau MPLS, un label lui est attribué par les routeurs d'extrémité (LER), ce qui permet aux autres routeurs d'utiliser ce label correspondant sans avoir à effectuer de recherche dans leur table de routage.

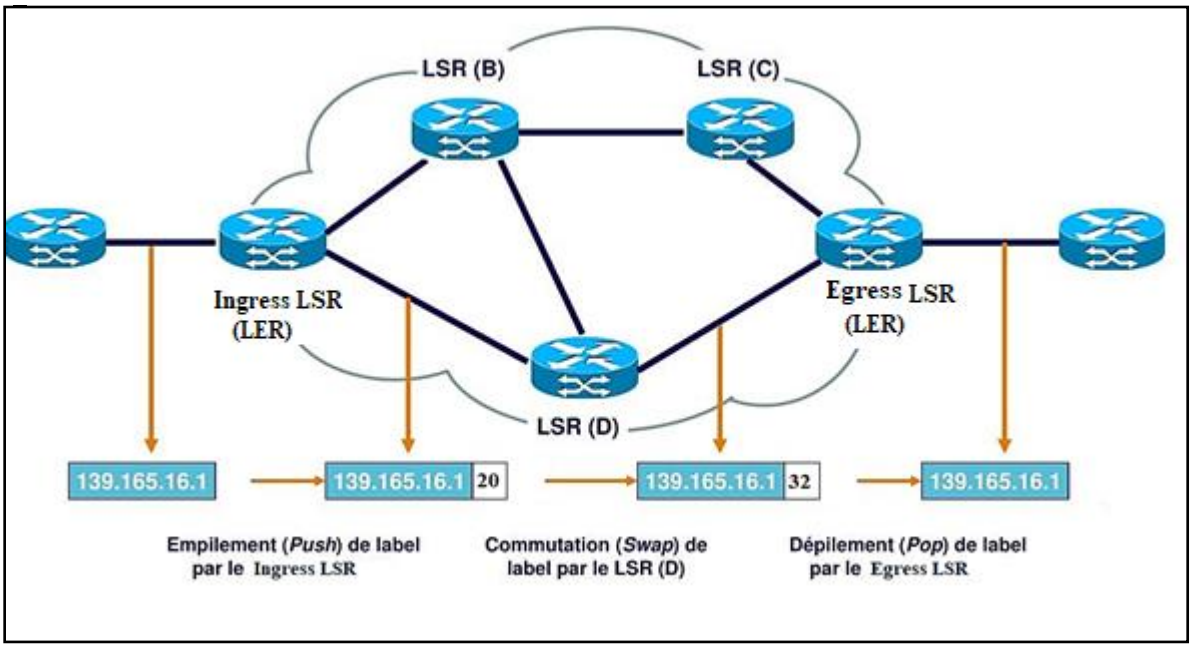


Figure.I.13 : Fonctionnement du MPLS

Cela accélère le transfert des paquets. Les labels sont ajoutés une seule fois à l'entrée du paquet par l'Ingress LSR et sont permutés par les routeurs LSR. Le label d'entrée est remplacé par un label de sortie avant que le paquet ne soit envoyé au nœud suivant. Lorsque le paquet atteint l'Egress LSR, le label est supprimé et le paquet est remis à son destinataire, faisant ainsi apparaitre de nouveau le paquet IP pour le réseau qui suit **[5]**.

I.5 Application MPLS

Les applications clés de MPLS nécessitent l'utilisation des composants spécifiques pour les fonctionnalités souhaitées. Ainsi, l'implémentation de MPLS variera selon les objectifs visés, notamment en termes de classification et de distribution des labels qui sont les mécanismes fondamentaux d'acheminement des paquets communs à toutes les approches. Les principales applications de MPLS sont les suivantes : **[5]**

I.5.1 Ingénierie de trafic (TE)

Le trafic peut être efficacement réparti dans le réseau afin d'éviter les liaisons sous-utilisées et surutilisées, tout en prenant en compte la largeur de bande et les attributs des liaisons tels que le délai et la gigue. Cette technologie s'adapte automatiquement aux changements de bande passante et d'attributs de liaison. Pour que MPLS TE fonctionne correctement, plusieurs éléments sont nécessaires, tels que les contraintes de liaison qui déterminent la quantité de trafic qu'une liaison peut supporter et quel tunnel TE peut utiliser la liaison. Les informations TE sont distribuées par le protocole de routage à l'état de lien MPLS TE et un algorithme est utilisé pour calculer le meilleur chemin entre le LSR de tête et le LSR de fin. De plus, un protocole de signalisation appelé RSVP est utilisé pour signaler le tunnel TE à travers le réseau et un moyen de transmettre le trafic sur le tunnel.

I.5.2 MPLS VPN

Un réseau privé virtuel (VPN) est un réseau privé qui est construit sur une infrastructure publique. Des mécanismes de sécurité, tels que le cryptage, permettent aux utilisateurs de VPN d'accéder en toute sécurité à un réseau depuis différents endroits via un réseau Backbone d'un opérateur de télécommunication.

Les fournisseurs de services peuvent déployer deux grands modèles de VPN pour fournir des services VPN à leurs clients :

- **MPLS L2VPN** : Qui fournit une connexion de couche 2 entre les CEs et il permet aux clients d'exécuter plus de protocoles sur une liaison de couche 2, cela signifie que les informations de routage ne sont pas partagées avec le fournisseur de services. Dans cette solution aucune interaction de routage ne se produit entre le client et le fournisseur de services (CE commute le trafic vers le FAI PE au format de couche 2, le client a un contrôle total sur ses politiques et son routage) (Figure.I.14).

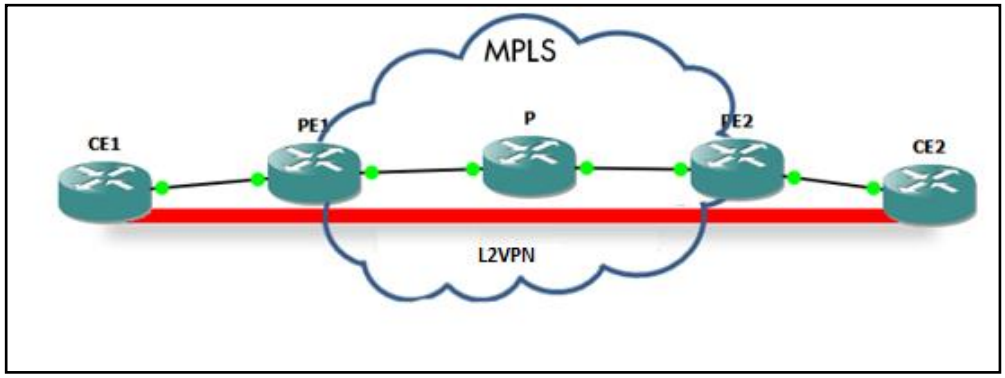


Figure.I.14 : Principe L2VPN

- **MPLS L3VPN** : Qui fournit une connexion de couche 3 pour les sites distants des clients CEs. Les routeurs du nuage MPLS de l'opérateur acheminent et transfèrent le trafic VPN aux points d'entrée et sortie des routeur PEs, dont ces derniers annoncent et importent le routage des clients dans des tables VRFs appropriées

Les VRF garantissent que les informations de routage des différents clients restent séparées, et le MPLS dans le backbone garantit que les paquets sont transmis sur la base des informations de l'étiquette et non des informations de l'en-tête IP. L'annonce des VRFs se fait à l'aide du protocole MP-BGP dans les PEs d'extrémité du backbone MPLS. La Figure.I.15 illustre le concept L3VPN :

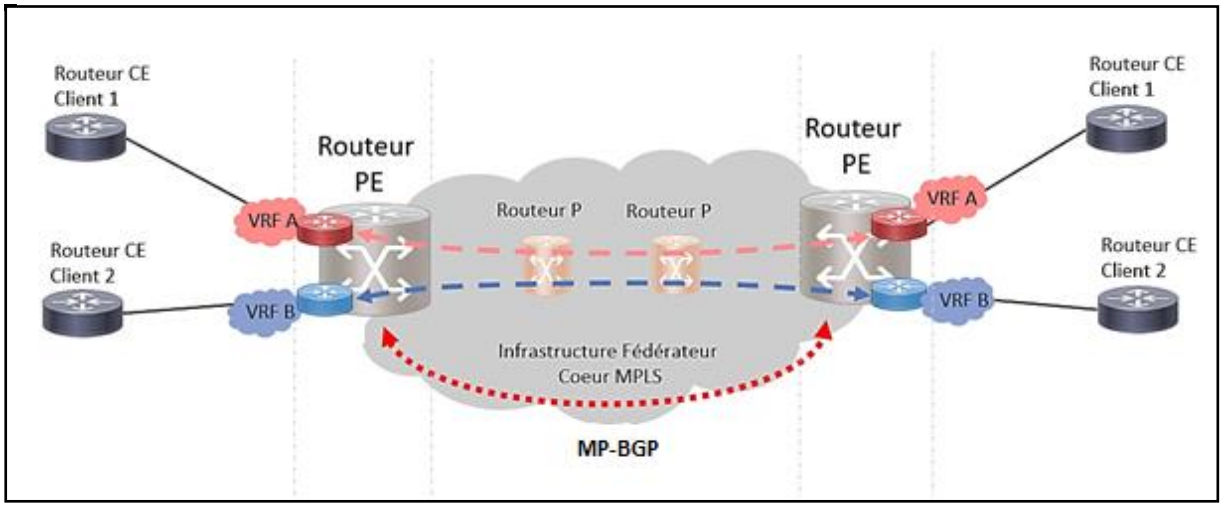


Figure.I.15 : Principe L3VPN

I.5.3 AToM

La fonction par laquelle n'importe quelle trame de couche 2 est transportée à travers le backbone MPLS est appelée *Any Transport over MPLS* (AToM). Elle permet au fournisseur de services de fournir aux clients le même service de niveau 2 qu'avec n'importe quel réseau spécifique non-MPLS. En même temps, le fournisseur de services n'a besoin que d'une infrastructure de réseau unifiée pour transporter tous les types de trafic des clients.

I.5.4 QoS

La qualité de service est un facteur qui permet d'optimiser les performances d'un réseau et assurer les performances désirées pour un trafic donnée dans le réseau, ces principaux indicateurs sont : le débit, latence, gigue et perte de paquet. Dans un réseau MPLS en utilise deux modèles complémentaires, IntServ applique un contrôle de bout en bout sur les ressources utilisées (réservations de ressources RSVP), tandis que DiffServ spécifie le comportement par saut.

I.6 Conclusion

En conclusion, ce chapitre a porté sur les réseaux IP, les protocoles de routage et les principes de base de MPLS, notamment l'étiquetage des paquets et l'établissement des chemins de commutation. Enfin, nous avons examiné les différentes applications de MPLS. Dans le deuxième chapitre nous allons étudier et analyser la nouvelle technologie SDN exploitée dans notre projet pour une meilleure gestion et un bon contrôle des services offerts par un fournisseur réseau.

CHAPITRE II

Architecture SDN : OpenFlow

II.1 Introduction

Le SDN (Software Defined Networking) est basé sur le protocole OpenFlow. C'est un protocole de communication permettant de contrôler les flux de données dans un réseau. Dans ce chapitre, nous explorerons les concepts fondamentaux du SDN et d'OpenFlow, et nous aborderons les avantages qu'ils offrent aux entreprises en termes de flexibilité, d'efficacité et de sécurité du réseau. Nous examinerons également les différentes architectures et implémentations de SDN, ainsi que les défis et les opportunités associés à l'utilisation de cette technologie.

II.2 Concept de base de la virtualisation

La virtualisation est un mécanisme informatique qui permet de faire fonctionner plusieurs systèmes, serveurs ou applications sur une même machine physique (Figure.II.1), comme s'ils fonctionnaient sur des machines physiques distinctes. Pour que la virtualisation soit efficace, elle doit respecter certains principes fondamentaux, notamment le cloisonnement, qui assure que chaque système d'exploitation fonctionne de manière indépendante et ne peut interférer avec les autres.

Un autre principe est la transparence, qui implique que le fonctionnement en mode virtualisé ne modifie pas le fonctionnement du système d'exploitation et des applications. La transparence garantit la compatibilité : toutes les applications peuvent fonctionner sur un système virtualisé sans être modifiées **[8]**.

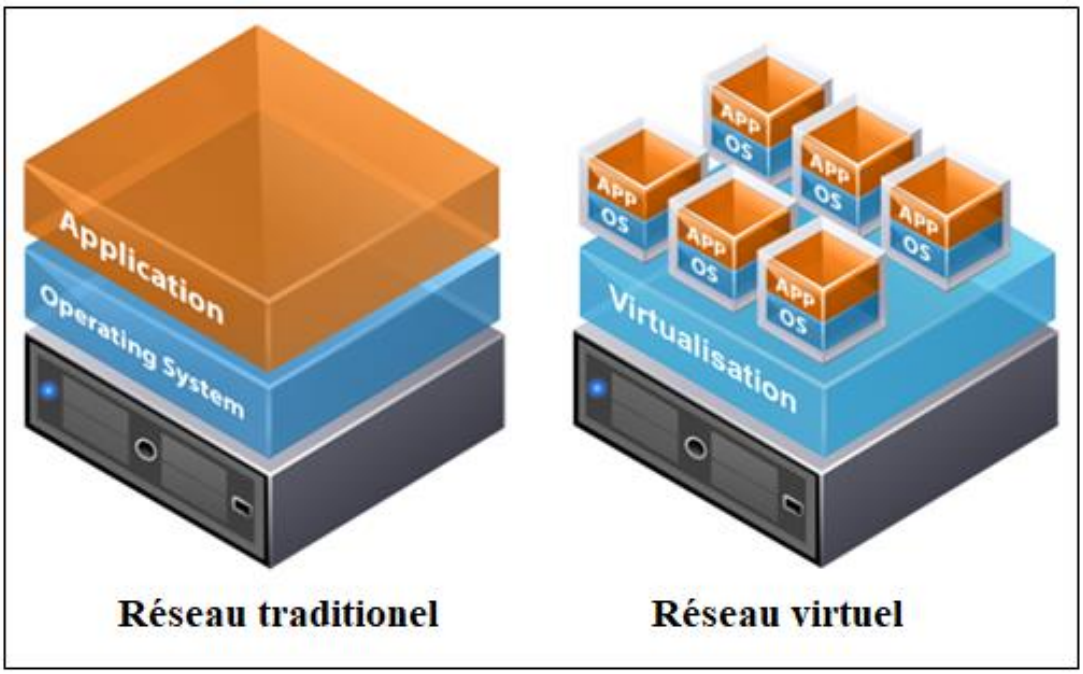


Figure. II.1 : Concept de la virtualisation

La virtualisation permet d'installer plusieurs systèmes d'exploitation sur un seul serveur physique en créant des machines virtuelles, chacune disposant de son propre environnement indépendant. Un logiciel appelé *hyperviseur* est installé sur le système hôte et permet de créer et de gérer ces machines virtuelles. Chaque machine virtuelle dispose d'un accès aux ressources du serveur physique et peut être configurée pour exécuter un système

d'exploitation différent. La virtualisation consiste à reproduire complètement l'environnement matériel dans une machine virtuelle pour accueillir un système d'exploitation complet.

L'hyperviseur joue un rôle important dans la virtualisation en assurant plusieurs fonctionnalités telles que la gestion des ressources de la machine hôte, l'allocation des ressources nécessaires à chaque machine virtuelle et la prévention des interférences entre les machines virtuelles. En bref, il garantit un environnement isolé et stable pour chaque machine virtuelle.

II.2.1 Types d'hyperviseurs

Il existe deux types d'hyperviseurs, pour le type 1, aussi appelé *hyperviseur natif*, s'exécute directement sur le matériel de l'hôte pour gérer les systèmes d'exploitation invités. Il planifie directement les ressources des machines virtuelles sur le matériel et est souvent utilisé dans les data centers d'entreprise. Des exemples d'hyperviseurs de *type 1* sont KVM, Microsoft Hyper-V et VMware vSphere. En revanche, le *type 2* aussi appelé *hyperviseur hébergé*, s'exécute sur un système d'exploitation traditionnel en tant que couche logicielle ou application. Il fonctionne en dissociant les systèmes d'exploitation invités du système d'exploitation hôte et planifie les ressources des machines virtuelles au niveau de l'hôte. Ce type d'hyperviseur convient aux utilisateurs qui souhaitent exécuter plusieurs systèmes d'exploitation sur un ordinateur personnel. Des exemples d'hyperviseurs de type 2 sont VMware Workstation et Oracle VirtualBox.

II.2.2 Avantage de la virtualisation

La virtualisation présente plusieurs avantages tels que l'agilité, qui permet de modifier la topologie du réseau virtuel sans intervention sur l'infrastructure physique existante. En outre, la consommation d'énergie est réduite car elle limite le nombre de matériel physique, ce qui divise le coût en électricité et en climatisation. La fiabilité est également un avantage, car en cas d'erreur ou de perte de configuration du réseau, la sauvegarde et la restauration du réseau virtuel sont possibles avec une reprise rapide en quelques instructions. Enfin, la capacité multi-client permet à plusieurs réseaux logiques de cohabiter indépendamment sur un même réseau physique, ce qui évite que les réseaux ne se voient les uns les autres et permet à un équipement physique d'être utilisé par les deux réseaux virtuels.

II.2.3 Types de virtualisation d'un réseau opérateur

Pour garantir une expérience utilisateur de qualité à un coût abordable, les futurs réseaux virtualisés doivent adopter une architecture plus flexible que celle des réseaux traditionnels. Ainsi, il existe trois méthodes de virtualisation des réseaux d'opérateurs :

- La première est la virtualisation locale, qui consiste à diviser chaque station de base en plusieurs ressources pour créer des stations de base virtuelles (VBS). L'hyperviseur gère l'allocation des ressources physiques et la synchronisation entre les différentes instances virtuelles installées sur la même VBS.
- La deuxième est la virtualisation centralisée, qui consiste à centraliser tous les traitements du réseau dans une seule entité appelée *orchestrateur réseau*. Cette dernière doit maintenir

à jour les informations sur la topologie pour déployer les réseaux virtuels de manière optimale.

- La troisième approche est la virtualisation hybride, qui combine de manière optimale les approches locale et centrale.

Comme le *SDN* est l'un des objectifs de ce travail nous présentons cette technologie dans la suite de ce chapitre.

II.3 Passage au SDN

Les réseaux traditionnels actuels présentent des inconvénients en raison de leur nature statique, ce qui exige une intervention manuelle pour toute modification de la configuration et peut prendre beaucoup de temps et d'énergie. Cela a créé une pression considérable sur les entreprises. Le SDN est apparu comme une solution pour la création et la gestion de réseaux dynamiques, face à ces problèmes des réseaux traditionnels.

En 2008, les équipes de recherche des universités de Berkeley, Stanford et du MIT ont travaillé ensemble pour créer un nouveau concept d'architecture réseau. En 2011, l'ONF (Open Networking Foundation), une organisation à but non lucratif visant à promouvoir le SDN, a été créée par des géants des télécommunications et des technologies, tels que Deutsche Telekom, Facebook, Google, Microsoft, Verizon et Yahoo, ainsi que les principaux fabricants IT, notamment Cisco, Juniper, HP, Dell, Broadcom et IBM. Le SDN repose sur le principe des réseaux virtuels et permet la coexistence de plusieurs services sur une infrastructure physique unique de manière dynamique.

Le SDN est bien plus qu'un simple travail de recherche universitaire, c'est une nouvelle organisation des réseaux qui intéresse vivement tous les grands acteurs du secteur **[9]**.

II.3.1 Historique et Naissance du SDN

En 2007, Martin Casado et deux autres personnes ont fondé l'entreprise NICIRA, tandis que Martin travaillait sur sa thèse de doctorat portant sur le développement d'OpenFlow. Cela lui a permis d'exécuter le premier OVS (OpenVirtual Switch qui prend en charge OpenFlow) au sein de son entreprise. Cinq ans plus tard, VMware a racheté NICIRA car elle avait compris que la notion d'OpenFlow allait changer le paysage des réseaux dans les années à venir. Martin a été nommé directeur de l'entreprise après son acquisition.

Le rôle d'OpenFlow est de séparer le plan de contrôle et de le placer dans un équipement appelé *contrôleur*, tandis que le plan de données devient simplement l'exécution des ordres du contrôleur, permettant ainsi une gestion centralisée et plus facile. C'est ainsi qu'est né le concept de SDN grâce à OpenFlow, qui fait office d'intermédiaire entre les deux plans **[10]**.

II.3.2 Définition du SDN

Le SDN est une technologie permettant de définir une architecture réseau à travers des applications. Cette technologie a pour 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 changeante des applications. Elle permet de séparer les fonctions de contrôle du réseau et de transfert de données en virtualisant la couche réseau et en créant une couche logicielle d'hyperviseur qui se place au-dessus des outils matériels pour offrir une vue unique des équipements et rendre le réseau directement programmable.

Aujourd'hui, le SDN est reconnu comme une architecture qui ouvre le réseau aux applications. Elle permet aux applications de programmer le réseau pour accélérer le déploiement et au réseau d'identifier plus efficacement les applications transportées pour mieux les gérer en termes de qualité de service, de sécurité et d'ingénierie de trafic **[11]**.

II.3.3 Architecture du SDN

Le SDN est une architecture qui permet de séparer les fonctions de contrôle et de transfert de données du réseau afin d'avoir une infrastructure physique totalement indépendante de tout service réseau. La base de cette architecture est la séparation du plan de contrôle et du plan de données (Figure.II.2) **[12]**.

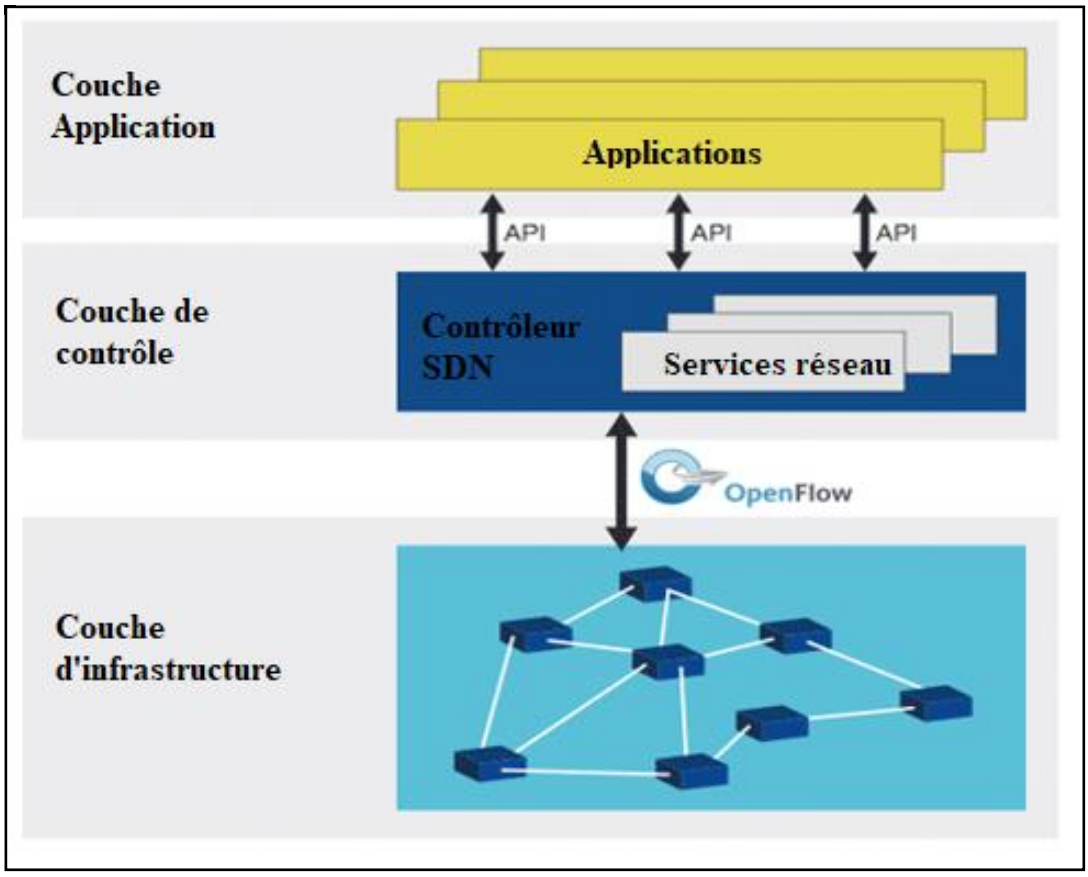


Figure. II.2 : Architecture de référence SDN proposé par l'ONF

II.3.3.1 Les couches de l'architecture SDN

L'architecture SDN repose sur trois couches fondamentales (Figure.II.2) :

- **Couche d'application** : C'est la couche supérieure de l'architecture SDN qui contient les programmes et outils nécessaires pour exécuter des logiciels, des protocoles de routage ou des ajouts au contrôleur.
- **Couche de contrôle** : C'est la couche intermédiaire de l'architecture SDN qui gère les logiciels ou les applications par un ou plusieurs contrôleurs reçus de la couche d'application et se concentre sur les équipements de la couche d'infrastructure tels que les commutateurs OpenFlow.
- **Couche d'infrastructure** : Qui contient les équipements réseau tels que les commutateurs OpenFlow. Dans cette couche, les commutateurs et les routeurs classiques ne sont pas utilisés, c'est-à-dire qu'il n'y a pas de table d'adresses MAC ni de tables de routage. Au lieu de cela, on utilise des OVS qui fonctionnent avec OpenFlow en utilisant des tables de flux (*plan de données*).

II.3.3.2 Interfaces entre couches SDN

Les interfaces dans un réseau SDN sont des éléments clés pour la communication et la gestion du réseau, permettant d'optimiser la performance, la sécurité et la qualité de service. Il existe trois types d'interfaces dans un réseau SDN représenté dans la Figure.II.3 : **[13]**

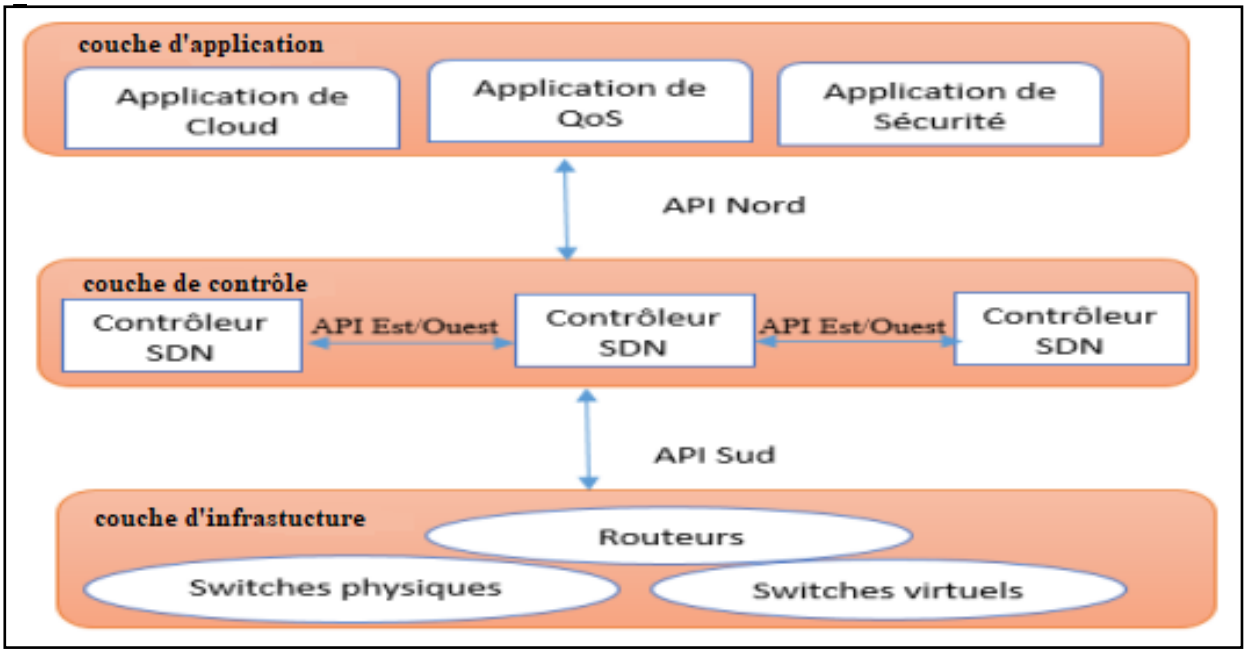


Figure. II.3 : Différents interfaces d'un réseau SDN

- **L'interface Nord** permet la communication entre le contrôleur et la couche applicative. Elle est considérée davantage comme une API plutôt qu'un protocole de programmation et de gestion de réseau.

- **L'interface Sud** permet la communication entre le contrôleur et les switchs et autres éléments de la couche infrastructure réseau. Cette interface utilise le protocole OpenFlow dans le cas du standard Open SDN pour injecter les différentes politiques aux équipements et récupérer les informations nécessaires à la construction d'une vue globale du réseau par les applications.
- **L'interface Est/Ouest** est utilisée dans les architectures distribuées pour permettre la communication entre contrôleurs afin de synchroniser les états du réseau. Il n'y a pas encore de standard disponible pour ce type d'interfaces.

II.4 Protocole OpenFlow

OpenFlow est un protocole de communication avancé entre un plan de contrôle centralisé (contrôleurs) et le plan de données (commutateurs de réseau). Il est standardisé par l'ONF une organisation fondée par les géants de la technologie informatique les plus respectés tels que Facebook, Google, Microsoft, Intel, Cisco et de nombreux autres (ONF, 2017).

Dans le contexte d'OpenFlow, le contrôleur prend des décisions de routage pour chaque flux de données, qui sont ensuite transmises aux commutateurs sous forme d'instructions de commutation simples. La séparation entre le plan de contrôle et le plan de données permet une transmission à très haut débit et permet aux composants responsables du plan de données d'être optimisés pour le transfert de paquets, évitant ainsi d'alourdir le traitement avec des fonctions de contrôle supplémentaires.

OpenFlow est un protocole qui opère dans la deuxième couche du modèle OSI, ce qui le distingue du protocole MPLS qui fonctionne à la fois dans la couche liaison de données et la couche réseau. Les commutateurs réseau sont les dispositifs les plus courants qui fonctionnent dans la couche liaison de données. Selon Goransson et Black (2014), trois composants sont nécessaires pour créer un réseau basé sur la technologie OpenFlow : un commutateur qui prend en charge ce protocole, un contrôleur et un canal de communication utilisé par le protocole OpenFlow par lequel le contrôleur et le commutateur peuvent communiquer **[14]**.

II.4.1 Composant d'OpenFlow

L'architecture réseau OpenFlow se compose d'un contrôleur et des switchs OpenFlow illustré dans la Figure.II.4 **[15]**.

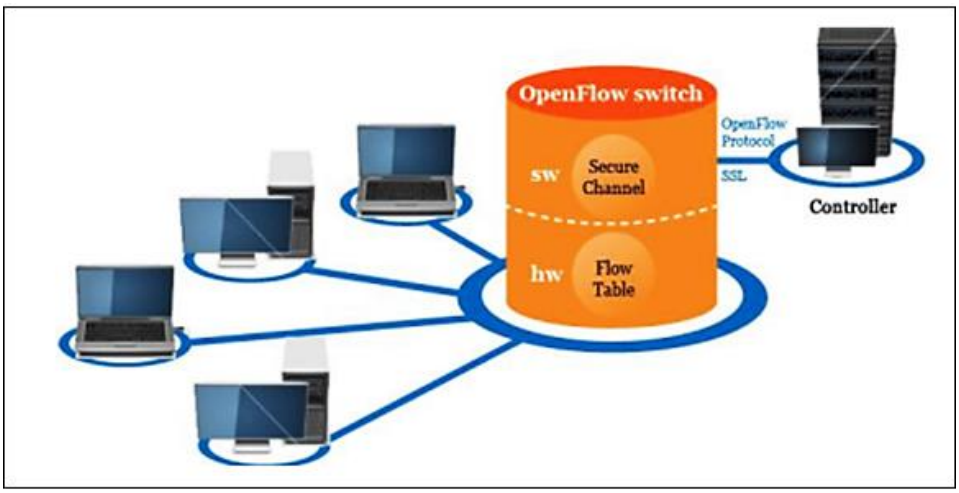


Figure. II.4 : Architecture réseau OpenFlow

II.4.1.1 Le contrôleur

Le contrôleur dans les réseaux SDN est une application qui agit comme une couche d'abstraction du réseau, permettant de le visualiser comme un système global. Il permet une gestion intelligente du réseau en traduisant rapidement les demandes en actions sur les équipements, grâce au protocole OpenFlow qui lui permet de communiquer avec les commutateurs et de configurer les périphériques réseau. Le contrôleur peut ainsi choisir le chemin optimal pour le trafic des applications et optimiser le débit. En résumé, le contrôleur joue le rôle de cerveau du réseau en faisant office de système d'exploitation et en offrant une vue centralisée complète, facilitant ainsi la gestion automatisée et l'intégration des applications d'entreprise. Le contrôleur peut fonctionner en deux modes opérationnels :

- **Mode réactif** : Dans ce modèle, tous les paquets qui arrivent au commutateur sont envoyés directement au contrôleur pour décider de leur traitement. Cependant, cette méthode peut causer une augmentation de la latence, une surcharge du contrôleur et le rendre sensible aux pannes.
- **Mode proactif** : Dans ce modèle, le contrôleur configure à l'avance des règles dans les commutateurs pour leur permettre de traiter localement les paquets, sans nécessiter d'une redirection vers le contrôleur. Cela permet de réduire le nombre de paquets envoyés au contrôleur, ce qui peut améliorer l'efficacité et le débit de l'ensemble de l'architecture réseau. Cette approche permet également de soulager le contrôleur, de diminuer la latence et d'améliorer les performances globales du réseau.

➢ **Type des contrôleurs**

De nombreux contrôleurs ont été créés, dont la plupart sont *open source* et prennent en charge le protocole OpenFlow. Ces contrôleurs présentent des différences en termes de langage de programmation, de version d'OpenFlow supportée, de techniques utilisées (multitâche) ainsi que de performances (débit). Voici une liste non exhaustive des contrôleurs SDN disponibles :

- **NOX et POX** : Ont été les deux premiers contrôleurs OpenFlow.
- **Ryu** : Est un contrôleur SDN capable de configurer les équipements réseau en utilisant plusieurs protocoles, notamment OpenFlow, NetConf et OF-config.
- **Floodlight :** Un contrôleur SDN largement utilisé développé par Big Switch Networks et disponible en open source, est un contrôleur OpenFlow basé sur Java sous licence Apache. Son architecture modulaire comprend des composants tels que la gestion de la topologie, la gestion des périphériques, le calcul de chemin, le stockage de données et une abstraction de stockage généralisée **[11]**.
- **OpenDaylight** : Est un projet open source soutenu par plusieurs fournisseurs de réseaux. Cette plateforme de contrôle SDN est développée en Java et a été créée pour aborder le défi critique du contrôleur dans une architecture SDN. L'objectif du consortium derrière ce projet est de rassembler les efforts de l'industrie et du monde académique pour développer un contrôleur réellement utilisable. Il est conçu pour être déployé sur n'importe quelle plate-forme matérielle et système d'exploitation.
- **ONOS (Open Networking Operating System) :** Il s'agit d'un nouveau système d'exploitation réseau ou contrôleur SDN qui a récemment été mis à disposition. Son principal objectif est de répondre aux besoins des fournisseurs de services. Le système est conçu pour fonctionner sous forme de cluster, avec différentes couches, ce qui lui permet d'être évolutif, hautement disponible et d'offrir des performances optimales. De plus, il propose des abstractions qui facilitent la création d'applications et de services **[9]**.

II.4.1.2 Switchs OpenFlow

Un commutateur OpenFlow est un composant d'infrastructure réseau qui fonctionne dans la deuxième couche du modèle OSI et qui contient dans sa mémoire des tables de flux (Figure.II.5) et dispose d'un canal de communication qui peut être utilisé pour communiquer avec le contrôleur **[15]**.

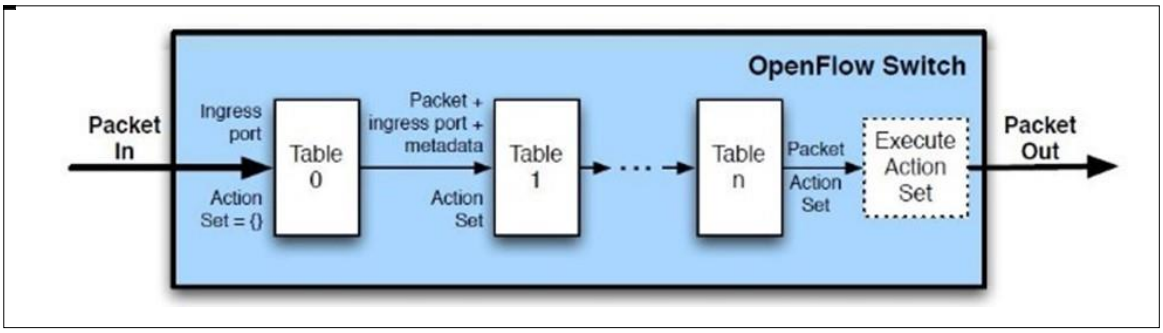


Figure. II.5 : Table OpenFlow

La table de flux se compose de trois éléments principaux : *les champs d'en-tête* qui sont créés à partir de l'en-tête de paquet, *les compteurs* qui fournissent des informations statistiques telles que le nombre de paquets et d'octets envoyés et le temps depuis le dernier paquet correspondant à la règle, *les champs d'action* qui spécifient la manière dont le paquet a été traité. Les entrées y sont ajoutées via le contrôleur. Elles précisent comment le switch doit se comporter après avoir reçu le paquet qui correspond à la condition de correspondance. Le switch peut envoyer des données au port de sortie, le rejeter ou l'envoyer au contrôleur. Cette dernière action est le plus souvent effectuée lorsque le switch, après avoir reçu le paquet, est incapable de le faire correspondre à l'une des règles existantes. Dans cette situation, le contrôleur prend la décision appropriée sur les règles à ajouter à la table de flux pour que l'action contre le paquet similaire suivant soit prise uniquement par le switch. La règle appropriée est ajoutée à sa table de flux dans le switch via le contrôleur. Les fonctions du contrôleur peuvent être assurées par un programme très sophistiqué qui contrôle le flux selon certains critères ou par un mécanisme très simple qui ajoutera des entrées statiques à la mémoire du switch.

Le canal de communication est un élément très important dans les réseaux basés sur la technologie OpenFlow. Il est utilisé pour faciliter la communication entre les commutateurs et les contrôleurs, ce qui est crucial car les décisions réelles sur la gestion du trafic réseau sont prises par le contrôleur et doivent être propagées aux commutateurs. Les données envoyées via ce canal doivent être conformes à la spécification OpenFlow et sont généralement cryptées à l'aide de Transport Layer Security (TLS).

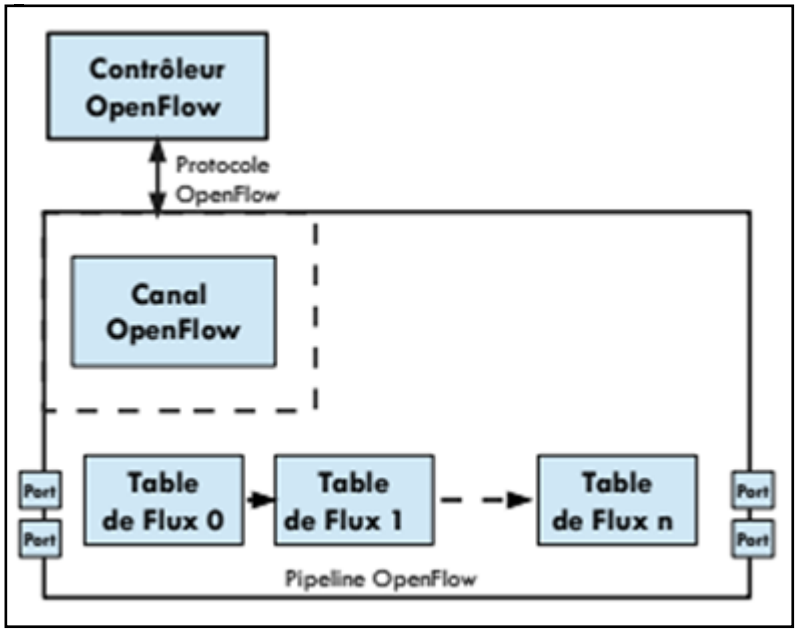


Figure. II.6 : Anatomie switch OpenFlow

La plupart des contrôleurs prennent en charge la version OpenFlow 1.3(.2) plutôt que 1.4 ou 1.5. Le commutateur OpenFlow peut prendre en charge une ou plusieurs tables de flux dans le pipeline, mais n'a besoin de prendre en charge qu'une seule table pour être conforme à la norme. Cependant, l'utilisation de plusieurs tables permet une évolutivité pour les infrastructures plus importantes.

Le canal OpenFlow du commutateur fournit une connexion au contrôleur externe (Figure.II.6).

Le contrôleur est dissocié du commutateur dans le plan de contrôle, généralement exécuté sur une boîte Linux et il gère le commutateur via OF pour ajouter, mettre à jour et supprimer des entrées de flux de manière réactive ou proactive.

Chaque commutateur peut avoir jusqu'à 254 tables de flux et la correspondance des paquets commence sur la table de flux 0 qui était initialement prise en charge dans la version OF 1.0, tandis que d'autres versions prennent en charge plusieurs tables dans le pipeline en utilisant des instructions goto-table. Les entrées de flux sont appariées par ordre de priorité de plus élevée à plus basse lors de l'exécution des instructions et si aucune entrée n'est appariée, une entrée de table manquante avec une priorité de 0 est utilisée.

Il est possible de différencier deux types de switch OpenFlow :

- **OpenFlow-only** (Pure) : Le switch ne prend en charge que les actions nécessaires pour faire fonctionner le protocole OpenFlow.
- **OpenFlow-enabled** (hybride) : Ce switch, compatible avec OpenFlow, est capable de supporter les actions standards d'un switch classique en plus des actions nécessaires au fonctionnement du protocole OpenFlow.

II.4.2 Messages OpenFlow

II.4.2.1 Messages symétrique

Les messages symétriques peuvent être envoyés par le contrôleur ou le switch sans avoir été sollicités par l'autre. Les messages HELLO sont échangés après l'établissement du canal sécurisé pour déterminer le numéro de version OpenFlow le plus élevé pris en charge par les pairs. Le protocole spécifie que la version la plus basse doit être utilisée pour la communication contrôleur-switch sur cette instance de canal sécurisé. Les messages ECHO sont utilisés par l'une ou l'autre des parties pendant la durée de vie du canal pour vérifier que la connexion est toujours active et pour mesurer la latence ou la bande passante actuelle de la connexion. Les messages VENDOR sont disponibles pour des expérimentations ou des améliorations spécifiques au vendeur.

II.4.2.2 Messages asynchrones

Les messages asynchrones sont envoyés du switch au contrôleur sans avoir été sollicités par le contrôleur. Le message PACKET_IN est la façon dont le switch renvoie les paquets de données au contrôleur pour une gestion d'exception. Le trafic du plan de contrôle sera généralement renvoyé au contrôleur via ce message. Le switch peut informer le contrôleur qu'une entrée de flux est supprimée de la table de flux via le message FLOW_REMOVED. PORT_STATUS est utilisé pour communiquer les changements d'état de port, que ce soit par intervention directe de l'utilisateur ou par un changement physique dans le support de communication lui-même. Enfin, le switch utilise le message ERROR pour notifier le contrôleur des problèmes.

II.4.2.3 Messages contrôleur-switch

Le contrôleur-switch est la catégorie la plus large de messages OpenFlow (Figure.II.7). En fait ils peuvent être divisés en cinq sous-catégories : configuration du switch, commande du contrôleur, statistiques, configuration de file d'attente et barrière. Les messages de configuration du switch consistent en un message de configuration unidirectionnel et deux paires de messages de demande-réponse. Le contrôleur utilise le message unidirectionnel SET_CONFIG pour définir les paramètres de configuration dans le switch. Le message SET_CONFIG envoyé pendant la phase d'initialisation du dialogue contrôleur-switch. Le contrôleur utilise la paire de messages FEATURES pour interroger le switch sur les fonctionnalités qu'il prend en charge. De même, la paire de messages GET_CONFIG est utilisée pour récupérer les paramètres de configuration d'un switch **[15]**.

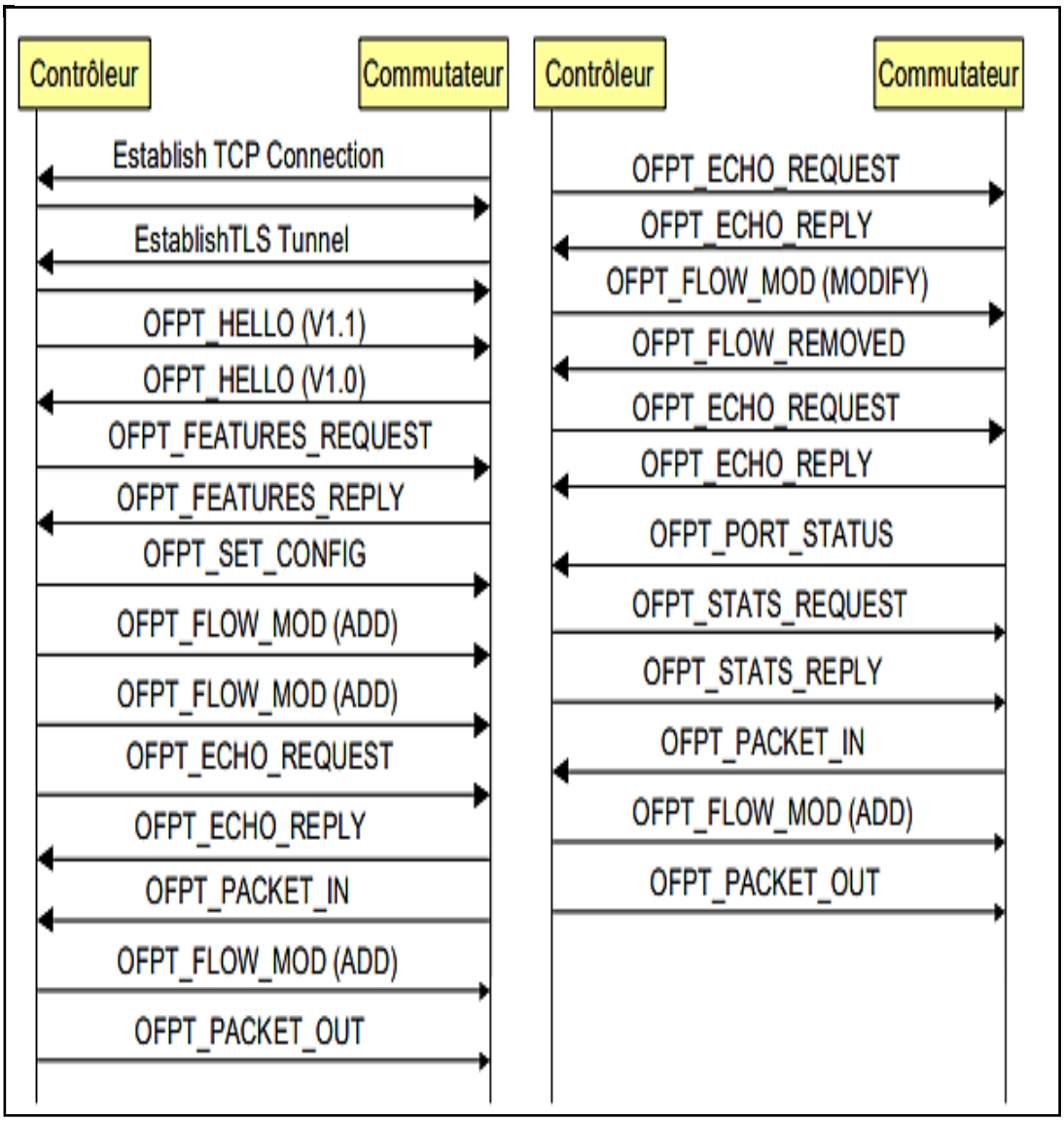


Figure. II.7 : Différents messages entre contrôleur et switch

II.4.3 Fonctionnement OpenFlow

La Figure.II.8 représente le fonctionnement de l'architecture OpenFlow lors de la transmission d'un paquet.

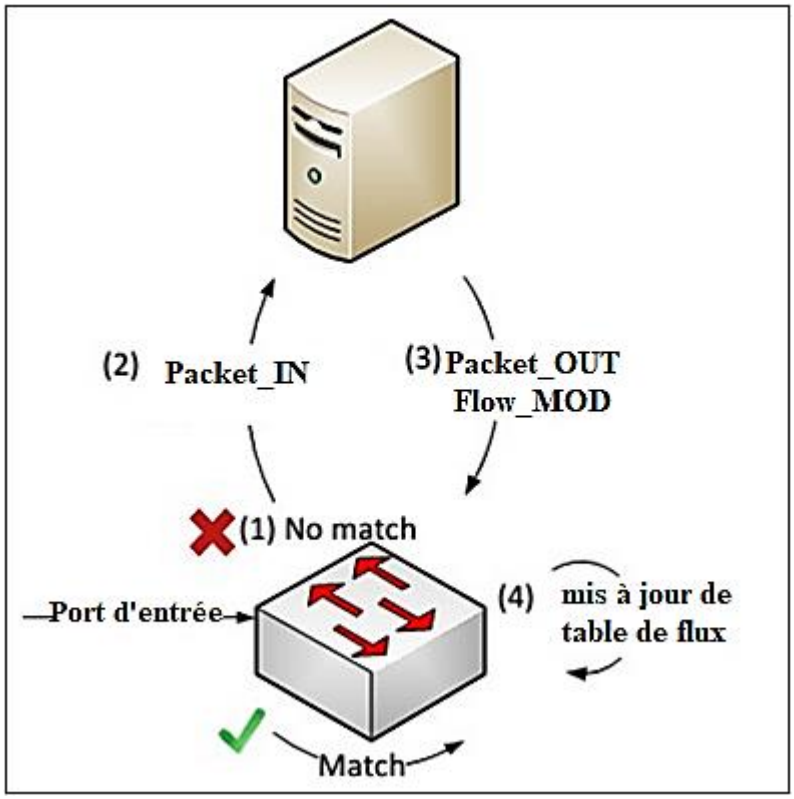


Figure. II.8 : Fonctionnement OpenFlow

Lorsqu'un paquet atteint un commutateur, ce dernier vérifie s'il existe une entrée correspondante dans sa table de flux en fonction de l'en-tête du paquet. Si une correspondance est trouvée, le commutateur exécute l'action associée à cette entrée dans la table de flux. Dans le cas contraire, c'est-à-dire s'il n'y a pas d'entrée correspondante (1), le commutateur génère un message asynchrone appelé "Packet_IN" et l'envoie au contrôleur (2). Le contrôleur, en fonction de sa configuration, prend une décision concernant l'action à prendre pour ce paquet, puis envoie une nouvelle règle de transmission au commutateur sous la forme de messages "Packet_OUT" et "Flow_MOD" (3). Enfin, la table de flux du commutateur est mise à jour pour intégrer la nouvelle règle établie par le contrôleur (4) **[15]**.

II.5 Avantages du SDN

Le réseau défini par logiciel (SDN) présente plusieurs avantages, notamment : **[9]**

- ✓ Gestion centralisée : Avec SDN, la gestion du réseau peut être centralisée et automatisée, ce qui permet une configuration plus rapide et une résolution plus efficace des problèmes.
- ✓ Flexibilité : Les réseaux SDN sont plus flexibles et adaptables que les réseaux traditionnels, car les règles de gestion peuvent être modifiées rapidement pour répondre aux besoins changeants des applications et des utilisateurs.
- ✓ Réduction des coûts : Les réseaux SDN peuvent réduire les coûts d'exploitation en automatisant les tâches répétitives et en simplifiant la gestion du réseau.
- ✓ Meilleure sécurité : La sécurité est renforcée dans les réseaux SDN grâce à la possibilité de segmenter le trafic, de contrôler l'accès et de détecter rapidement les menaces.
- ✓ Amélioration des performances : Les réseaux SDN peuvent améliorer les performances en optimisant l'utilisation des ressources réseau et en éliminant les goulets d'étranglement.
- ✓ Innovation : Les réseaux SDN permettent l'innovation en offrant une plateforme pour le développement d'applications réseau personnalisées et d'outils de gestion.
- ✓ L'automatisation des réseaux provenant de différents fournisseurs facilite le déploiement rapide et à grande échelle des services réseau, tout en réduisant les risques d'erreurs humaines.

II.6 Conclusion

Ce chapitre a permis de se familiariser avec les fondamentaux de la virtualisation, ainsi que les différentes formes de cette technologie et les avantages qu'elle offre. Nous avons également abordé l'architecture SDN, avec une mise en évidence particulière du protocole OpenFlow. Cela nous permettra de mieux comprendre la configuration et la simulation des systèmes MPLS et SDN qui seront présentées dans le dernier chapitre.

CHAPITRE III

Implémentation et Résultats

III.1 Introduction

Dans le cadre de notre projet de fin d'études (PFE) nous allons proposer une étude comparative des performances entre deux technologies de réseau bien connues : MPLS et OpenFlow (SDN). Dans ce chapitre, nous nous concentrons sur la partie pratique de notre travail où nous allons implémenter deux topologies de réseau distinctes.

L'objectif principal de cette partie pratique était d'évaluer et de comparer les performances de MPLS et OpenFlow dans plusieurs aspects clés du réseau comme : le forwarding du trafic, la scalabilité (Scale-up), mesure de la latence RTT, mesure de débit (Troughput) et mesure de la gigue (Jitter). Ces résultats nous permettront de mieux comprendre les avantages et les limitations de chaque technologie, ce qui peut être utile pour les décisions de conception et de déploiement futurs. Dans les sections suivantes de ce chapitre, nous présenterons en détail la méthodologie utilisée pour la mise en place des topologies MPLS et OpenFlow.

III.2 Environnement de tests

L'implémentation des deux solutions est réalisée dans un environnement virtuel plus précisément dans VMWARE Workstation Pro17 dans une configuration hardware suivante : PC portable Intel® Core™ i5-8350U, RAM 16G et Disque dur SSD 256G.

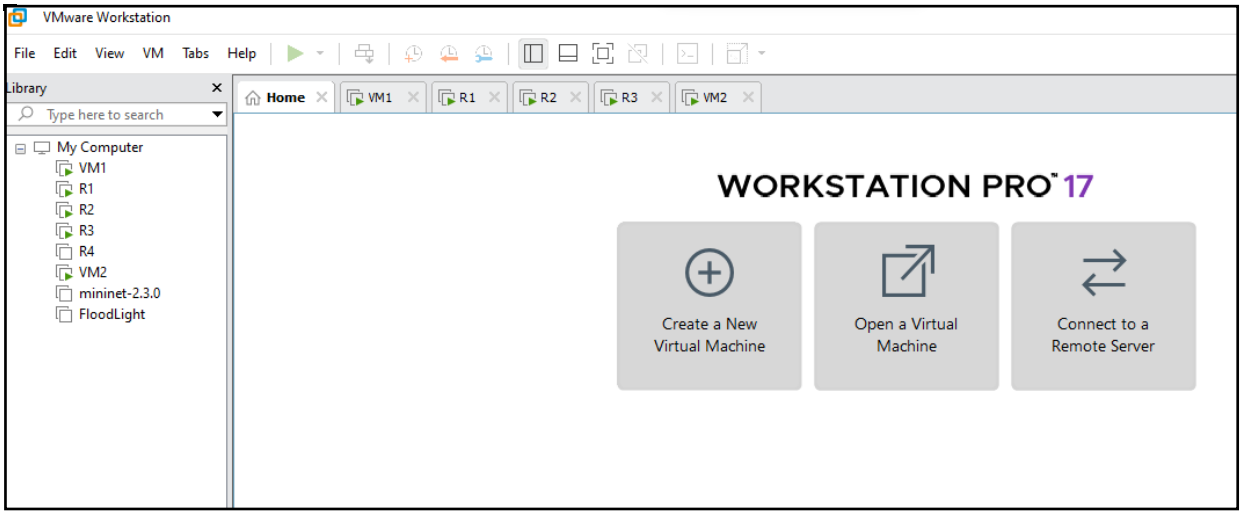


Figure.III.1 : Interfaces de logiciel VMware

Pour la solution MPLS nous avons utilisé des routeurs Cisco CSR1000v (Cloud Services Router) pour implémenter le backbone MPLS et des hôtes clients sous Ubuntu 16.04 LTS pour les sites clients (chaque équipement est installé dans une machine virtuelle autonome).

Et concernant la solution OpenFlow (SDN) dans le même environnement virtuel nous avons utilisé l'outil mininet pour déployer les switchs OVS (Open Virtual Switch) et les hôtes. Et pour le contrôleur SDN Floodlight a été installé dans une machine virtuelle indépendante.

Et pour enrichir notre étude comparative en ce qui concerne chaque approche en termes de performance nous avons utilisé les outils de tests Iperf et Wireshark dans les différents scénarios de réseau.

Cette approche pratique nous permettra d'approfondir notre compréhension des technologies de réseau avancées et de leur application dans des environnements réels.

Tous les composants et les outils utilisés dans les deux solutions sont détaillés ci-dessous.

N.B : Les étapes d'installation de chaque packages sont présentées en détail dans l'annexe **B**.

III.2.1 VMware Workstation

C'est un logiciel qui permet de créer des machines virtuelles pour exécuter différents systèmes d'exploitation sur un même ordinateur hôte. Chaque machine virtuelle peut exécuter une instance unique de tout système d'exploitation, tel que Microsoft Windows, Linux ou un autre OS. Ce logiciel est conçu pour assurer une compatibilité matérielle optimale et agit comme un intermédiaire entre l'ordinateur hôte et la machine virtuelle pour permettre l'utilisation de diverses ressources matérielles (Cartes réseaux, disques durs, périphériques USB et lecteurs CD-ROM.

Cependant, VMware a continué à innover en utilisant le programme "Virtual Network Editor" fourni avec VMware Workstation. Vous pouvez également virtualiser des commutateurs avec ou sans serveur DHCP intégré et les connecter à divers réseaux et/ou à Internet. Une fois que vous maîtriserez cette fonctionnalité, vous serez capable de virtualiser des infrastructures complexes afin de tester efficacement votre futur environnement.

Cet outil Virtual Network intègre plusieurs modes de fonctionnement :

- *Mode Bridged* : Connecte les machines virtuelles à votre réseau local physique, leur permettant de se comporter comme des machines physiques indépendantes ;
- *Mode NAT* : Permet de connecter les machines virtuelles à Internet via la connexion réseau du PC hôte ;
- *Mode Host-only* : Permet de créer un réseau virtuel reliant uniquement le PC hôte et la machine virtuelle ;
- *LAN segments (ou VLANs)* : Permettent de relier certaines machines virtuelles ;
- *Programme Virtual Network Editor de VMware* : Permet de créer des réseaux virtuels personnalisés **[16]**.

Plusieurs outils ont été installés sur VMware Workstation afin de simuler les deux technologies :

III.2.2 VM Ubuntu 16.04 LTS

C'est une machine virtuelle qui permet d'exécuter le système d'exploitation Ubuntu 16.04 LTS sur une plateforme de virtualisation (VMware Workstation). Elle fonctionne comme un ordinateur autonome possédant son propre processeur, sa propre mémoire vive, son propre disque dur virtuel et ses propres périphériques d'entrée/sortie. Cette isolation du système d'exploitation hôte offre plus de flexibilité pour les tests, le développement et les opérations **[17]**.

III.2.3 Cisco Cloud Services Router 1000v

Le Cisco Cloud Services Router 1000v (CSR 1000v) est un routeur virtuel qui fournit des fonctions de passerelle WAN et de services réseau pour les environnements virtuels et de cloud. Il utilise des capacités de mise en réseau de Cisco IOS XE Software pour permettre aux entreprises d'étendre leur WAN dans des clouds hébergés par des fournisseurs. Il permet

également aux fournisseurs de cloud de fournir des services de mise en réseau de qualité entreprise à leurs locataires ou clients **[18]**.

III.2.4 Mininet

C'est un logiciel *open-source* utilisé pour créer des réseaux virtuels et expérimenter des réseaux OpenFlow dans l'enseignement, la recherche et le développement. Il peut être exécuté sur des machines physiques ou virtuelles tels que VMware Workstation. Il est particulièrement adapté pour l'innovation dans le domaine du SDN grâce à son interface programmable Python et sa CLI (Figure.III.2). Il fonctionne sur le noyau de Linux et utilise Python pour son API permettant aux utilisateurs de personnaliser les topologies en fonction de leurs besoins.

Mininet peut émuler plusieurs hôtes et switchs OVS et créer des réseaux complexes sur une seule machine grâce à la virtualisation réseau de Linux **[19]**.

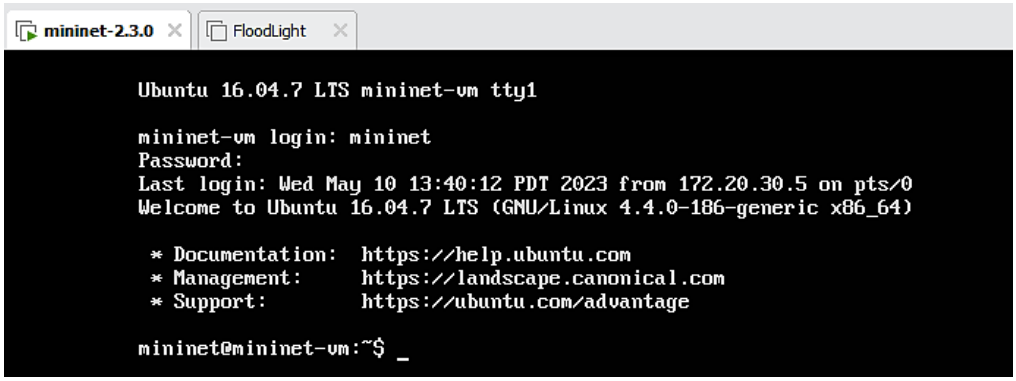


Figure.III.2 : Interface CLI de Mininet

III.2.5 Floodlight Contrôleur

Floodlight agit comme un "cerveau" centralisé pour les commutateurs OpenFlow dans un réseau SDN. Il reçoit des informations de l'état du réseau à partir des commutateurs, prend des décisions sur le chemin de transmission du trafic et envoie ensuite des instructions aux commutateurs pour configurer leur comportement. De plus, il inclut un circuit (pusher) basé sur Python qui automatise l'installation de règles OpenFlow persistantes pour établir une connexion entre deux hôtes. Son interface graphique est illustrée dans la Figure.III.3 **[20]**.

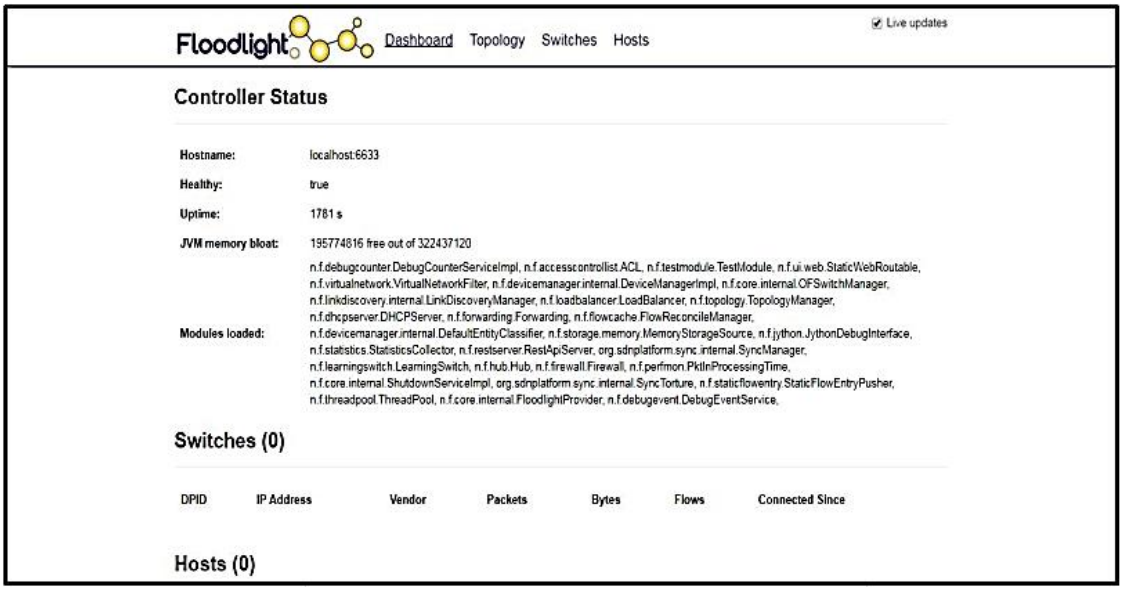


Figure.III.3 : Interface Graphique de Floodlight

Nous avons installé Floodlight sur une machine virtuelle exécutant Ubuntu 14.04.6 avec une adresse IP 172.20.30.254

III.2.6 Outil génération de trafic IPERF

IPERF est un outil *open-source* pour mesurer le débit et la qualité de transmission des données sur un réseau IP. Il permet d'effectuer des tests de performance entre deux nœuds. Iperf fonctionne en mode client-serveur et peut être utilisé pour tester la performance de différents protocoles de communication : TCP, UDP **[21]**.

III.2.7 Outil de capture de trafic Wireshark

C'est un logiciel *open-source* utilisé pour la capture et l'analyse de paquets réseau. Il offre la possibilité d'effectuer des analyses approfondies du trafic réseau, ce qui permet de résoudre des problèmes de connectivité, de performances ou de sécurité. Wireshark prend en charge la capture et l'analyse de divers protocoles réseau et offre des fonctionnalités avancées telles que l'analyse de la qualité de service, la détection d'intrusions, la reconstruction de sessions et l'exportation de données. Il dispose également d'une interface graphique (Figure.III.4) facile à utiliser **[22]**.

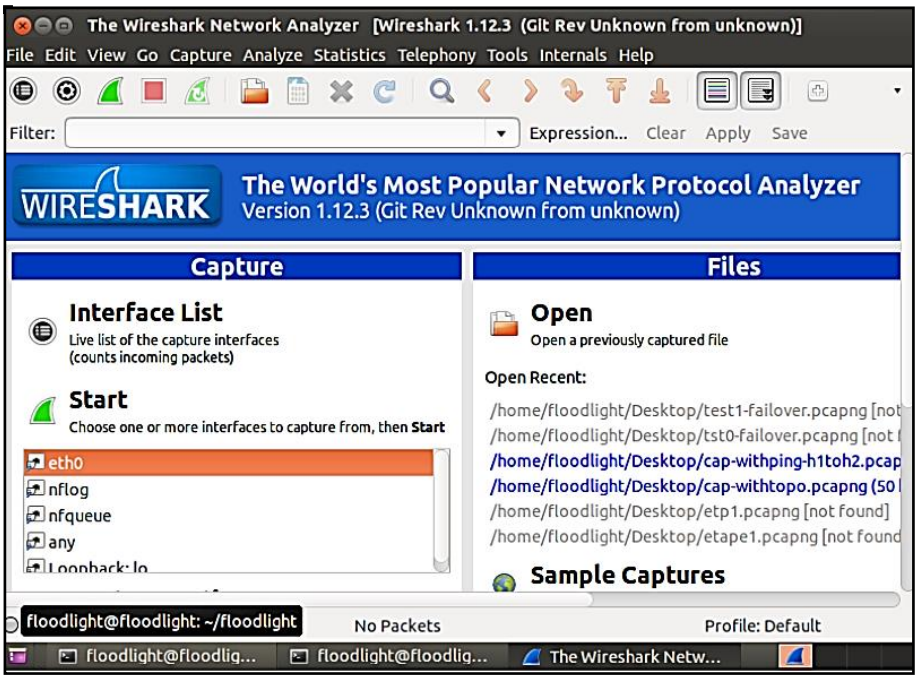


Figure.III.4 : Interfaces graphique de Wireshark

III.3 Implémentation de la topologie

Le but de notre travail est d'implémenter une topologie contenant un backbone Core et dans les extrémités il y a deux machines clientes pour communiquer entre eux en traversant ce backbone. Le design de la même topologie sera appliqué dans les deux solutions MPLS et OpenFlow (SDN).

La première topologie et vue la configuration hardware du PC que nous avons utilisé ; pour le backbone MPLS nous avons implémenté que trois routeurs Cisco CSR1000v et pour les sites clients nous avons utilisé deux machines virtuelles (VM).

L'objectif principal de cette topologie est de mettre en œuvre un réseau MPLS afin d'offrir une commutation de couche 2 et un acheminement de couche 3 améliorés. Nous explorerons les mécanismes de création de tunnels et d'étiquetage de paquets pour faciliter la commutation et le routage efficaces dans le réseau MPLS.

Concernant la deuxième topologie OpenFlow SDN nous avons utilisé l'environnement Mininet pour implémenter les switchs OVS et deux hôtes clientes et dans une autre VM le contrôleur Floodlight. Ce contrôleur permet de gérer et contrôler le comportement des commutateurs OpenFlow dans notre réseau. Cette topologie nous permet d'explorer les concepts d'architecture de réseau programmable et de contrôle centralisé offerts par OpenFlow.

III.3.1 Topologie MPLS

Dans la partie MPLS nous proposons une configuration réseau où un backbone IP MPLS est utilisé, composé de 3 routeurs et de 2 machines virtuelles sous Linux/Ubuntu. Dans le but d'interconnecter ces deux machines, nous avons mis en place un L3/VPN en utilisant des VRF et le protocole MP-BGP.

Notre solution est présentée sur la Figure.III.5. Les différentes étapes de configuration et d'exécution seront abordées dans le point suivant.

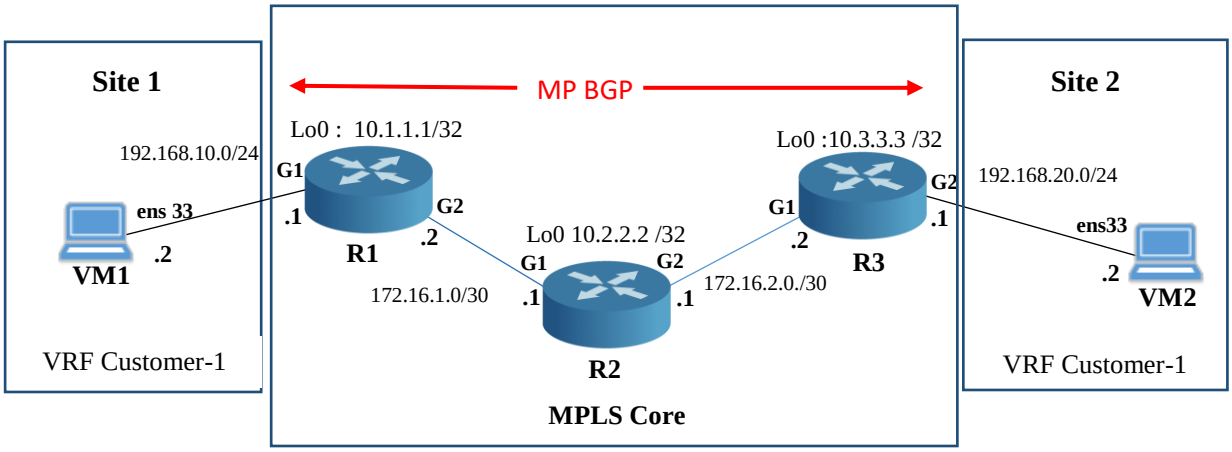


Figure.III.5 : Diagramme de topologie MPLS

A. Configuration des routeurs

1. Adressage IP de MPLS Core et OSPF :

Qui consiste à configurer l'adressage (physique et loopback) des routeurs du nuage MPLS ainsi que le routage IGP (Interior Gateway) en utilisant le protocole de routage OSPF pour assurer la connectivité entre R1 et R3.

Voici le test de Ping entre les adresses loopback de R1, R2 et R3 (Figure.III.6) :

```
R1#
R1#ping 10.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 3/3/4 ms
R1#ping 10.3.3.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/4/6 ms
R1#_
```

Figure.III.6 : Ping entre les routeurs

2. Configurez le protocole de distribution des étiquettes (Label Distribution Protocol - LDP) sur tous les routeurs mpls en ajoutant la commande (mpls ldp autoconfig) dans le process OSPF elle activera le protocole de distribution d'étiquettes mpls sur chaque interface.

Pour vérifier les interfaces mpls, nous utilisons la commande : ***sh mpls interface***

```
R1#sh mpls interfaces
Interface              IP           Tunnel   BGP Static Operational
GigabitEthernet2       Yes (ldp)    No       No  No     Yes
R1#
```

```
R2#sh mpls interfaces
Interface              IP           Tunnel   BGP Static Operational
GigabitEthernet1       Yes (ldp)    No       No  No     Yes
GigabitEthernet2       Yes (ldp)    No       No  No     Yes
R2#_
```

```
R3#sh mpls interfaces
Interface              IP           Tunnel   BGP Static Operational
GigabitEthernet1       Yes (ldp)    No       No  No     Yes
R3#_
```

Figure.III.7 : Vérification des interfaces MPLS dans les routeurs

Vous pouvez également vérifier les voisins LDP avec la commande ***sh mpls ldp neighbors***

```
R1#sh mpls ldp neighbor
    Peer LDP Ident: 10.2.2.2:0; Local LDP Ident 10.1.1.1:0
        TCP connection: 10.2.2.2.44740 - 10.1.1.1.646
        State: Oper; Msgs sent/rcvd: 31/31; Downstream
        Up time: 00:20:23
        LDP discovery sources:
          GigabitEthernet2, Src IP addr: 172.16.1.1
          GigabitEthernet2, Src IP addr: 172.16.2.1
        Addresses bound to peer LDP Ident:
          172.16.1.1      172.16.2.1      10.2.2.2
    Peer LDP Ident: 10.3.3.3:0; Local LDP Ident 10.1.1.1:0
        TCP connection: 10.3.3.3.40486 - 10.1.1.1.646
        State: Oper; Msgs sent/rcvd: 14/14; Downstream
        Up time: 00:05:47
        LDP discovery sources:
          GigabitEthernet2, Src IP addr: 172.16.2.2
        Addresses bound to peer LDP Ident:
          172.16.2.2      10.3.3.3
R1#_
```

(a)

```
R2#sh mpls ldp neighbor
    Peer LDP Ident: 10.1.1.1:0; Local LDP Ident 10.2.2.2:0
        TCP connection: 10.1.1.1.646 - 10.2.2.2.44740
        State: Oper; Msgs sent/rcvd: 30/30; Downstream
        Up time: 00:20:03
        LDP discovery sources:
          GigabitEthernet1, Src IP addr: 172.16.1.2
          GigabitEthernet2, Src IP addr: 172.16.1.2
        Addresses bound to peer LDP Ident:
          172.16.1.2      10.1.1.1
    Peer LDP Ident: 10.3.3.3:0; Local LDP Ident 10.2.2.2:0
        TCP connection: 10.3.3.3.30794 - 10.2.2.2.646
        State: Oper; Msgs sent/rcvd: 14/14; Downstream
        Up time: 00:05:27
        LDP discovery sources:
          GigabitEthernet1, Src IP addr: 172.16.2.2
          GigabitEthernet2, Src IP addr: 172.16.2.2
        Addresses bound to peer LDP Ident:
          172.16.2.2      10.3.3.3
R2#_
```

(b)

```
R3#sh mpls ldp neighbor
    Peer LDP Ident: 10.2.2.2:0; Local LDP Ident 10.3.3.3:0
        TCP connection: 10.2.2.2.646 - 10.3.3.3.30794
        State: Oper; Msgs sent/rcvd: 13/14; Downstream
        Up time: 00:05:06
        LDP discovery sources:
          GigabitEthernet1, Src IP addr: 172.16.1.1
          GigabitEthernet1, Src IP addr: 172.16.2.1
        Addresses bound to peer LDP Ident:
          172.16.1.1      172.16.2.1      10.2.2.2
    Peer LDP Ident: 10.1.1.1:0; Local LDP Ident 10.3.3.3:0
        TCP connection: 10.1.1.1.646 - 10.3.3.3.40486
        State: Oper; Msgs sent/rcvd: 13/13; Downstream
        Up time: 00:05:06
        LDP discovery sources:
          GigabitEthernet1, Src IP addr: 172.16.1.2
        Addresses bound to peer LDP Ident:
          172.16.1.2      10.1.1.1
R3#_
```

(c)

Figure.III.8 (a), (b), (c) : Vérification des voisins LDP dans R3

3. Configurez le Multi-Protocol BGP (MP-BGP), sur les routeurs R1 et R3 pour permettre l'échange de routes VPN entre les sites distants.

Pour vérifier la session BGP entre R1 et R3, on exécute la commande ***sh bgp vpnv4 unicast all summary***

```
R1#sh bgp vpnv4 unicast all summary
BGP router identifier 10.1.1.1, local AS number 1
BGP table version is 5, main routing table version 5
2 network entries using 512 bytes of memory
2 path entries using 240 bytes of memory
2/1 BGP path/bestpath attribute entries using 528 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1304 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State
/PfxRcd
10.3.3.3        4            1      25      27        1    0    0 00:19:49
  1
R1#_
```

```
R3#sh bgp vpnv4 unicast all summary
BGP router identifier 10.3.3.3, local AS number 1
BGP table version is 4, main routing table version 4
2 network entries using 512 bytes of memory
2 path entries using 240 bytes of memory
2/2 BGP path/bestpath attribute entries using 528 bytes of memory
1 BGP extended community entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 1304 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State
/PfxRcd
10.1.1.1        4            1      15      15        4    0    0 00:08:47
  1
R3#
```

Figure.III.9 : Vérification BGP dans R1 et R3

4. Création des instances VRF pour le VPN-Custumer-1 et la configuration de l'adressage IP dans les interfaces des routeurs R1 R3 (gateway des sites du client) et les associés avec les VRF concernés.
L'utilisation des VRFs permet à plusieurs instances d'une table de routage de coexister dans un routeur mais sans interférer les unes avec les autres, la distinction entre plusieurs VRFs se fait à l'aide des Route Distinguisher.
Aussi associer les interfaces des routeurs R1 et R3 (Gateway des deux sites clients) avec la VRF VPN-Custumer-1 par la commande :

```
interface GigabitEthernet X
ip vrf forwarding Customer-1
ip address x.x.x.x x.x.x.x
```

5. Configurez les politiques d'exportation et d'importation de routes VPN sur les routeurs R1 et R3 pour contrôler les routes L3 VPN qui sont partagées entre les sites distants (dans notre cas la redistribution des routes sera effectué en mode *directly connected*)

N.B : Les commandes de la configuration des routeurs sont citées dans l'Annexe **A**.

B. Configuration des machines virtuelles (Côté Client)

Et finalement nous passons à la configuration de l'adressage IP des hôtes clients VM1 et VM2 en ouvrant le terminal en utilisons cette commande :

sudo nano /etc/network/interfaces

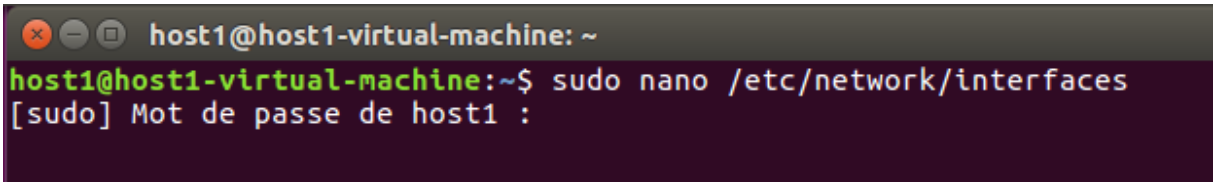


Figure.III.10 : Commande utilisé en VM1

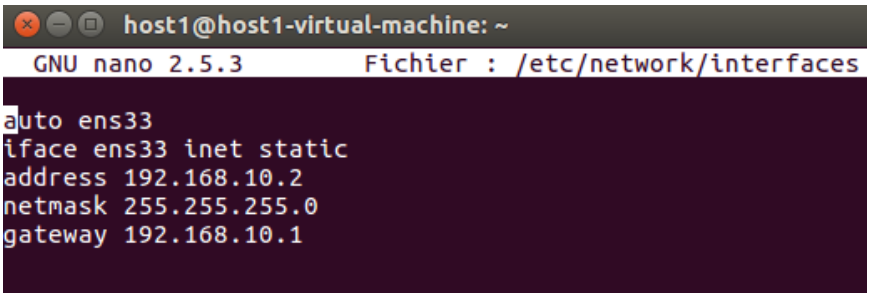


Figure.III.11 : Adresse IP VM1

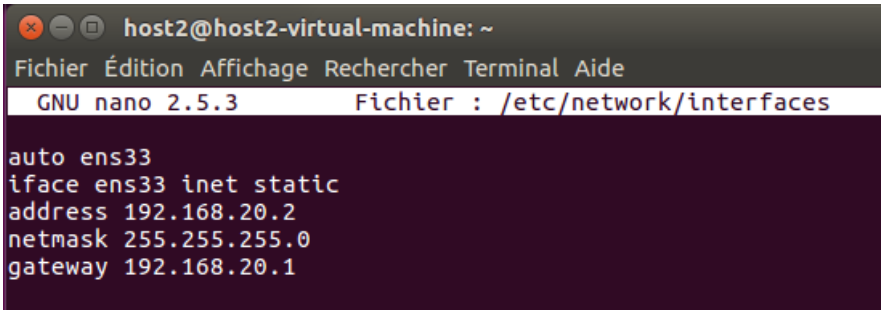


Figure.III.12 Adresse IP VM2

C. Test Connectivite après configuration

Comme il est indiqué précédemment dans les routeurs R1 et R3 nous aurons une table de routage globale ainsi qu'une table de routage de la VRF VPN-Custumer-1.
En utilisant la commande ***show ip route*** dans R1, R2 et R3 nous aurons (Figures.III.13) :

```
R1#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/32 is subnetted, 3 subnets
C        10.1.1.1 is directly connected, Loopback0
O        10.2.2.2 [110/2] via 172.16.1.1, 00:55:22, GigabitEthernet2
O        10.3.3.3 [110/3] via 172.16.1.1, 00:55:22, GigabitEthernet2
     172.16.0.0/16 is variably subnetted, 3 subnets, 2 masks
C        172.16.1.0/30 is directly connected, GigabitEthernet2
L        172.16.1.2/32 is directly connected, GigabitEthernet2
O        172.16.2.0/30 [110/2] via 172.16.1.1, 00:55:22, GigabitEthernet2
R1#
```

```
R2#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/32 is subnetted, 3 subnets
O        10.1.1.1 [110/2] via 172.16.1.2, 01:51:54, GigabitEthernet1
C        10.2.2.2 is directly connected, Loopback0
O        10.3.3.3 [110/2] via 172.16.2.2, 01:49:44, GigabitEthernet2
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
C        172.16.1.0/30 is directly connected, GigabitEthernet1
L        172.16.1.1/32 is directly connected, GigabitEthernet1
C        172.16.2.0/30 is directly connected, GigabitEthernet2
L        172.16.2.1/32 is directly connected, GigabitEthernet2
R2#
```

```
R3#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      10.0.0.0/32 is subnetted, 3 subnets
O        10.1.1.1 [110/3] via 172.16.2.1, 01:51:56, GigabitEthernet1
O        10.2.2.2 [110/2] via 172.16.2.1, 01:51:56, GigabitEthernet1
C        10.3.3.3 is directly connected, Loopback0
     172.16.0.0/16 is variably subnetted, 3 subnets, 2 masks
O        172.16.1.0/30 [110/2] via 172.16.2.1, 01:51:56, GigabitEthernet1
C        172.16.2.0/30 is directly connected, GigabitEthernet1
L        172.16.2.2/32 is directly connected, GigabitEthernet1
R3#
```

Figure.III.13 : Table de routage global des routeurs

En utilisant la commande ***show ip route vrf customer-1*** dans R1 et R3 nous aurons (Figure.III.14) :

```
R1#sh ip route vrf customer-1

Routing Table: customer-1
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

      192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.10.0/24 is directly connected, GigabitEthernet1
L        192.168.10.1/32 is directly connected, GigabitEthernet1
B     192.168.20.0/24 [200/0] via 10.3.3.3, 00:58:50
R1#
```

```
R3#sh ip route vrf customer-1

Routing Table: customer-1
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is not set

B     192.168.10.0/24 [200/0] via 10.1.1.1, 00:11:19
      192.168.20.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.20.0/24 is directly connected, GigabitEthernet2
L        192.168.20.1/32 is directly connected, GigabitEthernet2
R3#
```

Figure.III.14 : Table de routage VRF du routeur R1, R3

En comparaison, si vous effectuez un ping dans les routeurs LER (R1 et R3) en utilisant les adresses des interfaces (gateway) qui correspond aux sites clients sans spécifier la VRF dans un réseau MPLS backbone, le ping sera traité dans le contexte de la table de routage globale du routeur qui contient les informations de routage pour du réseau MPLS, ce qui va échouer comme il indiqué ci-dessous (Figure.III.15), par contre en spécifiant la VRF le ping passe.

```
R3#ping 192.168.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R3#ping vrf customer-1 192.168.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
```

Figure.III.15 : Test ping sans et avec VRF

Pour vérifier la connectivité entre les deux sites clients en traversant le nuage MPLS nous avons utilisé un ping ICMP à partir de l'une des deux machines :

```
host2@host2-virtual-machine:~$ ping 192.168.10.2 -c 10
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=61 time=1.82 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=61 time=1.85 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=61 time=2.05 ms
64 bytes from 192.168.10.2: icmp_seq=4 ttl=61 time=1.88 ms
64 bytes from 192.168.10.2: icmp_seq=5 ttl=61 time=2.17 ms
64 bytes from 192.168.10.2: icmp_seq=6 ttl=61 time=2.62 ms
64 bytes from 192.168.10.2: icmp_seq=7 ttl=61 time=10.3 ms
64 bytes from 192.168.10.2: icmp_seq=8 ttl=61 time=1.76 ms
64 bytes from 192.168.10.2: icmp_seq=9 ttl=61 time=1.62 ms
64 bytes from 192.168.10.2: icmp_seq=10 ttl=61 time=2.06 ms

--- 192.168.10.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9017ms
rtt min/avg/max/mdev = 1.621/2.824/10.369/2.529 ms
host2@host2-virtual-machine:~$
```

Figure.III.16 : Ping entre VM1 et VM2

III.3.2 Topologie OpenFlow

Nous avons mis en place une topologie dans Mininet sous une machine virtuelle Ubuntu 16.04 TLS comprenant 3 switchs OVS et deux hôtes. Le contrôleur Floodlight dans une autre machine virtuelle est utilisé pour gérer cette topologie comme il est indiqué dans la Figure.III.17.

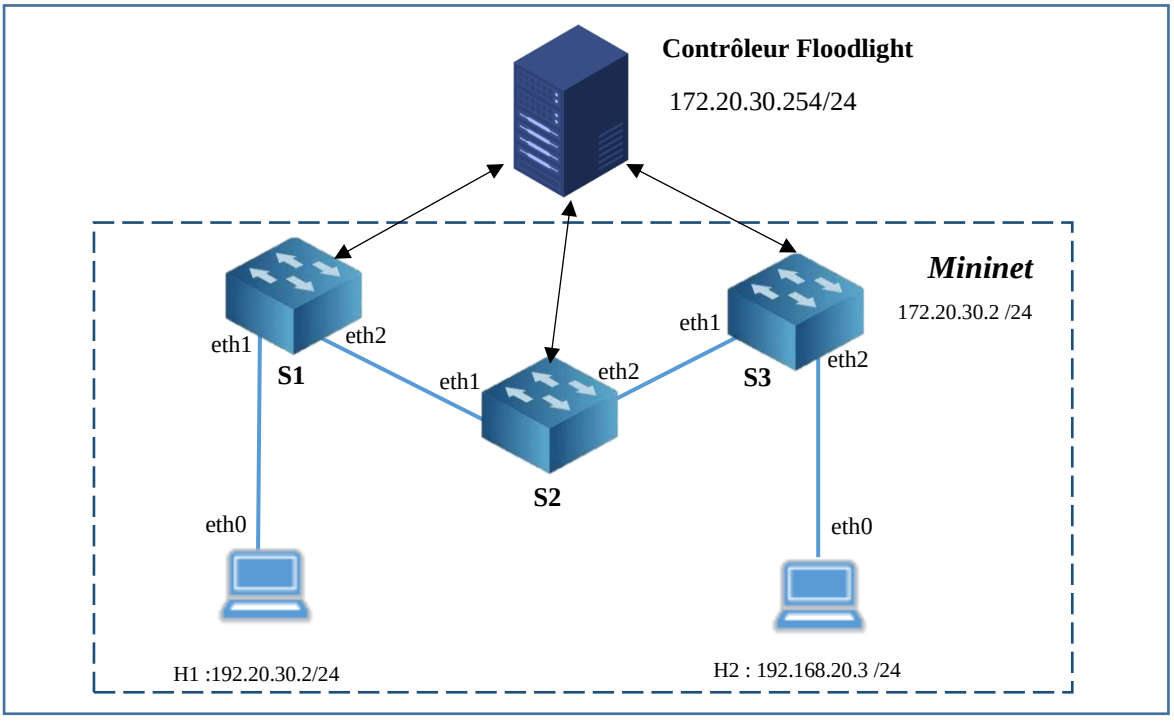


Figure.III.17 : Diagramme de la topologie OpenFlow

III.3.2.1 Configuration des différents composant Mininet et contrôleur

La topologie a été définie à l'aide d'un script Python personnalisé et sauvegardée dans le répertoire « *mininet/custom* ».

La commande *sudo nano matopo.py* nous permettra d'accéder à ce fichier et d'écrire le script Python (Figure.III.18).

```
  GNU nano 2.5.3                        File: matopo.py

from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.node import RemoteController, OVSSwitch

class MinimalTopo( Topo ):
    "Minimal topology with a single switch and two hosts"

    def build( self ):
        # Create two hosts.
        h1 = self.addHost( 'h1',ip='192.168.20.2/24')
        h2 = self.addHost( 'h2',ip='192.168.20.3/24')

       # Create a switch
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        s3 = self.addSwitch( 's3' )

        # Add links between the switch and each host
        self.addLink( s1, h1 )
        self.addLink( s1, s2 )
        self.addLink( s2, s3 )
        self.addLink( s3, h2 )

def runMinimalTopo():
    "Bootstrap a Mininet network using the Minimal Topology"

    # Create an instance of our topology
    topo = MinimalTopo()
```

```
# Create a network based on the topology using OVS and controlled by
# a remote controller.
net = Mininet(
    topo=topo,
    controller=lambda name: RemoteController( name, ip='172.20.30.252:6653' ),
    switch=OVSSwitch,
    autoSetMacs=True )

# Actually start the network
net.start()

# Drop the user in to a CLI so user can run commands.
CLI( net )

# After the user exits the CLI, shutdown the network.
net.stop()
```

```
    # After the user exits the CLI, shutdown the network.
    net.stop()

if __name__ == '__main__':
    # This runs if this file is executed directly

    setLogLevel( 'info' )
    runMinimalTopo()

# Allows the file to be imported using `mn --custom <filename> --topo minimal`
topos = {
    'minimal': MinimalTopo
}
```

Figure.III.18 : Script python de la topologie

Avant de lancer la topologie, vous devez démarrer le contrôleur Floodlight en accédant au terminal et en exécutant les commandes suivantes :

cd ~/floodlight

ant

java -jar target/floodlight.jar

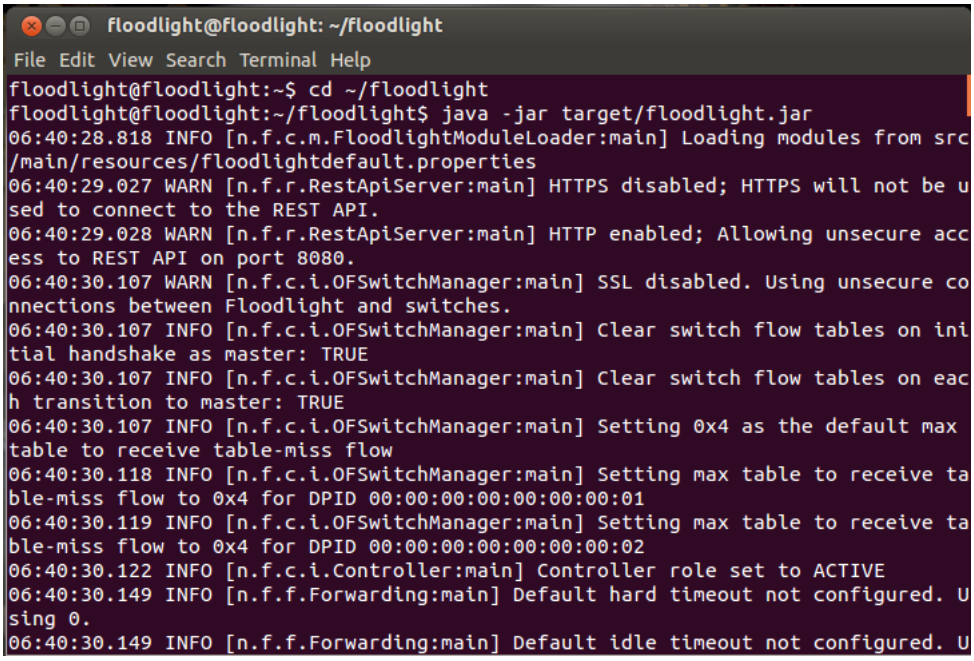


Figure.III.19 : Démarrage du contrôleur Floodlight

En exécutant le script mentionné précédemment, la topologie physique de notre réseau sera créée avec les différents éléments tels que les switchs OpenFlow et les hôtes. Cette topologie sera ensuite connectée au contrôleur Floodlight à distance en utilisant la commande suivante : ***sudo mn --custom nom_script --topo nom_topologie --controller=remote ,ip=IPyt_contrôleur, port=6653 --switch ovs, protocols=OpenFlow13***

Une fois le script lancé, le contrôleur Floodlight détectera automatiquement les switchs OpenFlow et les hôtes qui ont été créés dans la topologie. Ceci est illustré dans la Figure.III.20 :

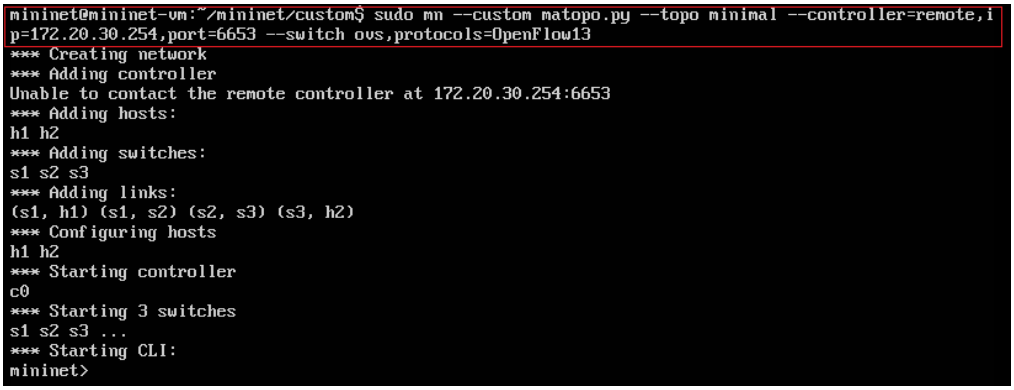


Figure.III.20 : Exécution de la topologie sur Mininet

Après avoir effectué un ping global pour tester la connectivité entre les hôtes de notre réseau, voici le résultat obtenu :

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

Figure.III.21 : Test de connectivite

Une fois la topologie créée, tous les détails seront affichés dans la page d'accueil de Floodlight (Figure.III.22 et Figure.III.23) en accédant par un navigateur à l'adresse serveur par cette url : **http://172.20.30.254:8080/ui/index.html**

Switches (3)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:00:00:00:00:00:02	/172.20.30.2:59594	Nicira, Inc.	12	750	5	15/05/2023 12:38:57
00:00:00:00:00:00:00:03	/172.20.30.2:59598	Nicira, Inc.	8	487	5	15/05/2023 12:38:57
00:00:00:00:00:00:00:01	/172.20.30.2:59596	Nicira, Inc.	8	487	5	15/05/2023 12:38:57

Hosts (2)

MAC Address	IP Address	Switch Port	Last Seen
7a:55:c2:6e:96:66	0.0.0.0,192.168.20.3	00:00:00:00:00:00:00:03-2	15/05/2023 12:39:40
de:97:41:b8:ed:87	192.168.20.2	00:00:00:00:00:00:00:01-1	15/05/2023 12:39:45

Figure.III.22 : Détails des composants de topologie

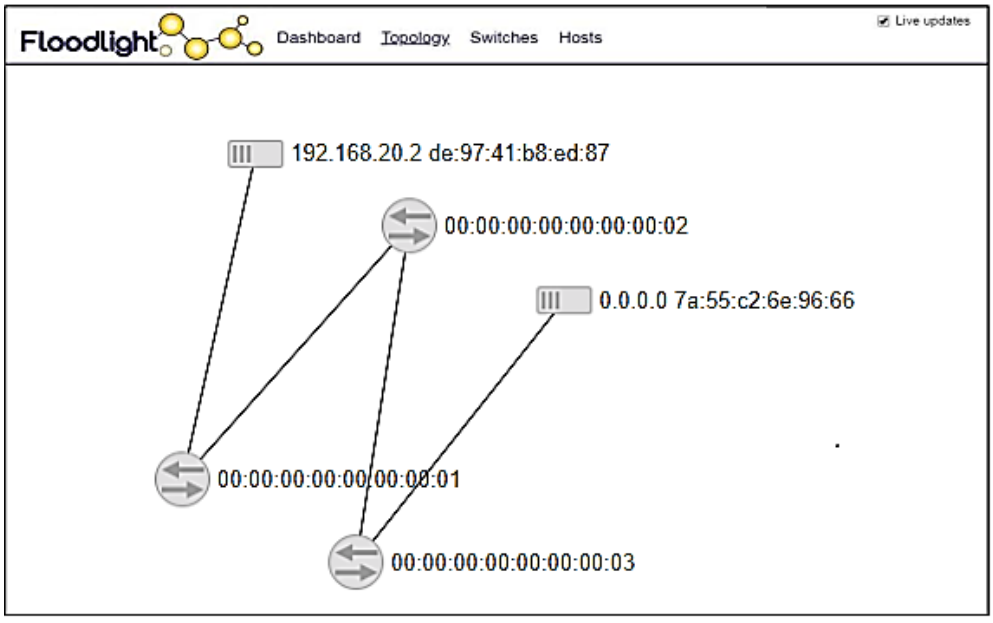


Figure.III.23 : Topologie affichée sur Floodlight

III.4 Tests effectués

Dans cette partie et suite des scénarios indiqués précédemment nous avons effectué une série d'expériences et des tests pour faire une comparaison entre les deux solutions MPLS et OpenFlow SDN. Ces derniers s'articulent sur différents aspects à savoir :

- Forwarding du trafic ;
- Scalabilité ;
- Latence RTT ;
- Débit ;
- Jitter.

III.4.1 Forwarding du trafic

Dans cette partie nous allons expérimenter pour chaque protocole la manière du forwarding du trafic et précisément les plans de contrôle et de données.

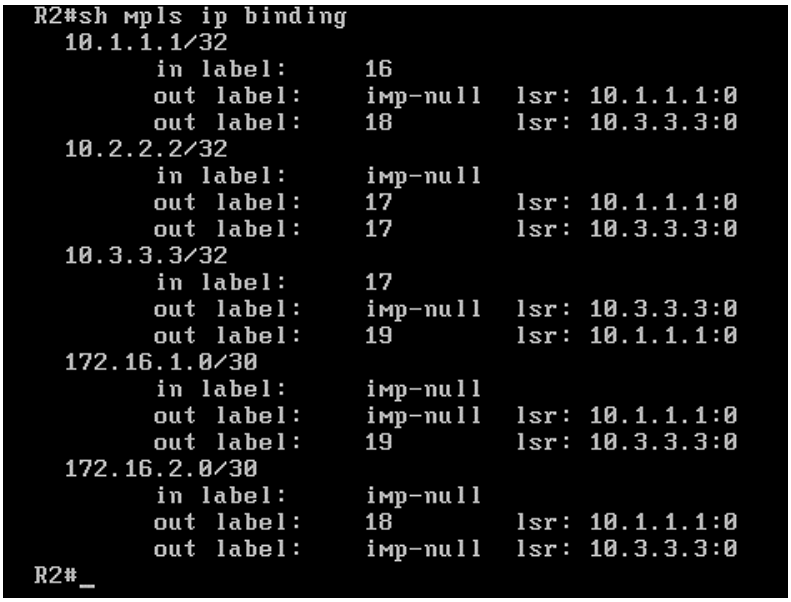
III.4.1.1 MPLS

Comme il est indiqué dans le 1er chapitre la technologie MPLS est basé sur l'allocation des labels leurs distribution et leur commutation. Le protocole utilisé dans notre solution pour la distribution des labels est le LDP.

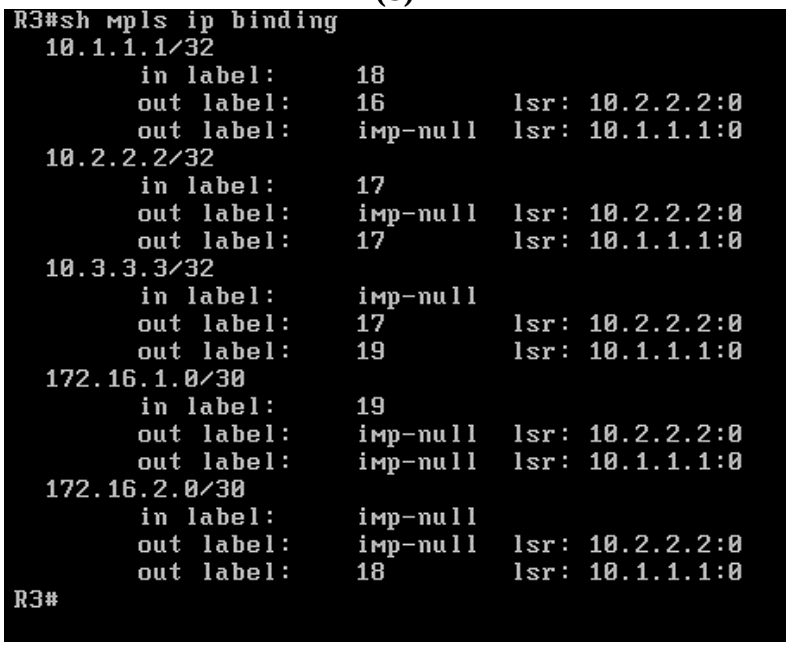
Les labels sont alloués au fur et à mesure de l'apparition des routes dans les routeurs qui dépend des annonces de l'IGP dans notre cas l'OSPF.
Comme il indiqué ci-dessous chaque route dans la table de routage (Routing Information Base (RIB)) un label est attribué localement. Et par conséquent dans chaque routeur MPLS une table LIB sera créé (Control plan) comme il est indiqué dans la Figure.III.24.

```
R1#sh mpls ip binding
  10.1.1.1/32
        in label:     imp-null
        out label:    16         lsr: 10.2.2.2:0
        out label:    18         lsr: 10.3.3.3:0
  10.2.2.2/32
        in label:     17
        out label:    imp-null   lsr: 10.2.2.2:0
        out label:    17         lsr: 10.3.3.3:0
  10.3.3.3/32
        in label:     19
        out label:    imp-null   lsr: 10.3.3.3:0
        out label:    17         lsr: 10.2.2.2:0
  172.16.1.0/30
        in label:     imp-null
        out label:    imp-null   lsr: 10.2.2.2:0
        out label:    19         lsr: 10.3.3.3:0
  172.16.2.0/30
        in label:     18
        out label:    imp-null   lsr: 10.2.2.2:0
        out label:    imp-null   lsr: 10.3.3.3:0
R1#
```

(a)



(b)



(c)

Figure.III.24 : **(a), (b), (c)** : Table LIB des routeurs R1, R2, et R3

Maintenant que chaque routeur a alloué des labels à chaque FEC (chaque route), il leur reste à les distribuer aux voisins.

Par exemple si nous prenons table LIB du R1 pour la destination 10.3.3.3 nous trouvons :

- Attribué localement le label 19 *in label*
- Reçu de R2 le label 17 *out label*
- Reçu de R3 le label *imp_null outlabel* (localement dans R3)

Et à partir de la table LIB (Label Information Base) qui contient les informations du plan de contrôle et chaque routeur établit la table LFIB (Label Forwarding Information Base) utilisée dans le plan de données.

```
R1#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.2.2.2/32      0             Gi2        172.16.1.1
17         17         10.3.3.3/32      0             Gi2        172.16.1.1
18         Pop Label  172.16.2.0/30    0             Gi2        172.16.1.1
19         No Label   192.168.10.0/24[V]   \
                                       2940          aggregate/customer-1
```

```
R2#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.1.1.1/32      67514         Gi1        172.16.1.2
17         Pop Label  10.3.3.3/32      18904         Gi2        172.16.2.2
```

```
R3#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         16         10.1.1.1/32      0             Gi1        172.16.2.1
17         Pop Label  10.2.2.2/32      0             Gi1        172.16.2.1
18         Pop Label  172.16.1.0/30    0             Gi1        172.16.2.1
20         No Label   192.168.20.0/24[V]   \
                                       3608          aggregate/customer-1
```

Figure.III.25 : Table LFIB de R1, R2 et R3

En suivant le même exemple pour atteindre la destination 10.3.3.3 à partir de R1 :

- R1 envoie le paquet MPLS avec l'étiquette 17 vers R2 (Push) ;
- R2 compare dans sa table LFIB la correspondance de in label 17 ensuite il l'envoie vers R3 (Pop).

Et ceci c'est le principe de ***label switching***.

A noter que dans la table LFIB on trouve que les destinations du nuage MPLS, et dans notre cas les paquets arrivant des VRFs du R1 et R3 sont transformé en paquets MPLS en utilisant l'adressage des routeurs MPLS et les labels selon la table LFIB.

Si par exemple nous lançons un ping à partir de VM1 vers VM2 : ***ping 192.168.20.2***

Le paquet sera encapsulé et envoyé vers le R1 et suite à la table de routage du VRF la destination sera atteindre via le BGP, et puis il est encapsulé dans un paquet MPLS avec label 17 selon la table LFIB vers R2 et ensuite envoyé sans label vers le R3 et puis vers VM2. Et le retour du ping ICMP de VM2 vers VM1 va être le même principe.

III.4.1.2 OpenFlow

Comme il est indiqué dans le 2éme chapitre la solution OpenFlow SDN s'articule sur la séparation entre le plan de contrôle et le plan des données dans un autre terme les décisions d'acheminement du trafic se décide au niveau du contrôleur et suite à ces décisions des tables de flux seront construites aux switch OVS. Dans cette partie nous allons voir en détail le mécanisme de forwarding appliquées dans ce protocole.

La première étape dans le contrôleur c'est la découverte à l'aide du protocole LLDP et le chargement de la topologie comme il indiqué dans les captures Wireshark (Figure.III.26).

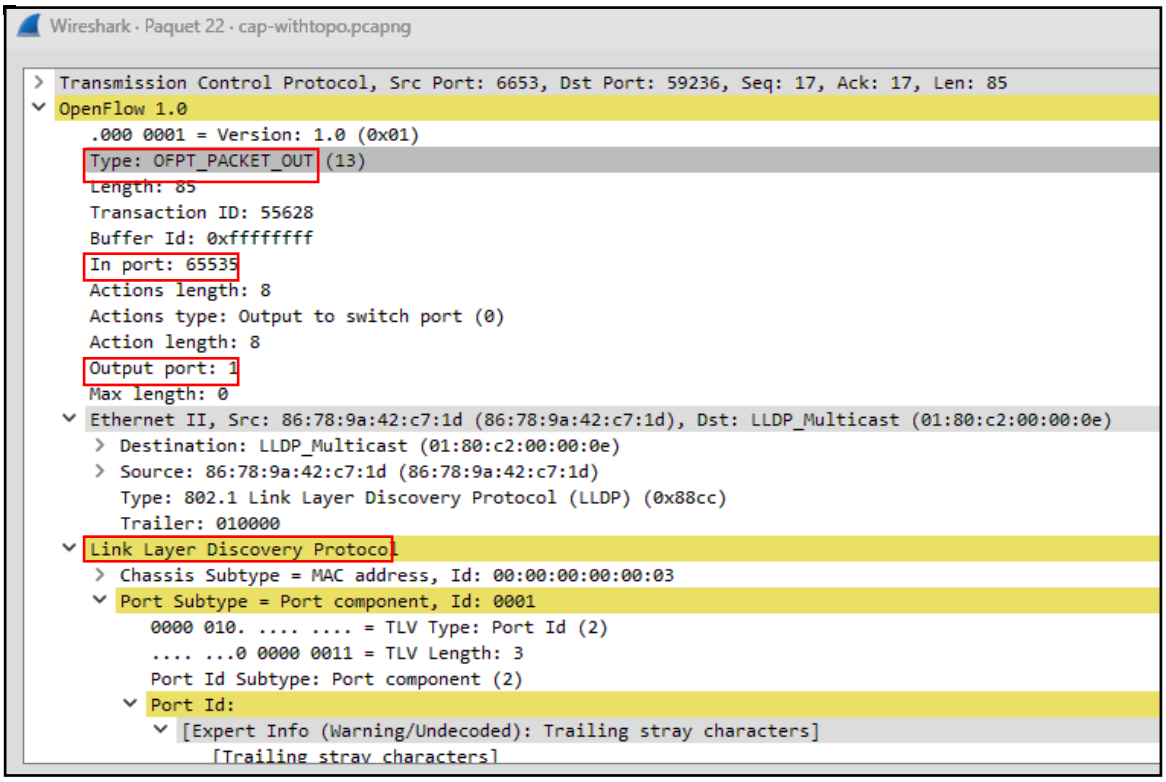


Figure.III.26 : Message de découvert

A ce moment les tables de flux dans chaque switch OVS sont vides (Figure.III.27, Figure.III.29).

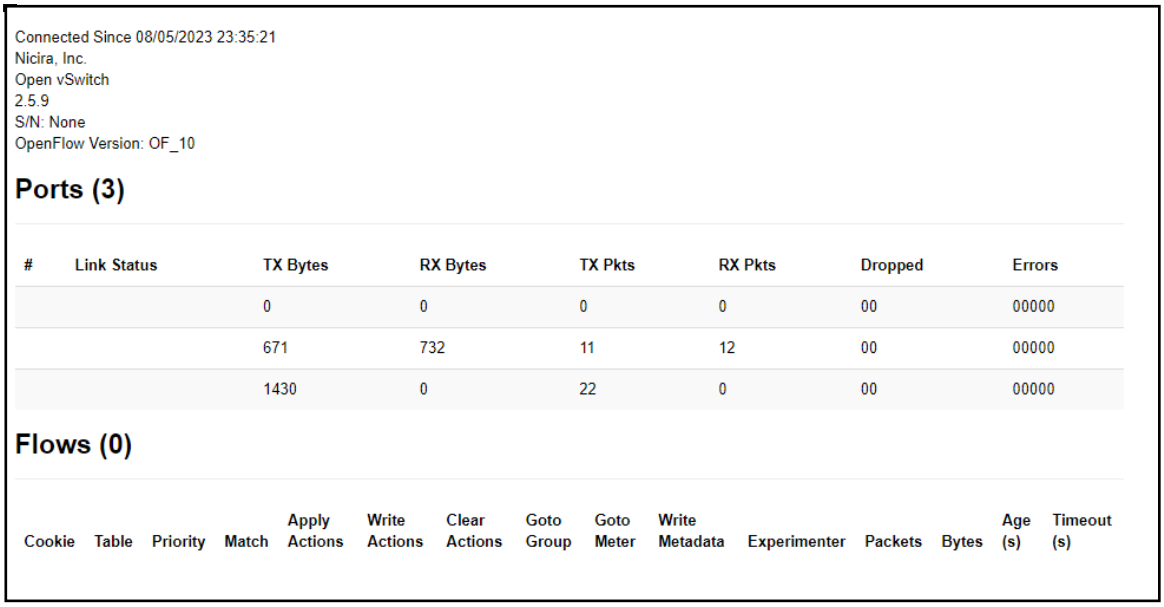


Figure.III.27 : Table de flux S1 avant le Ping

Une fois un switch reçoit un trafic dans l'un de ces ports à ce moment le switch OVS envoie des requêtes au contrôleur (Figure.III.31) pour la constitution de table de flux (Figure.III.28, Figure.III.30).

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
local (s1)	DOWN	0	0	0	0	10	00000
1 (s1-eth1)	UP 10 Gbps FDX	20224	14504	240	152	00	00000
2 (s1-eth2)	UP 10 Gbps FDX	17318	17326	198	198	00	00000

Flows (7)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)
9007199254740992	0x0	1	in_port=1 eth_dst=a6:1e:78:a6:19:7c eth_src=32:66:3c:cb:71:ff eth_type=0x0x800 ipv4_src=192.168.20.2 ipv4_dst=192.168.20.3	actions:output=2	---	---	---	---	---	---	109	10682	109
9007199254740992	0x0	1	in_port=2 eth_dst=32:66:3c:cb:71:ff eth_src=a6:1e:78:a6:19:7c eth_type=0x0x800 ipv4_src=192.168.20.3 ipv4_dst=192.168.20.2	actions:output=1	---	---	---	---	---	---	110	10780	109

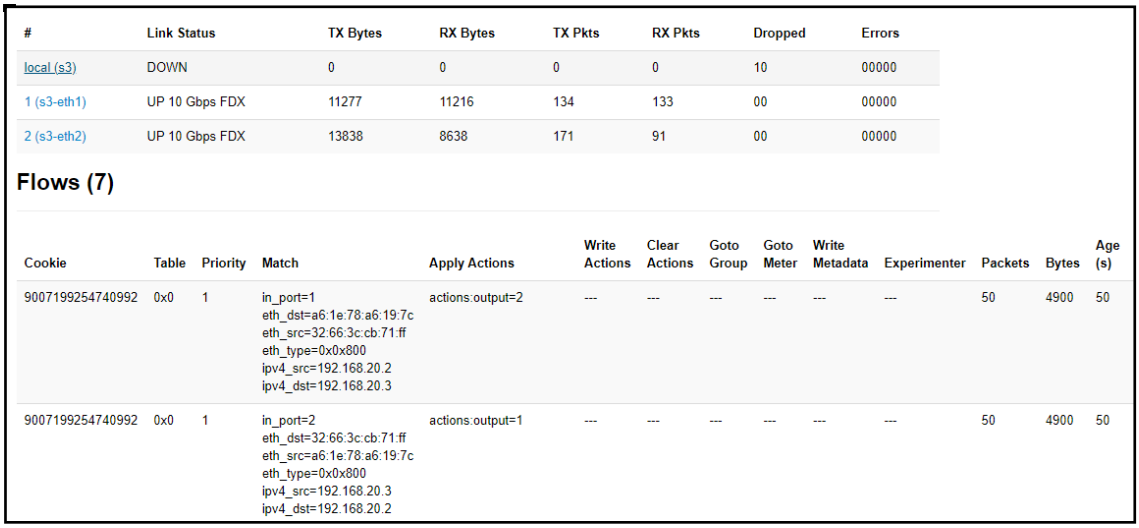
(a)

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
local (s2)	DOWN	0	0	0	0	10	00000
1 (s2-eth1)	UP 10 Gbps FDX	4067	4059	54	54	00	00000
2 (s2-eth2)	UP 10 Gbps FDX	4067	4128	54	55	00	00000

Flows (7)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)
9007199254740992	0x0	1	in_port=1 eth_dst=a6:1e:78:a6:19:7c eth_src=32:66:3c:cb:71:ff eth_type=0x0x800 ipv4_src=192.168.20.2 ipv4_dst=192.168.20.3	actions:output=2	---	---	---	---	---	---	19	1862	19
9007199254740992	0x0	1	in_port=2 eth_dst=32:66:3c:cb:71:ff eth_src=a6:1e:78:a6:19:7c eth_type=0x0x800 ipv4_src=192.168.20.3 ipv4_dst=192.168.20.2	actions:output=1	---	---	---	---	---	---	20	1960	19

(b)



#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
local (s3)	DOWN	0	0	0	0	10	00000
1 (s3-eth1)	UP 10 Gbps FDX	11277	11216	134	133	00	00000
2 (s3-eth2)	UP 10 Gbps FDX	13838	8638	171	91	00	00000

Flows (7)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)
9007199254740992	0x0	1	in_port=1 eth_dst=a6:1e:78:a6:19:7c eth_src=32:66:3c:cb:71:ff eth_type=0x0x800 ipv4_src=192.168.20.2 ipv4_dst=192.168.20.3	actions:output=2	---	---	---	---	---	---	50	4900	50
9007199254740992	0x0	1	in_port=2 eth_dst=32:66:3c:cb:71:ff eth_src=a6:1e:78:a6:19:7c eth_type=0x0x800 ipv4_src=192.168.20.3 ipv4_dst=192.168.20.2	actions:output=1	---	---	---	---	---	---	50	4900	50

(c)

Figure.III.28 (a), (b), (c) : Table de flux des switchs après Ping

Et si par exemple nous lançons un ping ICMP à partir du H1 vers H2 et vue que le contrôleur a une vue de la topologie il envoie des règles de flux pour chaque switch concerné pour l'acheminement de ce paquet.

Chaque règle contient :

- Le port In (l'interface)
- L'adresse IP source du paquet
- L'adresse IP destination du paquet
- L'adresse MAC source du paquet
- L'adresse MAC destination du paquet
- Action Output port (Interface)

Et à partir de Mininet en peut aussi visualiser les tables de flux dans chaque switch avec la commande ***sh ovs-ofctl dump-flows s1 –protocols=OpenFlow13***

```
mininet> sh ovs-ofctl dump-flows s1 --protocols=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=4.309s, table=0, n_packets=1, n_bytes=61, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=4.309s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=4.309s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=4.309s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=4.309s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-ofctl dump-flows s2 --protocols=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=15.161s, table=0, n_packets=4, n_bytes=244, priority=0 actions=CONTROLLER:6553
5
 cookie=0x0, duration=15.161s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=15.161s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=15.161s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=15.160s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-ofctl dump-flows s3 --protocols=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=21.123s, table=0, n_packets=2, n_bytes=122, priority=0 actions=CONTROLLER:6553
5
 cookie=0x0, duration=21.123s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=21.123s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=21.123s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=21.123s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
```

Figure.III.29 : Table de flux vide des 3 switchs à partir de Mininet

```
mininet> sh ovs-ofctl dump-flows s1 --protocols=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x20000000000000, duration=103.709s, table=0, n_packets=100, n_bytes=9800, idle_timeout=5, p
riority=1,ip,in_port=1,dl_src=46:49:fb:ec:e1:96,dl_dst=52:77:3d:73:25:ce,nw_src=192.168.20.2,nw_dst=
192.168.20.3 actions=output:2
 cookie=0x20000000000000, duration=103.705s, table=0, n_packets=101, n_bytes=9898, idle_timeout=5, p
riority=1,ip,in_port=2,dl_src=52:77:3d:73:25:ce,dl_dst=46:49:fb:ec:e1:96,nw_src=192.168.20.3,nw_dst=
192.168.20.2 actions=output:1
 cookie=0x0, duration=371.009s, table=0, n_packets=38, n_bytes=2339, priority=0 actions=CONTROLLER:6
5535
 cookie=0x0, duration=371.009s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=371.009s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=371.009s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=371.009s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-ofctl dump-flows s2 --protocols=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x20000000000000, duration=34.511s, table=0, n_packets=31, n_bytes=3038, idle_timeout=5, pri
ority=1,ip,in_port=1,dl_src=36:d2:52:8c:e1:10,dl_dst=9a:51:ce:ea:d8:1e,nw_src=192.168.20.2,nw_dst=19
2.168.20.3 actions=output:2
 cookie=0x20000000000000, duration=34.503s, table=0, n_packets=30, n_bytes=2940, idle_timeout=5, pri
ority=1,ip,in_port=2,dl_src=9a:51:ce:ea:d8:1e,dl_dst=36:d2:52:8c:e1:10,nw_src=192.168.20.3,nw_dst=19
2.168.20.2 actions=output:1
 cookie=0x0, duration=177.803s, table=0, n_packets=32, n_bytes=1966, priority=0 actions=CONTROLLER:6
5535
 cookie=0x0, duration=177.803s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=177.803s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=177.803s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=177.803s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-ofctl dump-flows s3 --protocols=OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x20000000000000, duration=31.439s, table=0, n_packets=29, n_bytes=2842, idle_timeout=5, pri
ority=1,ip,in_port=2,dl_src=36:d2:52:8c:e1:10,dl_dst=9a:51:ce:ea:d8:1e,nw_src=192.168.20.2,nw_dst=19
2.168.20.3 actions=output:1
 cookie=0x20000000000000, duration=31.433s, table=0, n_packets=28, n_bytes=2744, idle_timeout=5, pri
ority=1,ip,in_port=1,dl_src=9a:51:ce:ea:d8:1e,dl_dst=36:d2:52:8c:e1:10,nw_src=192.168.20.3,nw_dst=19
2.168.20.2 actions=output:2
 cookie=0x0, duration=310.692s, table=0, n_packets=33, n_bytes=2092, priority=0 actions=CONTROLLER:6
5535
 cookie=0x0, duration=310.692s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=310.692s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=310.692s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=310.692s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
```

Figure.III.30 : Table de flux des 3 switchs après ping

Et à partir de ces règles reçus du contrôleur les tables de flux seront construites au niveau des switchs.

Message de switch vers contrôleur pour demander la table de flux (Figure.III.31) :

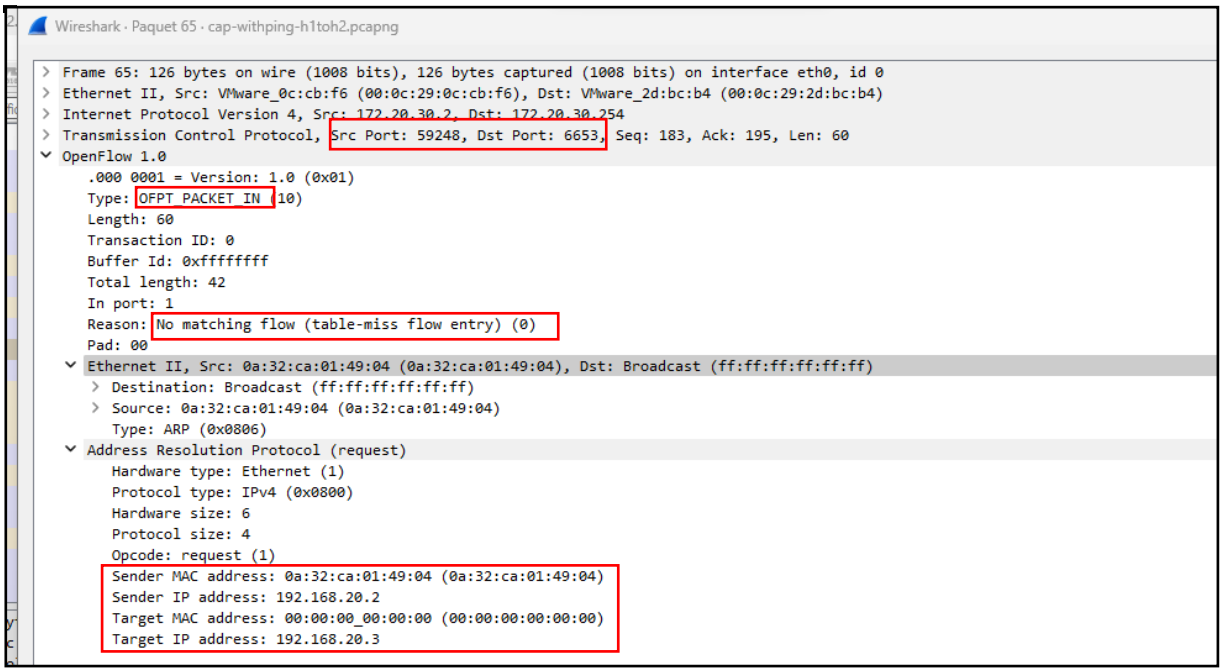


Figure.III.31 : Demande des règles de flux du switch vers contrôleur

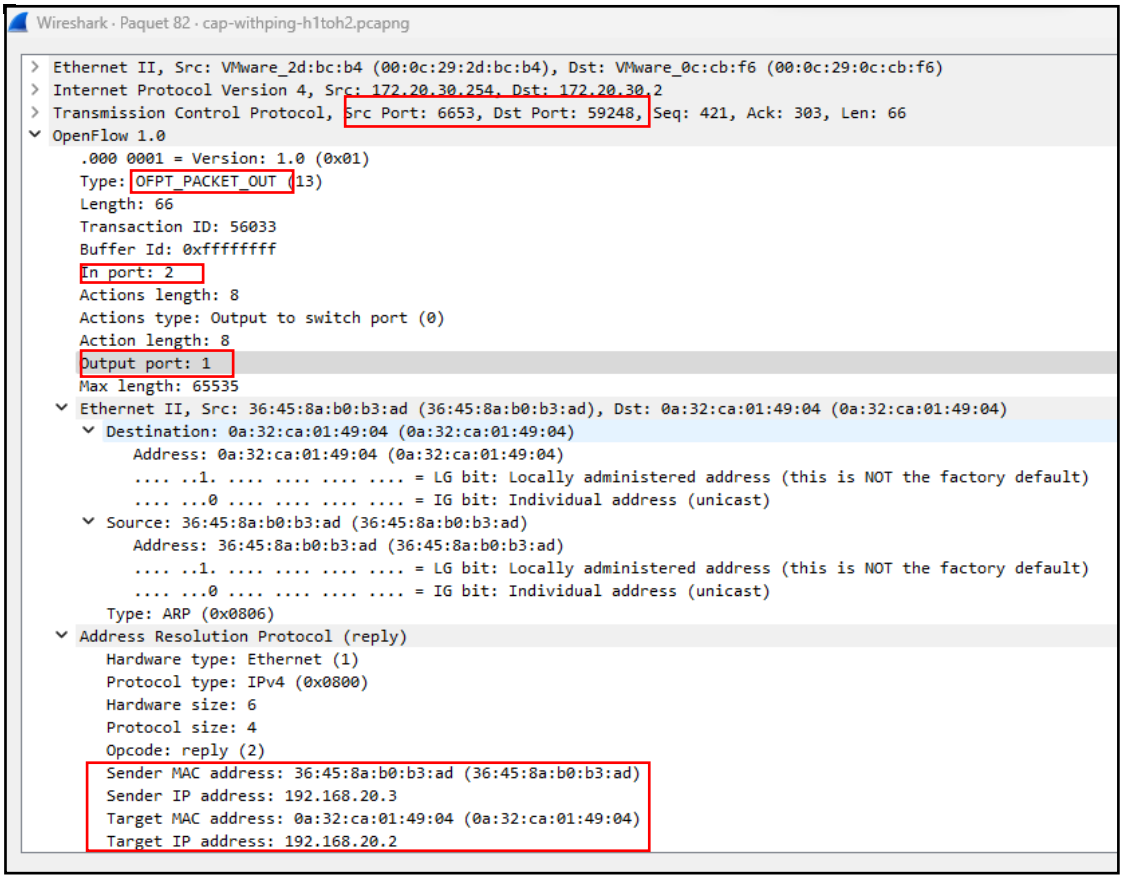


Figure.III.32 : Message de réponse vers le switch OVS

Et à cette manière les tables de flux seront construites dans les switchs OpenFlow.

Et comme il indiqué dans la Figure.II.8 du 2émé chapitre chaque paquet arrivant un port d'un switch, ce dernier va comparer est ce qu'il existe dans sa table la règle correspondante (In port, adresse source, adresse destination, out port) si oui il va encapsuler le paquet en L2 et l'envoi via le port output. Si non il va envoyer une requête vers le contrôleur pour mettre à jour sa table de flux.
Et par ce mécanisme se fait le forwarding des paquets dans OpenFlow.

III.4.2 Scalabilité

Dans cette partie nous allons aborder le sujet de la scalabilité (évolutivité du réseau) en en déployant un nouveau routeur R4 dans la partie core du réseau MPLS ainsi qu'un nouveau switch OVS S4 dans la partie core OpenFlow et voir les étapes d'intégration dans chaque environnement.

III.4.2.1 MPLS

Pour ajouter un routeur R4 (Figure.III.33) entre R1 et R3, il faut le configurer comme suit :

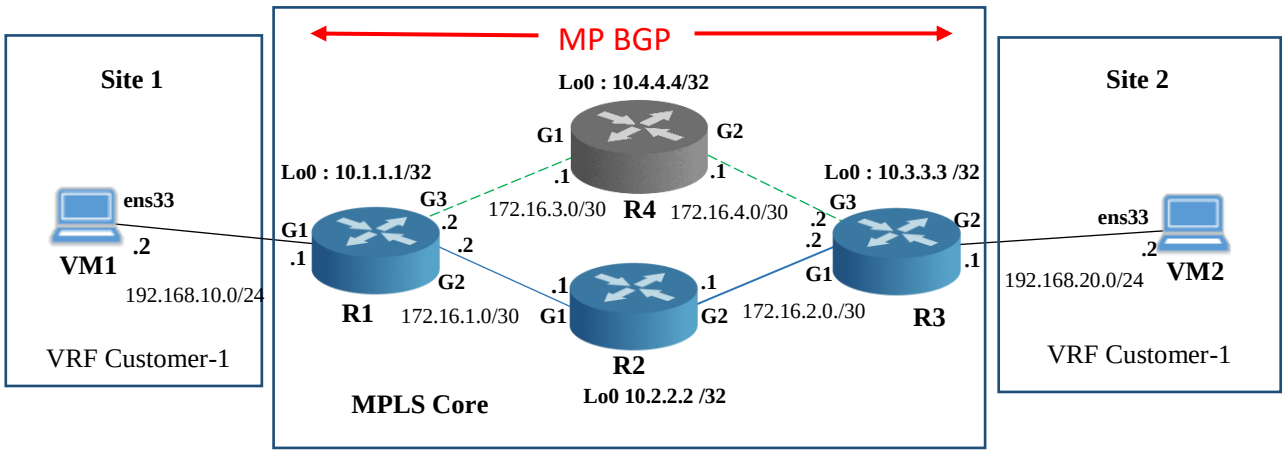


Figure.III.33 : Topologie MPLS scalabilité

Dans un réseau MPLS pour faire l'extension d'un nouveau nœud nous devons procéder comme suit :

- Au niveau du nouveau routeur R4 :
  - Configurer une instance OSPF avec l'activation des LDP MPLS ;
  - Configurez les interfaces du nouveau Routeur avec leurs adresses (physiques et Loopback) et les associer à l'OSPF pour l'échange des route ;
  - Pour vérifier les interfaces au niveau du routeur R4 nous utilisons cette commande : ***show running interface GigabitEthernet*** *x*

```
R4#sh run int g1
Building configuration...

Current configuration : 118 bytes
!
interface GigabitEthernet1
 ip address 172.16.3.1 255.255.255.252
 ip ospf 1 area 0
 negotiation auto
 mpls ip
end

R4#sh run int g2
Building configuration...

Current configuration : 118 bytes
!
interface GigabitEthernet2
 ip address 172.16.4.1 255.255.255.252
 ip ospf 1 area 0
 negotiation auto
 mpls ip
end

R4#_
```

Figure.III.34 : Interfaces GigabitEthernet du R4

- Au niveau des R1 et R3

Configurez l'adressage d'une 3éme interfaces du R1 et R3 et l'associer à l'OSPF pour l'échange des routes avec l'activation du MPLS.

```
R1#sh run int g3
Building configuration...

Current configuration : 118 bytes
!
interface GigabitEthernet3
 ip address 172.16.3.2 255.255.255.252
 ip ospf 1 area 0
 negotiation auto
 mpls ip
end

R1#
```

```
R3#sh run int g3
Building configuration...

Current configuration : 118 bytes
!
interface GigabitEthernet3
 ip address 172.16.4.2 255.255.255.252
 ip ospf 1 area 0
 negotiation auto
 mpls ip
end

R3#
```

Figure.III.35 : Interfaces g3 du R1 et R3

N.B : Les commandes de configuration de cette partie sont exposées dans l'annexe **A**.

Et sur suite à ces configurations il y aura une mise à jour dans les tables (RIB, LIB, FLIB) des quatre routeurs du core MPLS comme il indiqué dans les Figure.III.36, Figure.III.37 et Figure.III.38.

```
      10.0.0.0/32 is subnetted, 4 subnets
C        10.1.1.1 is directly connected, Loopback0
O        10.2.2.2 [110/2] via 172.16.1.1, 00:13:08, GigabitEthernet2
O        10.3.3.3 [110/3] via 172.16.3.1, 00:08:28, GigabitEthernet3
                  [110/3] via 172.16.1.1, 00:12:58, GigabitEthernet2
O        10.4.4.4 [110/2] via 172.16.3.1, 00:11:32, GigabitEthernet3
      172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C        172.16.1.0/30 is directly connected, GigabitEthernet2
L        172.16.1.2/32 is directly connected, GigabitEthernet2
O        172.16.2.0/30 [110/2] via 172.16.1.1, 00:13:08, GigabitEthernet2
C        172.16.3.0/30 is directly connected, GigabitEthernet3
L        172.16.3.2/32 is directly connected, GigabitEthernet3
O        172.16.4.0/30 [110/2] via 172.16.3.1, 00:11:32, GigabitEthernet3
R1#
```

```
      10.0.0.0/32 is subnetted, 4 subnets
O        10.1.1.1 [110/2] via 172.16.1.2, 00:17:47, GigabitEthernet1
C        10.2.2.2 is directly connected, Loopback0
O        10.3.3.3 [110/2] via 172.16.2.2, 00:17:07, GigabitEthernet2
O        10.4.4.4 [110/3] via 172.16.2.2, 00:10:20, GigabitEthernet2
                  [110/3] via 172.16.1.2, 00:13:24, GigabitEthernet1
      172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
C        172.16.1.0/30 is directly connected, GigabitEthernet1
L        172.16.1.1/32 is directly connected, GigabitEthernet1
C        172.16.2.0/30 is directly connected, GigabitEthernet2
L        172.16.2.1/32 is directly connected, GigabitEthernet2
O        172.16.3.0/30 [110/2] via 172.16.1.2, 00:13:24, GigabitEthernet1
O        172.16.4.0/30 [110/2] via 172.16.2.2, 00:10:20, GigabitEthernet2
R2#
```

```
      10.0.0.0/32 is subnetted, 4 subnets
O        10.1.1.1 [110/3] via 172.16.4.1, 00:11:07, GigabitEthernet3
                  [110/3] via 172.16.2.1, 00:15:37, GigabitEthernet1
O        10.2.2.2 [110/2] via 172.16.2.1, 00:15:47, GigabitEthernet1
C        10.3.3.3 is directly connected, Loopback0
O        10.4.4.4 [110/2] via 172.16.4.1, 00:11:07, GigabitEthernet3
      172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O        172.16.1.0/30 [110/2] via 172.16.2.1, 00:15:47, GigabitEthernet1
C        172.16.2.0/30 is directly connected, GigabitEthernet1
L        172.16.2.2/32 is directly connected, GigabitEthernet1
O        172.16.3.0/30 [110/2] via 172.16.4.1, 00:11:07, GigabitEthernet3
C        172.16.4.0/30 is directly connected, GigabitEthernet3
L        172.16.4.2/32 is directly connected, GigabitEthernet3
R3#
```

```
      10.0.0.0/32 is subnetted, 4 subnets
O        10.1.1.1 [110/2] via 172.16.3.2, 00:15:10, GigabitEthernet1
O        10.2.2.2 [110/3] via 172.16.4.2, 00:12:04, GigabitEthernet2
                  [110/3] via 172.16.3.2, 00:15:10, GigabitEthernet1
O        10.3.3.3 [110/2] via 172.16.4.2, 00:12:04, GigabitEthernet2
C        10.4.4.4 is directly connected, Loopback0
      172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O        172.16.1.0/30 [110/2] via 172.16.3.2, 00:15:10, GigabitEthernet1
O        172.16.2.0/30 [110/2] via 172.16.4.2, 00:12:04, GigabitEthernet2
C        172.16.3.0/30 is directly connected, GigabitEthernet1
L        172.16.3.1/32 is directly connected, GigabitEthernet1
C        172.16.4.0/30 is directly connected, GigabitEthernet2
L        172.16.4.1/32 is directly connected, GigabitEthernet2
R4#
```

Figure.III.36 : Table RIB des quatre routeurs

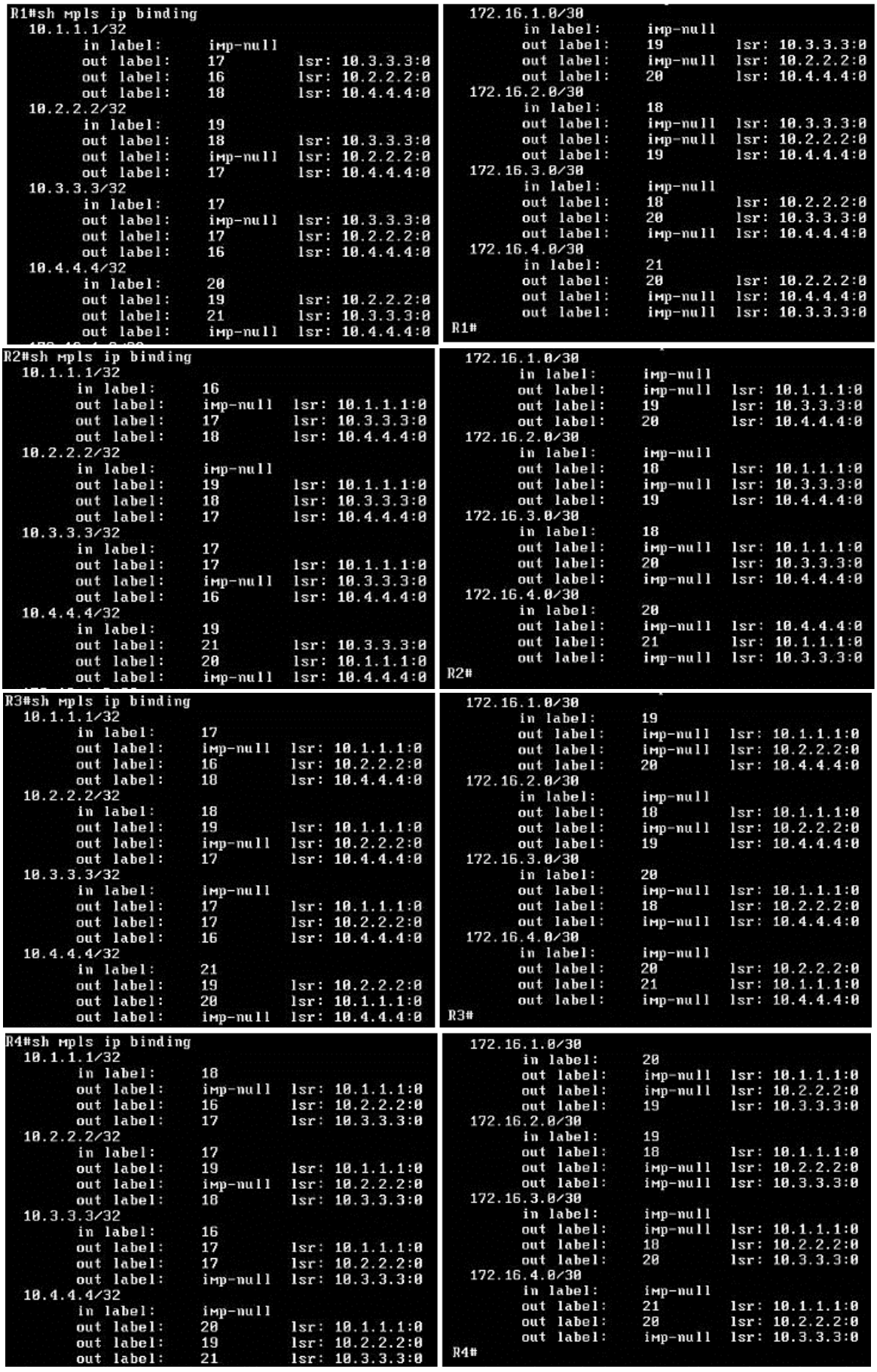


Figure.III.37 : Table LIB des quatre routeurs

```
R1#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         No Label   192.168.10.0/24[V]   \
                                       0             aggregate/customer-1
17         17         10.3.3.3/32      684           Gi2        172.16.1.1
           16         10.3.3.3/32      0             Gi3        172.16.3.1
18         Pop Label  172.16.2.0/30    0             Gi2        172.16.1.1
19         Pop Label  10.2.2.2/32      432           Gi2        172.16.1.1
20         Pop Label  10.4.4.4/32      5902          Gi3        172.16.3.1
21         Pop Label  172.16.4.0/30    0             Gi3        172.16.3.1
R1#_
```

```
R2#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.1.1.1/32      4466          Gi1        172.16.1.2
17         Pop Label  10.3.3.3/32      6343          Gi2        172.16.2.2
18         Pop Label  172.16.3.0/30    0             Gi1        172.16.1.2
19         20         10.4.4.4/32      1252          Gi1        172.16.1.2
           21         10.4.4.4/32      0             Gi2        172.16.2.2
20         Pop Label  172.16.4.0/30    0             Gi2        172.16.2.2
R2#
```

```
R3#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         No Label   192.168.20.0/24[V]   \
                                       0             aggregate/customer-1
17         16         10.1.1.1/32      0             Gi1        172.16.2.1
           18         10.1.1.1/32      0             Gi3        172.16.4.1
18         Pop Label  10.2.2.2/32      4032          Gi1        172.16.2.1
19         Pop Label  172.16.1.0/30    0             Gi1        172.16.2.1
20         Pop Label  172.16.3.0/30    0             Gi3        172.16.4.1
21         Pop Label  10.4.4.4/32      0             Gi3        172.16.4.1
R3#
```

```
R4#sh mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
16         Pop Label  10.3.3.3/32      8294          Gi2        172.16.4.2
17         19         10.2.2.2/32      0             Gi1        172.16.3.2
           18         10.2.2.2/32      0             Gi2        172.16.4.2
18         Pop Label  10.1.1.1/32      8222          Gi1        172.16.3.2
19         Pop Label  172.16.2.0/30    0             Gi2        172.16.4.2
20         Pop Label  172.16.1.0/30    0             Gi1        172.16.3.2
R4#_
```

Figure.III.38 : Table LFIB des quatre routeurs

Pour tester la connectivité le test de Ping entre les adresses loopback de R1, R2, R3 et R4 sera lancé (Figure.III.39).

```
R4#ping 10.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
R4#ping 10.3.3.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms
R4#ping 10.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 3/3/4 ms
R4#
```

Figure.III.39 : Test connectivité (R4 ping R2, R3 et R1)

Nous constatons pour faites cette extension il y aura des configurations dans plusieurs niveaux au niveau du réseau MPLS.

III.4.2.2 OpenFlow

Dans cette partie nous allons appliquer la même approche utilisée dans le scénario MPLS en ajoutant un nouveau switch OVS S4 entre S1 et S3 comme il indiquée dans la Figure.III.40 :

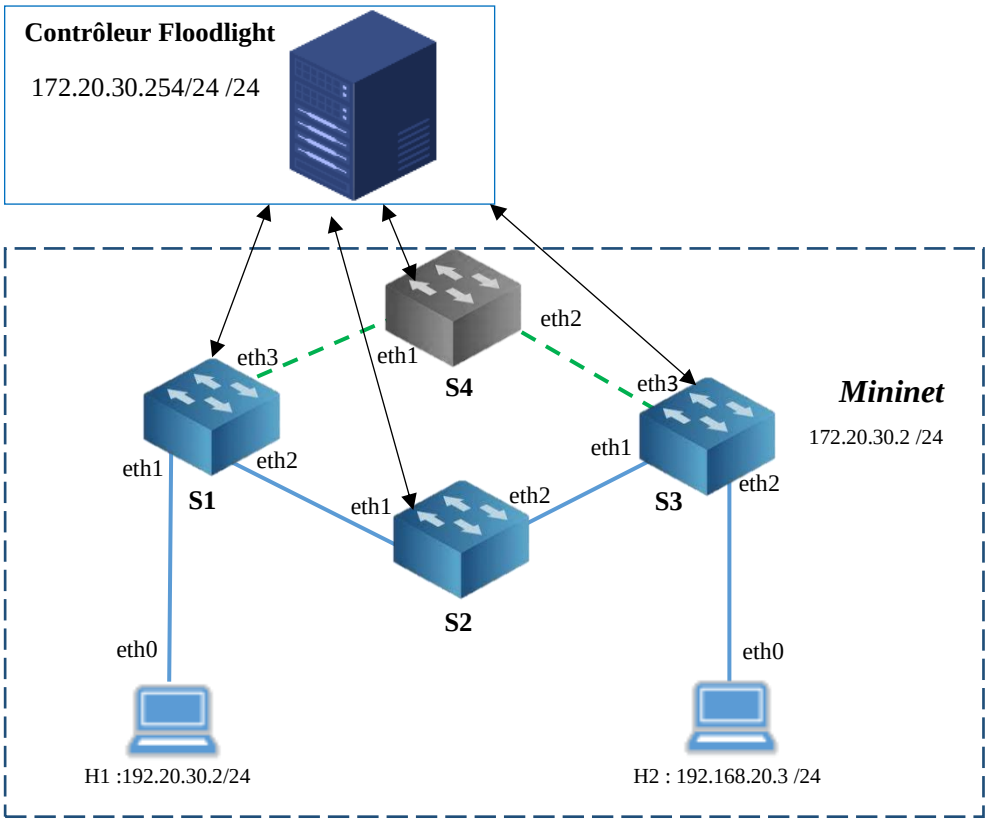


Figure.III.40 : Topologie OpenFlow scalabilité

Dans notre environnement virtualisé dans mininet pour procéder à ces changements c'est juste à ajouter dans le script python :

```
S4 = self.addSwitch( ‘s4’ )
Self.addLink( s1, s4 )
Self.addLink( s4, s3 )
```

Une fois que le script est exécuté, le contrôleur Floodlight détectera automatiquement l'ajout du switch S4 (Figure.III.41 et Figure.III.42).

local (s4)	DOWN	0	0	0	0	10	00000
1 (s4-eth1)	UP 10 Gbps FDX	1186	1499	14	21	00	00000
2 (s4-eth2)	UP 10 Gbps FDX	1186	1316	14	16	00	00000

Flows (7)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)
9007199254740992	0x0	1	in_port=1 eth_dst=62:7b:c3:4e:52:57 eth_src=32:0a:f2:08:7a:e6 eth_type=0x0x800 ipv4_src=192.168.20.2 ipv4_dst=192.168.20.3	actions:output=2	---	---	---	---	---	---	10	980	11
9007199254740992	0x0	1	in_port=2 eth_dst=32:0a:f2:08:7a:e6 eth_src=62:7b:c3:4e:52:57 eth_type=0x0x800 ipv4_src=192.168.20.3 ipv4_dst=192.168.20.2	actions:output=1	---	---	---	---	---	---	9	882	11

Figure.III.41 : Table de flux S4

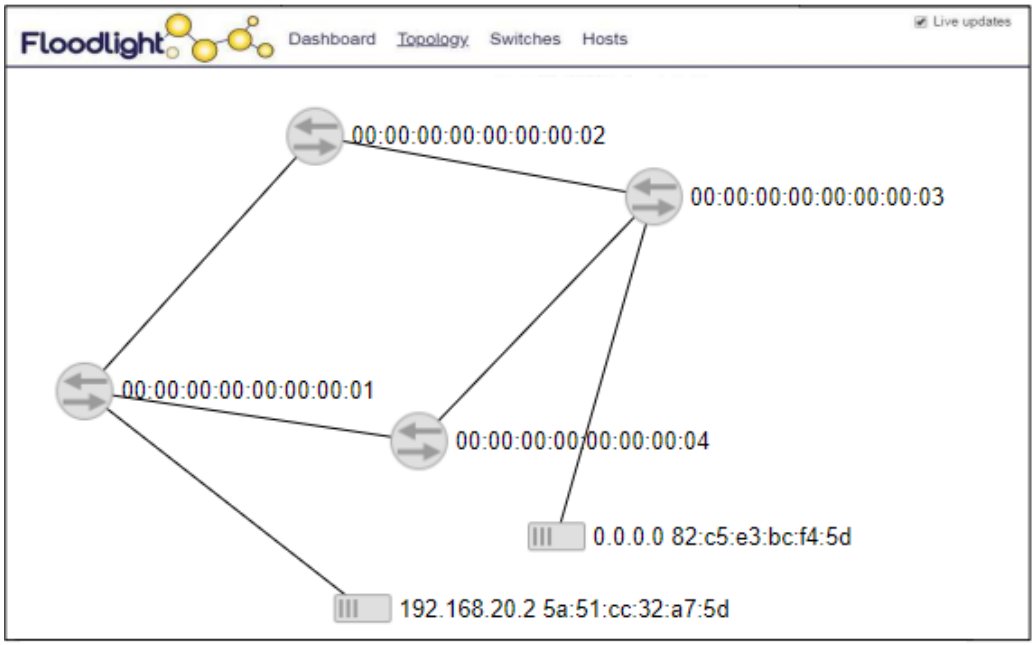


Figure.III.42 : Topologie scalabilité sur Floodlight

Dans le monde réel c’est juste déployer un switch OVS et l’interconnecter avec ses voisins et l’attribuer une adresse IP pour pouvoir communiquer avec le contrôleur.

III.4.3 Test de performance

L'objectif de cette partie est d'évaluer certains tests de performances dans les deux scénarios à savoir la latence, le débit, et la gigue.

III.4.3.1 Latence

La latence moyenne d'établissement des flux correspond au temps requis pour qu'un paquet aille de la source à une destination et revienne. Pour mesurer cette latence, nous utilisons la commande Ping, qui envoie un paquet ICMP à une adresse spécifique et attend ensuite la réponse ICMP.

A. MPLS

A Afin de mesurer la latence, nous exécutons la commande suivante et le résultat sera sauvegarder dans le fichier *test_ping.txt* à partir de la machine VM2 (Figure.III.43) :

Ping 192.168.10.2 -c 20 > test_ping.txt

N.B : **-c** signifie le nombre de paquets envoyés.

```
host2@host2-virtual-machine:~$ ping 192.168.10.2 -c 20 > test_ping.txt
host2@host2-virtual-machine:~$ cat test_ping.txt
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=61 time=2.36 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=61 time=3.40 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=61 time=8.52 ms
64 bytes from 192.168.10.2: icmp_seq=4 ttl=61 time=2.62 ms
64 bytes from 192.168.10.2: icmp_seq=5 ttl=61 time=2.53 ms
64 bytes from 192.168.10.2: icmp_seq=6 ttl=61 time=2.62 ms
64 bytes from 192.168.10.2: icmp_seq=7 ttl=61 time=3.01 ms
64 bytes from 192.168.10.2: icmp_seq=8 ttl=61 time=3.18 ms
64 bytes from 192.168.10.2: icmp_seq=9 ttl=61 time=2.40 ms
64 bytes from 192.168.10.2: icmp_seq=10 ttl=61 time=2.54 ms
64 bytes from 192.168.10.2: icmp_seq=11 ttl=61 time=2.20 ms
64 bytes from 192.168.10.2: icmp_seq=12 ttl=61 time=2.41 ms
64 bytes from 192.168.10.2: icmp_seq=13 ttl=61 time=2.44 ms
64 bytes from 192.168.10.2: icmp_seq=14 ttl=61 time=2.60 ms
64 bytes from 192.168.10.2: icmp_seq=15 ttl=61 time=2.00 ms
64 bytes from 192.168.10.2: icmp_seq=16 ttl=61 time=1.93 ms
64 bytes from 192.168.10.2: icmp_seq=17 ttl=61 time=3.32 ms
64 bytes from 192.168.10.2: icmp_seq=18 ttl=61 time=2.10 ms
64 bytes from 192.168.10.2: icmp_seq=19 ttl=61 time=3.03 ms
64 bytes from 192.168.10.2: icmp_seq=20 ttl=61 time=1.76 ms

--- 192.168.10.2 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19033ms
rtt min/avg/max/mdev = 1.769/2.853/8.528/1.374 ms
host2@host2-virtual-machine:~$
```

Figure.III.43 : Mesure de latence MPLS

B. OpenFlow

Afin de mesurer la latence, nous exécutons la commande suivante dans le xterminal de h2 et le résultat est sauvegardé dans le fichier *test_ping_of.txt* (Figure.III.44).

Ping 192.168.20.2 -c 20 > test_ping_of.txt

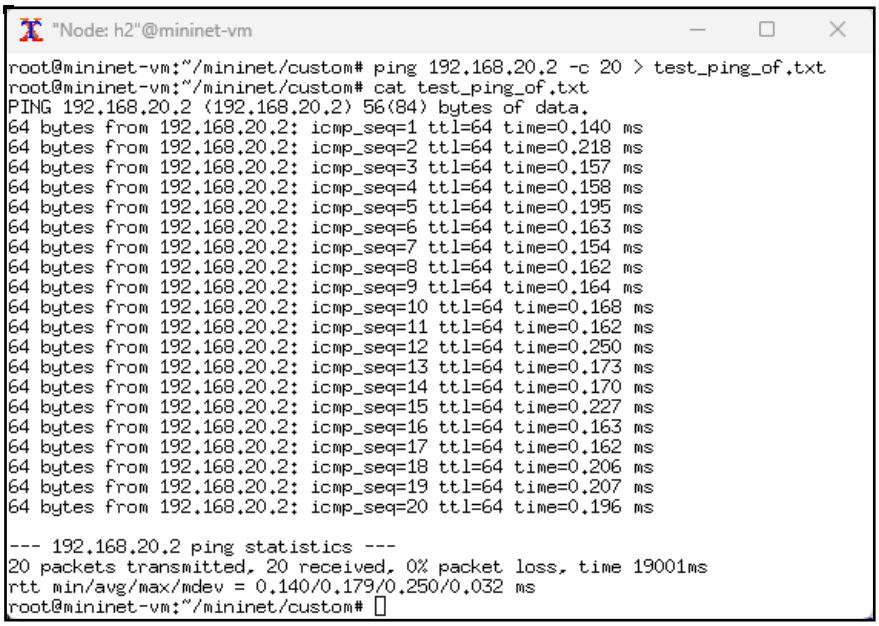


Figure.III.44 : Mesure de latence OpenFlow

III.4.4 Débit

Le débit est une mesure de la rapidité à laquelle les données peuvent être envoyées à travers un réseau et il est mesuré en bits par seconde (bps). La bande passante représente la capacité réelle de transfert qu'un support ou un réseau peut supporter mais cela ne signifie pas nécessairement que le débit est identique à la bande passante. En d'autres termes, le débit est la vitesse à laquelle les données peuvent passer à travers un support ou un réseau en une seconde.

III.4.4.1 MPLS

Pour tester le débit entre les deux VMs nous allons utiliser Iperf 3.

Commande de la partie serveur VM2 : *iperf3 –s –i 1*

Commande de la partie client VM1 : *iperf3 –c 192.168.20.2 –t 10 –i 1 –b 40mb*

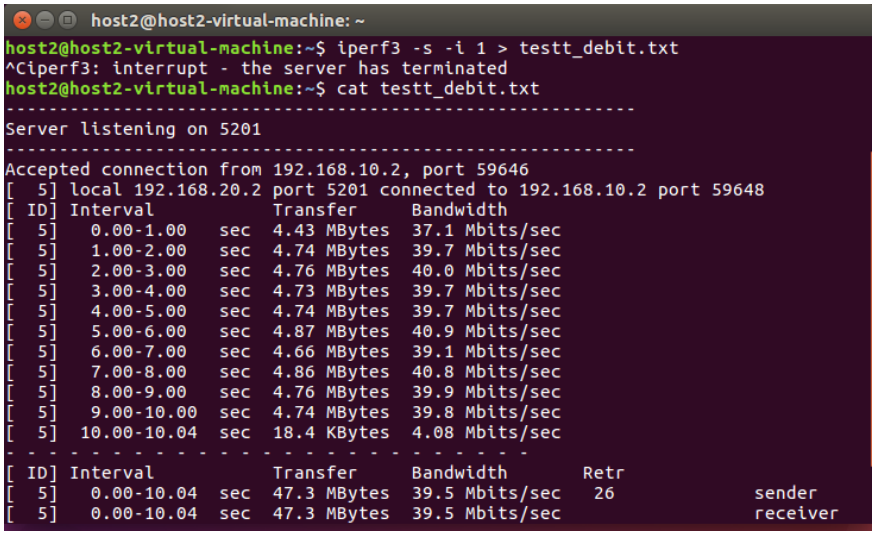


Figure.III.45 : Test débit MPLS coté serveur

A. OpenFlow

Pour tester le débit entre les deux hôtes nous allons utiliser Iperf3
Commande partie serveur : *iperf3 –s –i 1*
Commande partie client : *iperf3 –c 192.168.20.3 –t 10 –i 1 –b 40mb*

N.B : -s : serveur ; -i 1 : un pas chaque 1s.
-c : client ; -t 10 : durée de 10 s ; -b 40mb : limiter la bande passante à 40 Mbits.

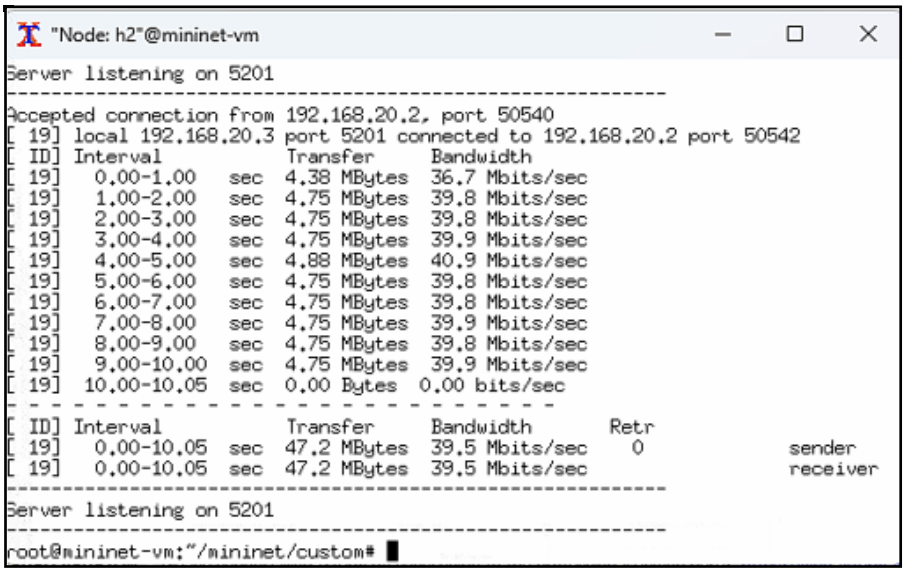


Figure.III.46 : Test débit OpenFlow coté serveur

III.4.4.2 Jitter

Le jitter, également appelé *gigue* se réfère à la variation du délai de transmission des paquets dans un réseau. Il mesure les fluctuations de la latence entre les paquets qui sont envoyés d'une source à une destination. Un faible jitter indique une transmission régulière et prévisible des paquets, tandis qu'un jitter élevé indique des variations significatives dans les délais de transmission. Le jitter peut entraîner des problèmes de qualité de service (QoS) tels que des retards, des interruptions ou une mauvaise qualité audio/vidéo lors de la communication en temps réel sur le réseau.

A. MPLS

Commande coté serveur VM2 : iperf3 -s -p 4000 -i 1 > test_jitter40mb.txt
Commande coté client VM1 *: iperf3 -c 192.168.20.2 -b 40mb -i 1 -u -p 4000*

```
host2@host2-virtual-machine:~$ iperf3 -s -p 4000 -i 1 >testt_jitter40mb.txt
^Ciperf3: interrupt - the server has terminated
host2@host2-virtual-machine:~$ cat testt_jitter40mb.txt
-----------------------------------------------------------
Server listening on 4000
-----------------------------------------------------------
Accepted connection from 192.168.10.2, port 35502
[  5] local 192.168.20.2 port 4000 connected to 192.168.10.2 port 49374
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datag
rams
[  5]   0.00-1.00   sec  4.34 MBytes  36.4 Mbits/sec  2.199 ms  0/555 (0%)
[  5]   1.00-2.00   sec  4.80 MBytes  40.2 Mbits/sec  1.538 ms  0/614 (0%)
[  5]   2.00-3.00   sec  4.80 MBytes  40.3 Mbits/sec  1.076 ms  0/615 (0%)
[  5]   3.00-4.00   sec  4.77 MBytes  40.0 Mbits/sec  1.062 ms  0/611 (0%)
[  5]   4.00-5.00   sec  4.70 MBytes  39.5 Mbits/sec  1.713 ms  0/602 (0%)
[  5]   5.00-6.00   sec  4.70 MBytes  39.4 Mbits/sec  3.299 ms  3/604 (0.5%)
[  5]   6.00-7.00   sec  4.81 MBytes  40.4 Mbits/sec  1.269 ms  0/616 (0%)
[  5]   7.00-8.00   sec  4.77 MBytes  40.0 Mbits/sec  2.442 ms  0/611 (0%)
[  5]   8.00-9.00   sec  4.84 MBytes  40.6 Mbits/sec  1.170 ms  0/620 (0%)
[  5]   9.00-10.00  sec  4.67 MBytes  39.2 Mbits/sec  1.483 ms  0/598 (0%)
[  5]  10.00-10.05  sec  8.00 KBytes  1.42 Mbits/sec  3.117 ms  0/1 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datag
rams
[  5]   0.00-10.05  sec  47.2 MBytes  39.4 Mbits/sec  3.117 ms  3/6047 (0.05%)
-----------------------------------------------------------
Server listening on 4000
-----------------------------------------------------------
host2@host2-virtual-machine:~$
```

Figure.III.47 : Test jitter MPLS coté serveur

B. OpenFlow

Commande coté serveur h2: iperf3 -s -p 4000 -i 1 > test_jitter_of1.txt
Commande coté client h1 : *iperf -c 192.168.20.3 -b 40mb -i 1 -t 10 -u -p 4000*

```
"Node: h2"@mininet-vm
^Croot@mininet-vm:~/mininet/custom# iperf -s -i 1 -u -p 4000 > test_jitter_of1.t
root@mininet-vm:~/mininet/custom# cat test_jitter_of1.txt
------------------------------------------------------------
Server listening on UDP port 4000
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 17] local 192.168.20.3 port 4000 connected with 192.168.20.2 port 60010
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 17]  0.0- 1.0 sec  4.98 MBytes  41.8 Mbits/sec   0.037 ms    1/ 3555 (0.028%)
[ 17]  0.0- 1.0 sec  141 datagrams received out-of-order
[ 17]  1.0- 2.0 sec  4.75 MBytes  39.9 Mbits/sec   0.024 ms    0/ 3391 (0%)
[ 17]  2.0- 3.0 sec  4.75 MBytes  39.8 Mbits/sec   0.019 ms    0/ 3385 (0%)
[ 17]  3.0- 4.0 sec  4.76 MBytes  39.9 Mbits/sec   0.117 ms    0/ 3395 (0%)
[ 17]  4.0- 5.0 sec  4.73 MBytes  39.7 Mbits/sec   0.135 ms    0/ 3377 (0%)
[ 17]  5.0- 6.0 sec  4.76 MBytes  39.9 Mbits/sec   0.027 ms    0/ 3395 (0%)
[ 17]  6.0- 7.0 sec  4.75 MBytes  39.8 Mbits/sec   0.089 ms    0/ 3388 (0%)
[ 17]  7.0- 8.0 sec  4.76 MBytes  39.9 Mbits/sec   0.023 ms    0/ 3396 (0%)
[ 17]  8.0- 9.0 sec  4.75 MBytes  39.9 Mbits/sec   0.039 ms    0/ 3391 (0%)
[ 17]  0.0- 9.9 sec  47.5 MBytes  40.1 Mbits/sec   0.027 ms  142/33871 (0.42%)
[ 17]  0.0- 9.9 sec  142 datagrams received out-of-order
root@mininet-vm:~/mininet/custom#
```

Figure.III.48 : Test jitter OpenFlow coté serveur

NB : -u : activer le mode UDP ; **-p** : spécifier le port d'écoute utilisé par le serveur

III.5 Résultat et discussions

Dans cette section, nous présentons les résultats d'évaluation des deux solutions MPLS et OpenFlow SDN.

- **Forwarding des paquets :**

Comme il est indiqué précédemment un réseau SDN s'articule sur la séparation entre le plan de contrôle et le plan de données où la décision de forwarding du trafic est centralisée au niveau du contrôleur.

Ce que nous avons constaté c'est que le protocole MPLS fonctionne sur les couches 2 et 3, pour déterminer l'acheminement de trafic dans les routeurs MPLS. Et plus précisément pour construire sa table LIB et FLIB il doit consulter la table RIB et FIB fournie par le protocole IGP utilisé dans notre cas l'OSPF. Le faites d'utiliser plusieurs processus dans les routeurs ainsi que les messages échangés entre les routeurs pour la construction des tables de forwarding va consommer plus de ressources dans ces équipements.

Par contre pour le protocole OpenFlow SDN opère sur la couche 2, la table des flux est construite sur la base des décisions prises au niveau du contrôleur (plan de contrôle) ce qui va alléger les ressources des switchs en termes de calculs et échange d'informations.

En ce qui concerne le forwarding du trafic (plan de données) au niveau du MPLS se fait en consultant la table LFIB par contre au niveau de l'OpenFlow se fait en consultant la table des flux (flow table).

Aussi parmi les avantages révolutionnaire au niveau du Open flow c'est que la décision du forwarding se base sur l'adresse source et la destination par contre au niveau du MPLS se fait que sur la base de la destination. Cette possibilité donne une flexibilité et programmabilité dans un réseau.

- **Scalabilité**

Concernant l'évolutivité et la scalabilité dans les deux technologies nous avons constaté que dans les réseaux MPLS ça demande des interventions manuelles sur le routeur nouvellement déployé ainsi que les routeurs voisins pour son intégration dans le réseau ce qui nécessitent des configurations dans plusieurs niveaux comme il est indiqué précédemment dans la partie test. Cette situation est plus complexe lors des déploiements et extensions dans les grands réseaux car elle nécessite plus de ressources humaines et plus de temps.

Par contre pour le protocole OpenFlow le déploiement et l'extension dans le réseau sera plus facile et plus rapide car ça ne nécessite pas beaucoup de configurations dans les nouveaux switchs OVS déployés (juste l'interconnexion physique et assuré une connectivité IP avec le contrôleur), car l'intégration dans le réseau est prise en charge par le contrôleur.

❖ **Performances (Latence, Débit et Jitter)**

En ce qui concerne les tests de performances nous constatons que le temps de latence de l'open flow est mieux que le MPLS comme indique le Tableau III.1 et la Figure.III.49 :

Tableau III.1 : Résultats moyens de la latence

Séquence ICMP	RTT (ms) MPLS	RTT (ms) OpenFlow
icmp_seq=1	2,36	0,14
icmp_seq=2	3,4	0,218
icmp_seq=3	8,52	0,157
icmp_seq=4	2,62	0,158
icmp_seq=5	2,53	0,195
icmp_seq=6	2,62	0,163
icmp_seq=7	3,01	0,154
icmp_seq=8	3,18	0,162
icmp_seq=9	2,4	0,164
icmp_seq=10	2,54	0,168
icmp_seq=11	2,2	0,162
icmp_seq=12	2,41	0,25
icmp_seq=13	2,44	0,173
icmp_seq=14	2,6	0,17
icmp_seq=15	2	0,227
icmp_seq=16	1,93	0,163
icmp_seq=17	3,32	0,162
icmp_seq=18	2,1	0,206
icmp_seq=19	3,03	0,207
icmp_seq=20	1,76	0,196
Moyenne	**2,853**	**0,179**

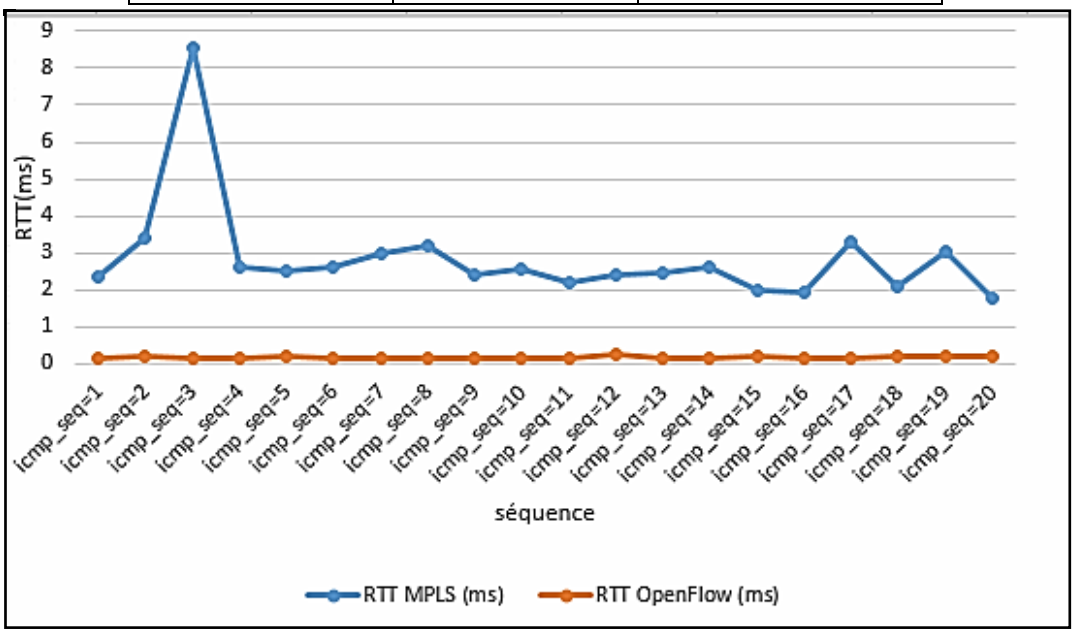


Figure.III.49 : Résultats des tests de la latence

Par contre les débits mesurés dans les deux scenarios presque ils sont identiques (Tableau III.2, Figure.III.50) vue que le forwarding dans les deux cas se fait pour le MPLS en consultant la table LFIB et tout juste après l'encapsulation en couche 2 et l'envoi du paquet vers le prochain saut. Et pour le OpenFlow en consultant la table des flux et tout juste après et l'envoie vers le prochain saut (couche 2).

Tableau III.2 : Résultats moyens de débit

Intervalle (second)	Débit MPLS (Mbit/sec)	Débit OpenFlow (Mbit/sec)
0.00-1.00	37,1	36,7
1.00-2.00	39,7	39,8
2.00-3.00	40	39,8
3.00-4.00	39,7	39,9
4.00-5.00	39,7	40,9
5.00-6.00	40,9	39,8
6.00-7.00	39,1	39,8
7.00-8.00	40,8	39,9
8.00-9.00	39,9	39,8
9.00-10.00	39,8	39,9

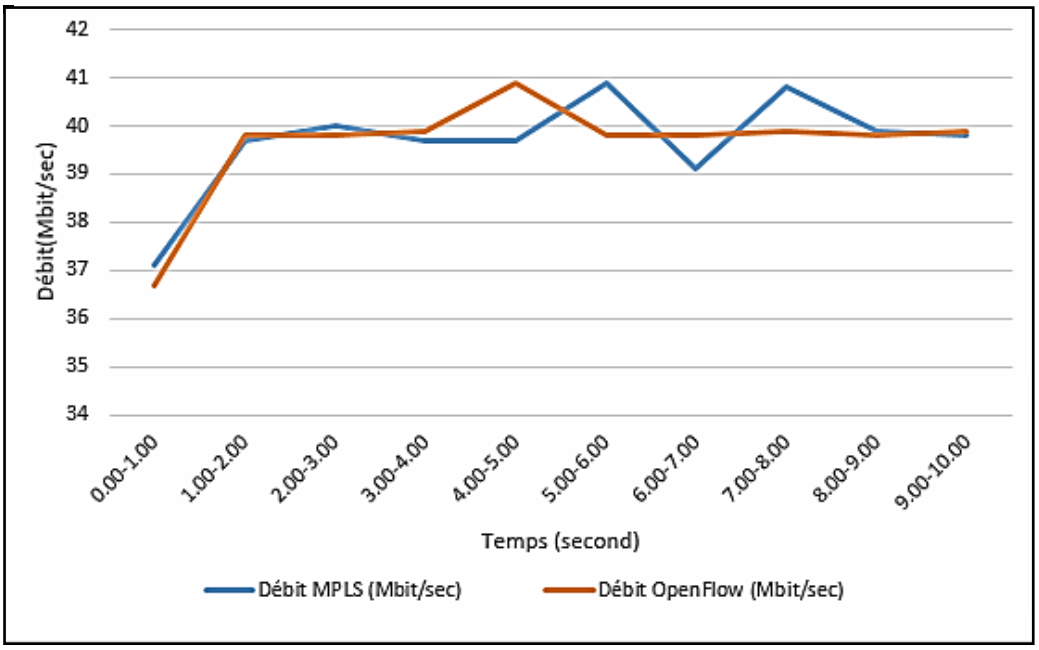


Figure.III.50 : Résultats des tests de débit

Concernant le test gigue nous avons constaté que la variation Jitter dans la solution Open Flow est meilleur légèrement par rapport au MPLS comme il est indiqué dans les tableaux III.3 et la Figure.III.51 :

Tableau III.3 : Résultats moyens de la gigue (Jitter)

Temp (s)	Jitter MPLS (ms)	Jitter OpenFlow (ms)
0.00-1.00	2,199	0,037
1.00-2.00	1,538	0,024
2.00-3.00	1,076	0,019
3.00-4.00	1,062	0,117
4.00-5.00	1,713	0,135
5.00-6.00	3,299	0,027
6.00-7.00	1,269	0,089
7.00-8.00	2,442	0,023
8.00-9.00	1,17	0,039
9.00-10.00	1,483	0,027
Moyenne	**3.117**	**0.027**

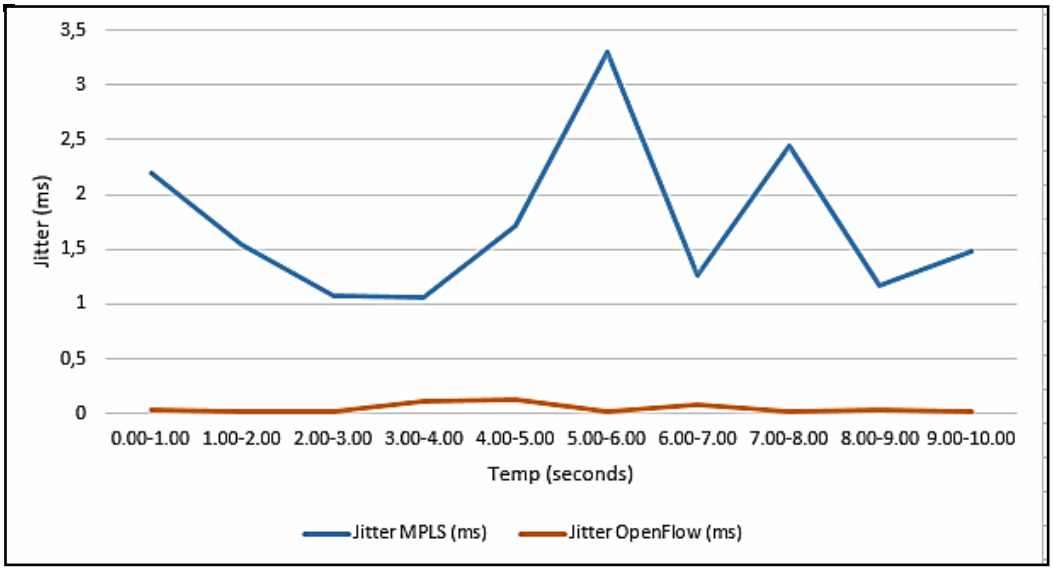


Figure.III.51 : Résultats des tests de la gigue

III.6 Conclusion

Dans cette importante phase et grâce à l'implémentation des différents scénarios nous a permis de comprendre en détail le fonctionnement des deux architectures MPLS et Open flow. Ce travail a été enrichi par des tests de scalabilité et évolutivité et aussi des tests de performance (Latence, Débit, Jitter).

Nous avons également pu constater les avantages offerts par le réseau Open SDN par rapport au réseau traditionnel MPLS. Le SDN présente notamment une plus grande flexibilité et programmabilité grâce à la séparation du plan de contrôle au plan de données.

CONCLUSION GÉNÉRALE

Conclusion Générale

Notre démarche dans ce projet de fin d'étude était d'aborder les aspects clés des technologies IP/MPLS et OpenFlow SDN. Et afin d'atteindre notre objectif il été nécessaire d'implémenter ces deux solutions dans un environnement virtuel, ce qui nous a permis d'explorer leurs comportements et le fonctionnements. La partie implémentation nous a permis de découvrir les avantages énormes que la technologie SDN peut offrir aux domaines réseaux.

L'approche de programmabilité des réseaux SDN offre la possibilité d'étendre les capacités et la taille du réseau grâce à des solutions logicielles de plus en plus intelligentes. Ce changement majeur réside dans le fait que le réseau évolue rapidement grâce à la programmation, plutôt que la limitation techniques des équipements.

Parmi les autres avantages du SDN est de dissocier la définition des politiques réseau, leur mise en œuvre dans le matériel de commutation et le transfert du trafic. La flexibilité souhaitée repose sur la décomposition du contrôle du réseau en éléments gérables contrairement au réseau MPLS qui porte la responsabilité du contrôle et forwarding des données dans chaque routeur.

Le SDN facilite également la création et l'introduction de nouvelles abstractions dans le réseau. Par conséquent, les réseaux deviennent personnalisables et programmables grâce à des applications (API) qui permettent de programmer les équipements réseau en utilisant différents langages.

De point de vue sécurité la présence d'un contrôleur SDN comme point central de commandement d'un réseau SDN pour la gestion des interactions entre les couches équipements réseaux (southbound) et applications (northbound) exige un haut niveau de protection des flux, ceci est afin d'éviter toute menace ou vulnérabilité dans le réseau. Aussi parmi les mesures à prendre est d'assurer une très haute disponibilité par la redondance des contrôleurs.

Actuellement le protocole OpenFlow est dans une phase d'expérimentation dans les domaines de recherche et vue les richesses qu'il offre pour mettre en œuvre un réseau SDN actuellement les géants de l'industrie des réseaux tel que Cisco, Huawei, Juniper … intègrent dans leurs équipements le support du protocole OpenFlow.

Pour les perspectives futurs de notre travail plusieurs recherches pourraient être apportées en profitant de l'aspect programmabilité qui offre le SDN et en particulier l'OpenFlow et de développer des API adaptées à des besoins particuliers dans différents environnement réseaux.

Références Bibliographiques

[1] p. atelin, (Réseaux informatiques notions fondamentales normes, architecture, modèle OSI, TCP/IP, Ethernet, WI-FI), edition ENI, 3ème édition, France, 2009.

[2] C. Servin, (*Réseaux et Télécom),* edition Dunod, Paris, 2003.

[3] d. gauchard, (*simulation hybrid des réseaux IP-Diffserv-MPLS multi services sur environnement d'éxécution distribuée), thèse de doctorat,* université paul sabatter - toulouse, 2003 https://theses.hal.science/tel-00011034/document.

[4] (*IP Routing : ISIS Configuration Guide*), edition Cisco Systems, Inc, USA, 2017.

[5] L. D. Ghein,(*MPLS Fundamentals),* edition Cisco Press*,* United States of America, 2007.

[6] p. andré, (*la sécurité des réseaux),*edition ISTE, Paris, 2014.

[7] (IMPLÉMENTATION DE MPLS AVEC CISCO) https://www.frameip.com/mpls-cisco/ (Consulté le : 29 mars 2023)

[8] Qu'est-ce que la virtualisation ? https://www.vmware.com/fr/solutions/virtualization.html#:~:text=La%20virtualisation%20s'appuie%,un%20seul%20et%20m%C3%AAme%20serveur. (Consulté le 6 avril 2023)

[9] O. L. S. e. T. MOUSSA, (*IMPLEMENTATION DU SDN DANS UNE STRUCTURE IP/MPLS),* projet fin d'étude*,* Tizi-Ouzou, 2018

https://www.ummto.dz/dspace/handle/ummto/6287.

[10] L. Zineb, (*Etude et implémentation d'une solution loadbalacing dans un réseau SDN),* projet fin d'édute, Tlemcen, 2020.

[11] N. K. G. Thomas D, (*SDN (Software Defined Networks)),* edition O'Reilly Media, United States of America, 2013.

[12] M. A. A. e. I. A. M. Alfudhail Gassan Alsharafi, (*Migrating UST Network To SDN and NFV),*projet fin d'étude, Yamen , 2018

https://ust.edu.ye/cit/wp-content/uploads/sites/6/2022/03/Migrating-UST-Network-To-SDN-and-NFV_2_compressed.pdf.

[13] A. A. e. A. Yanis, (*Etude et implémentation d'une architecture SDN LAN),*projet fin d'étude, Tzi Ouzou, 2018
https://www.ummto.dz/dspace/bitstream/handle/ummto/6707/AichaouiAnis_AitbelkacemYanis.pdf?

.

[14] A. S.-M. e. K. G. Szarkowicz, (*MPLS in the SDN Era),*edition O'Reilly Media, United States of America, 2015

[15] C. B. a. T. C. Paul Goranson, *Software Defined Networks (A Comprehensive Approch) edition Elsevier Inc,Second Edition, 50 Hampshire Street 5th Floor Cambridge MA 02139 United States, 2017*

[16] *(VMware Workstation) https://www.informatiweb-pro.net/virtualisation/vmware/vmware-workstation-15-modes-acces-reseau.html (Consulté le : 25 avril 2023).*

[17] *(Qu'est-ce qu'une machine virtuelle (VM) ?) https://www.oracle.com/fr/cloud/definition-machine-virtuelle-vm/ (Consulté le : 25 avril 2023.*

[18] *(Cisco CSR 1000v) https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/configuration/ b_CSR1000v_Configuration_Guide/b_CSR1000v_Configuration_Guide_chapter_00.html.* (Consulté le : 25 avril 2023

[19] *(Mininet Overview) http://mininet.org/overview/ (Consulté le 5 mai 2023).*

[20] *(Floodlight Controller) https://www.sciencedirect.com/topics/computer-science/floodlight-controller (Consulté le 5 mai 2023) .*

[21] *(What is iPerf / iPerf3 ?) https://iperf.fr/ (Consulté le : 13 mai 2023).*

[22] *(Wireshark) https://www.techtarget.com/whatis/definition/Wireshark (Consulté le : 13 mai 2023).*

Configuration

*** Étape 1– Adresse IP du core MPLS et OSPF ***

----------R1----------

```
hostname R1
!
router ospf 1
network 172.16.1.0 0.0.0.255 area 0
int lo0
ip ospf 1 area 0
!
int lo0
ip add 10.1.1.1 255.255.255.255
ip ospf 1 area 0

int g2
ip add 172.16.1.2 255.255.255.252
no shut
ip ospf 1 area 0
```

----------R2----------

```
hostname R2
!
router ospf 1
network 172.16.1.0 0.0.0.255 area 0
network 172.16.2.0 0.0.0.255 area 0
int lo0
ip ospf 1 area 0
!
int lo0
ip add 10.2.2.2 255.255.255.255
ip ospf 1 are 0

int g1
ip add 172.16.1.1 255.255.255.252
no shut
ip ospf 1 area 0

int g2
ip add 172.16.2.1 255.255.255.252
no shut
ip ospf 1 area 0
```

----------R3----------

```
hostname R3
!
```

```
router ospf 1
network 172.16.2.0 0.0.0.255 area 0
int lo0
ip ospf 1 area 0
!
int lo0
ip add 10.3.3.3 255.255.255.255
ip ospf 1 are 0

int g1
ip add 172.16.2.2 255.255.255.0
no shut
ip ospf 1 area 0
------------------
R1#ping 3.3.3.3 source lo0

*** Étape 2 - Configurer MPLS+ LDP sur toutes les interfaces du cœur MPLS ***
--------------R1-----------------
int g2
mpls ip
!
router ospf 1
mpls ldp autoconfig
--------------R2-----------------
int g1
mpls ip
int g2
mpls ip
!
router ospf 1
mpls ldp autoconfig
--------------R3-----------------
int g1
mpls ip
!
router ospf 1
mpls ldp autoconfig
!
end
!
sh mpls interface
sh mpls ldp neighbors

*** Étape 3 - Configuration MPLS BGP entre R1 et R3 ***
--------------R1-----------------
router bgp 1
 neighbor 10.3.3.3 remote-as 1
 neighbor 10.3.3.3 update-source Loopback0
```

```
 no auto-summary
 !
 address-family vpnv4
  neighbor 10.3.3.3 activate
----------R3----------
router bgp 1
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Loopback0
 no auto-summary
 !
 address-family vpnv4
  neighbor 10.1.1.1 activate

*** Étape 4 – creation des VRF ***
----------R1----------
ip vrf customer-1
rd 4:4
route-target both 4:4
!
int g1
ip vrf forwarding customer-1
ip add 192.168.10.1 255.255.255.0
no shut
!
router bgp 1
address-family ipv4 vrf customer-1
redistribute connected
!
end
!
sh run int g1
----------R3----------
ip vrf customer-1
rd 4:4
route-target both 4:4
!
int g2
ip vrf forwarding customer-1
ip add 192.168.20.1 255.255.255.0
no shut
!
router bgp 1
address-family ipv4 vrf customer-1
redistribute connected
!
end
!
sh ip route vrf customer-1
```

*** Étape configuration partie Scalabilité ***
----------R4----------
int g1
ip add 172.16.3.1 255.255.255.252
no shut
int g2
ip add 172.16.4.1 255.255.255.252
no shut
int lo0
ip add 10.4.4.4 255.255.255.255
----ospf---
router ospf 1
network 172.16.3.0 0.0.0.255 area 0
network 172.16.4.0 0.0.0.255 area 0
int lo0
ip ospf 1 area 0
--mpls--
int g1
mpls ip
int g2
mpls ip
----------R1----------
int g3
ip add 172.16.3.2 255.255.255.252
no shut
ip ospf 1 area 0
mpls ip
----------R3----------
int g3
ip add 172.16.4.2 255.255.255.252
ip ospf 1 area 0
mpls ip

Etapes d'installation des outils de travail

1. Installation VMware Worstation Pro 17

Télécharger le depuis ce site

https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

Après installation voici l'interface de logiciel

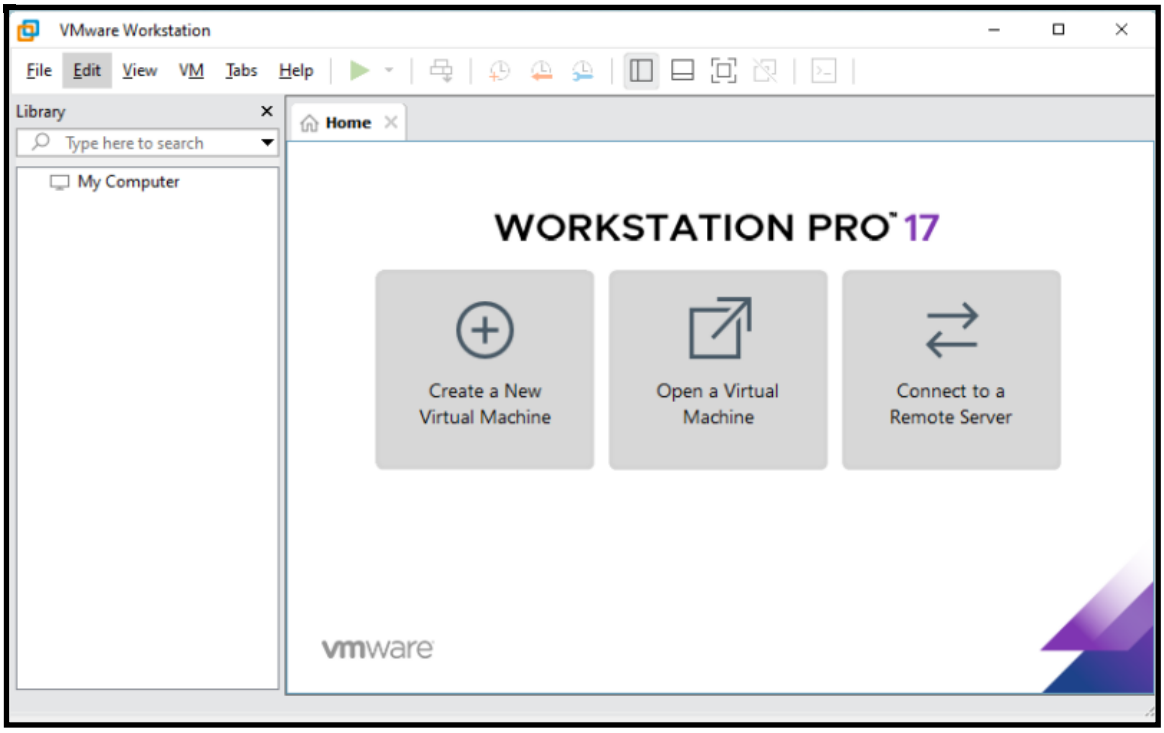


Figure.B.1 : Interface VMware Workstation pro17

2. Installation machine virtuelle Ubuntu

L'image de la VM Ubuntu est disponible dans ce site pour téléchargement

https://releases.ubuntu.com/16.04/

La version utilisée est la suivante

	ubuntu-16.04.6-desktop-i386.iso	2019-02-27 10:16	1.6G	Ubuntu 16.04.7 LTS (Xenial Xerus)

Allez sur VMware Workstation pro et ajouter une nouvelle machine virtuelle

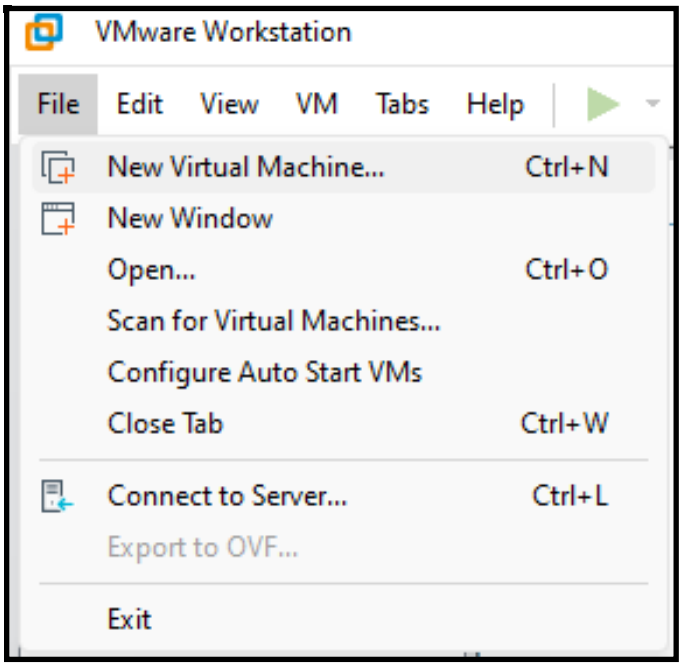


Figure.B.2 : Etape d'installation de VM Ubuntu

Sélectionnez "***Linux***" comme type d'OS et "***Ubuntu-64bit***" comme version de l'OS.

Configurez la quantité de mémoire vive et d'espace de stockage allouée à la machine virtuelle

Et Sélectionnez l'emplacement du fichier image *ISO* téléchargé à l'étape 1 comme disque d'installation.

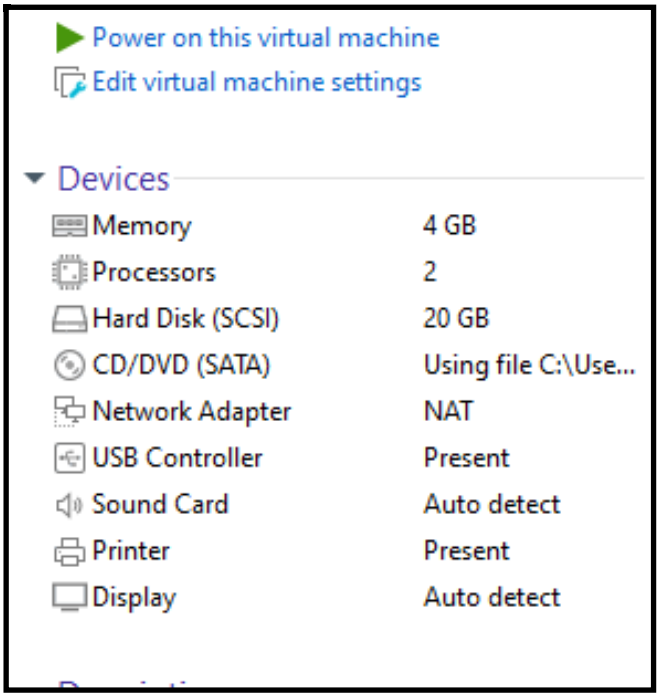


Figure.B.3 : Les paramètres à configuré dans les VM Ubuntu

Ensuite allumé la machine en cliquons sur Power on this virtual machine et commencé l'installation.

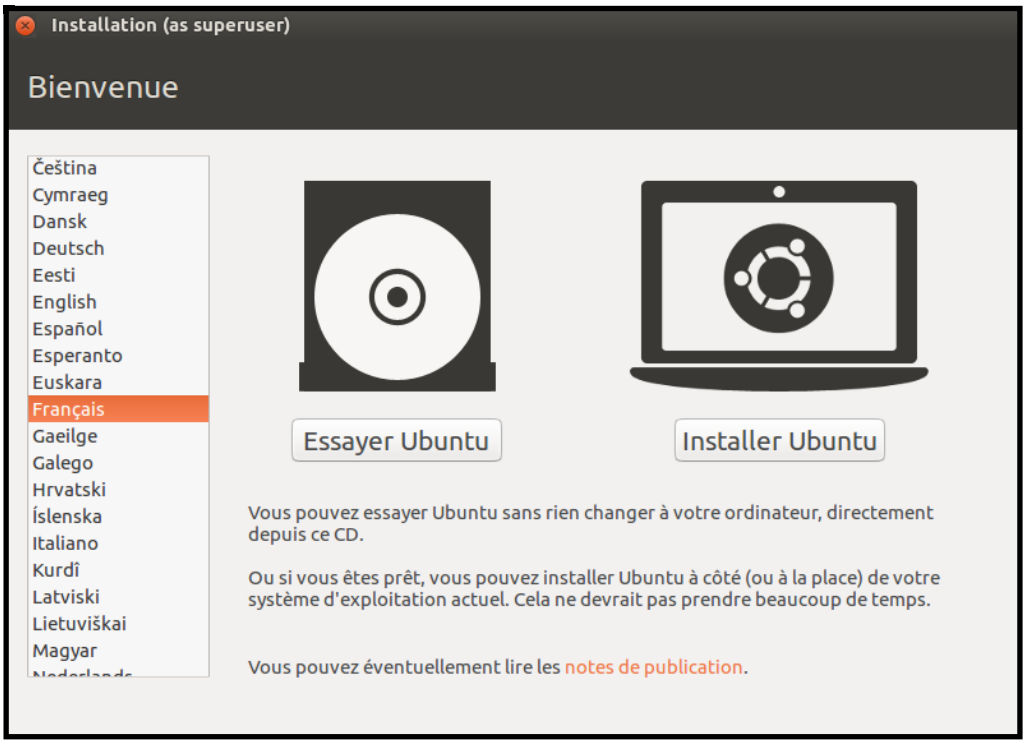


Figure.B.4 : Etape d'installation

Voici l'interface de VM Ubuntu utilisé (Figure.B.5) :

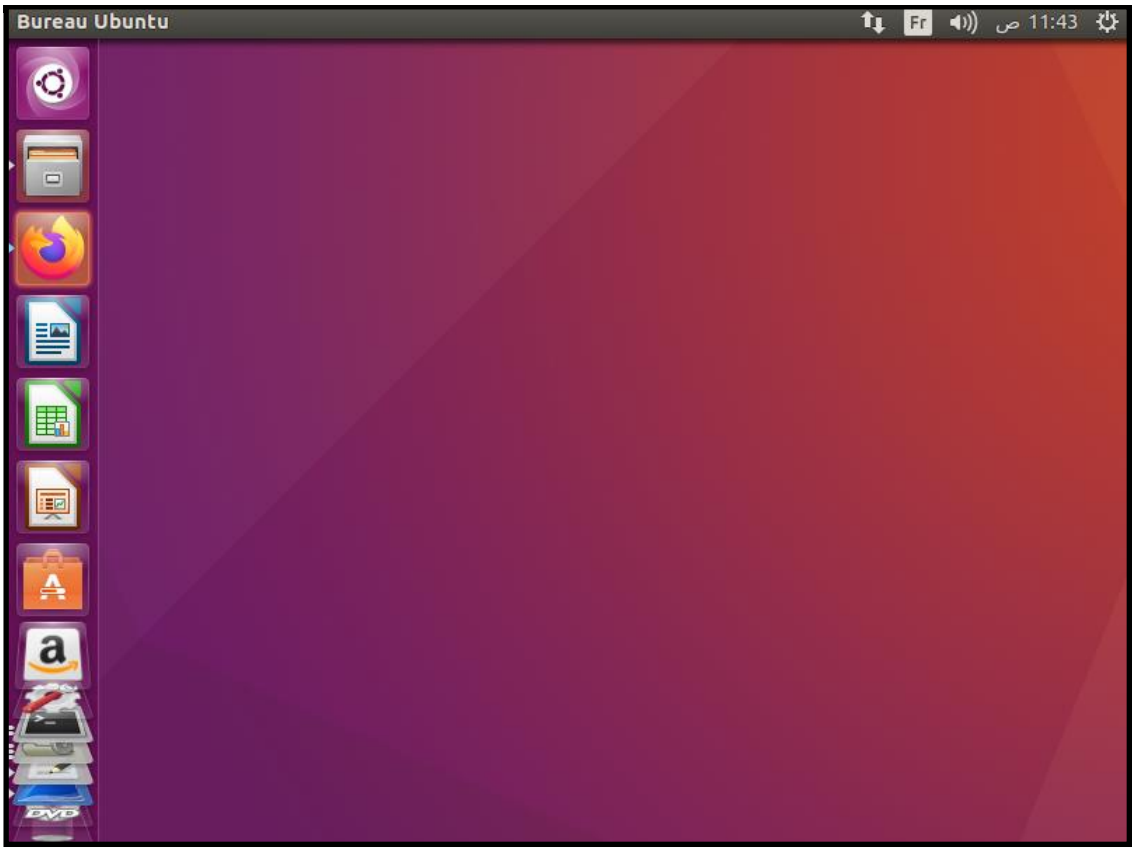


Figure.B.5 : Interface machine virtuel Ubuntu

3. Installation routeur CSR1000V

Pour avoir l'image de ce routeur il faut une License payante de 60jours.

- Ouvrir VMware Workstation et cliquer sur " ***New Virtual Machine***".
- Sélectionner "*custom*".
- Sélectionner "Installer disc image file (ISO)" (Installer un fichier image de disque ISO) et cliquer sur "*Browse*" pour sélectionner l'image disque Cisco CSR 1000v.

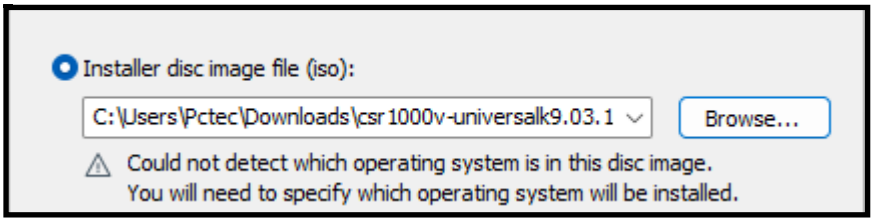


Figure.B.6 : Etape d'installation du routeur

- Sélectionner *Other* puis cliqué sur *Next* ;
- Cliquer sur *Next* et donner un nom à la VM ;
- Sélectionner un emplacement pour la VM et cliquer sur *Next* ;
- Sélectionner la taille du disque dur virtuel pour la VM et cliquer sur *Next*;
- Cliquer sur *Customize Hardware* (Personnaliser le matériel) pour modifier les paramètres matériels de la VM, tels que la RAM et le nombre de processeur ;
- Dans les paramètres de la VM, ajouter un adaptateur réseau virtuel et sélectionner "*Bridged*" (Ponté) pour permettre au routeur CSR 1000v d'accéder au réseau physique ;
- Démarrer la VM et ajouter "*Network Adapdater : Bridged*" pour avoir 2 interfaces de connexion ;
- Démarrer le routeur pour terminer l'installation.

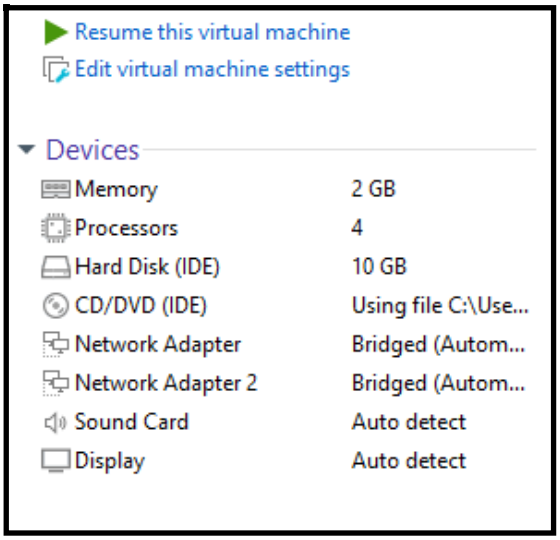


Figure.B.7 : Les paramètres à configuré dans les routeurs

4. Installation contrôleur

Après téléchargement en suivant les étapes d'installation suivante :
Ouvrir VMware Worstation et cliqué sur ***Create a New Virtual Machine*** et séléctionner *Custom*, puis cliquer sur *Next*, *Next*, sélecionner *I will install the operation system later*, Next, seléctionner *use bridged networking,,use an existing virtual disk,browse* et choisir le fichier extracter, ***edit virtual machine*** settings et donner les paramètres qui convient.

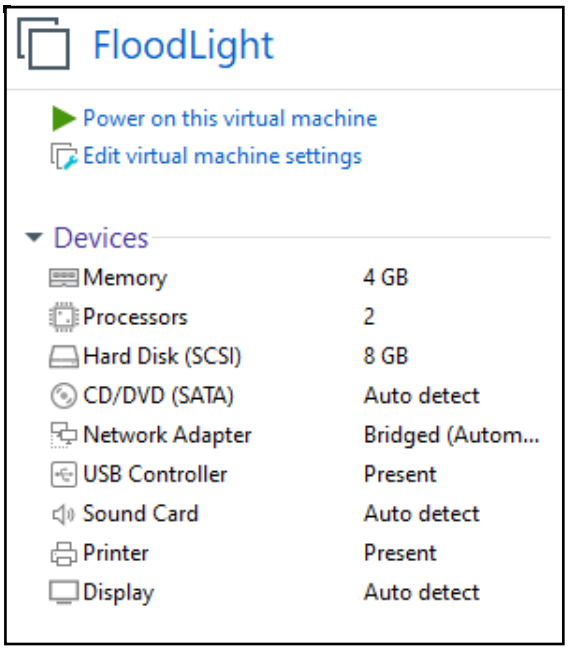


Figure.B.8 : Les paramètres à configuré dans le contrôleur

Cliquer sur ***Power on this virtual machine*** dès qu'il démarre entrer *floodlight* comme *password*.

Figure.B.9 : Interfaces de contrôleur Floodlight

5. Installation Mininet

Télécharger Mininet d'après le lien suivant :
https://github.com/mininet/mininet/releases/download/2.3.0/mininet-2.3.0-210211-ubuntu-16.04.7-server-amd64-ovf.zip
Ouvrir VMware Worstation et cliqué sur ***open a virtual machine*** et importer l'image du contrôleur, puis cliquer sur ***edit virtual machine*** settings et donner les paramètres qui convient.

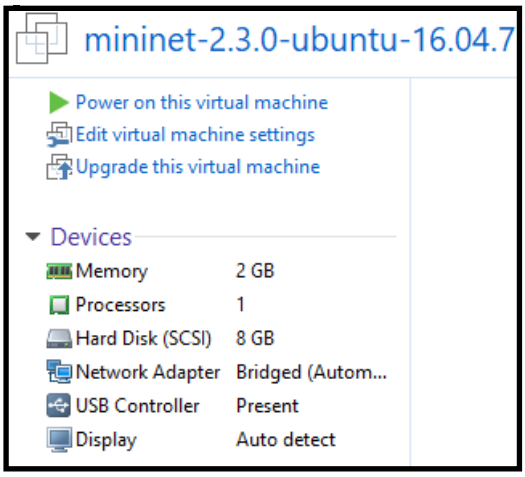


Figure.B.10 : Les paramètres à configuré dans Mininet

Cliquer sur ***Power on this virtual machine*** dès qu'il démarre entrer mininet comme *login* et *password*.

6. Installation Wireshark

Dans le contrôleur Floodlight on va faire les étapes suivantes
Entrer dans le terminal la commande suivante

- Mettez à jour les informations du package en exécutant la commande suivante :
- Installez Wireshark en exécutant la commande suivante :

Sudo apt-get install wireshark

Lorsque l'installation est terminé vous pouvez lancer Wireshark en exécutant la commande suivante : **Sudo wireshark**

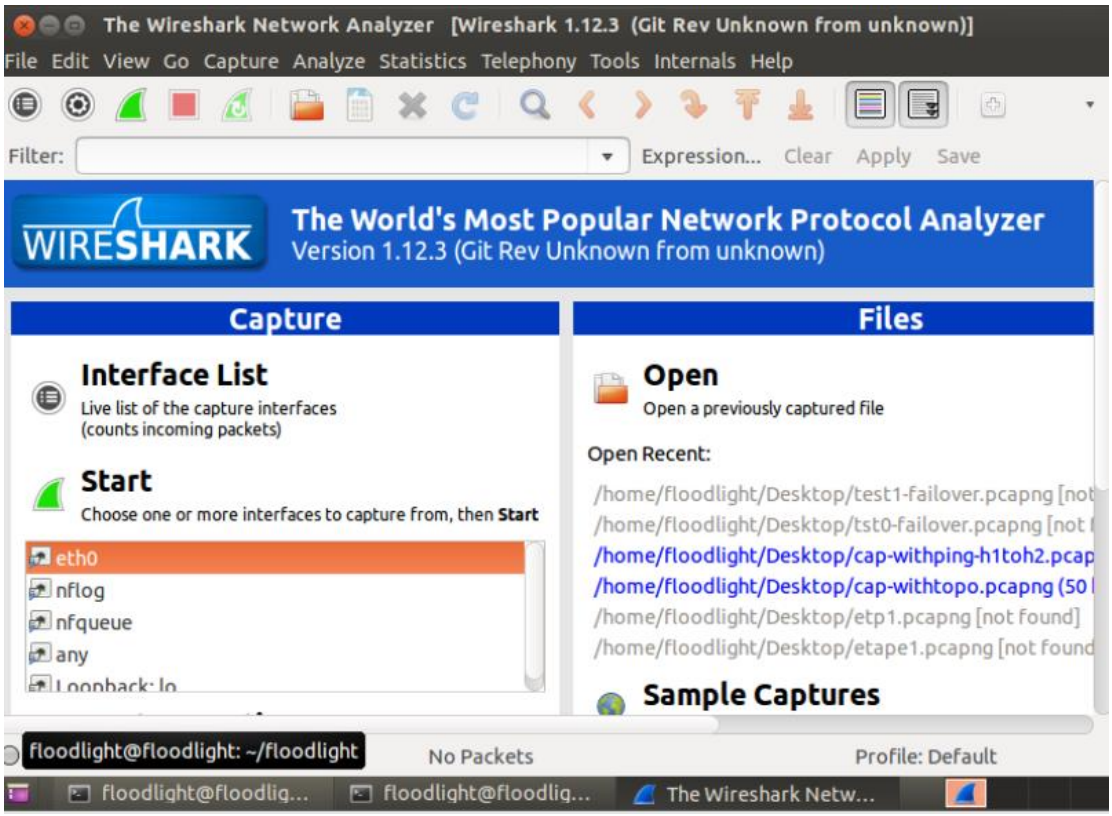


Figure.B.11 :Interface de Wireshark

7. Installation Iperf

- Télécharger dans VM Ubuntu iperf depuis le lien suivant : https://iperf.fr/iperf-download.php#ubuntu
- Ouvrez le terminal sur votre machine et tapez les commandes suivantes :
  Sudo apt-get update
  Sudo apt-get install iperf

RÉSUMÉ

MPLS (Multiprotocol Label Switching) est une technologie de réseau utilisée pour acheminer efficacement les paquets de données à travers un réseau.

SDN (Software Defined Network) est une technologie en développement qui permet de séparer le plan de contrôle du plan de données d'un équipement réseau, offrant une flexibilité ainsi une évolutivité indépendante du matériel.

Dans ce mémoire, nous avons évalué une infrastructure IP/MPLS ainsi qu'une infrastructure OpenFlow (SDN) dans un environnement virtuel afin de comparer leur fonctionnement et leurs performances.

Ce travail nous a permis de découvrir qu'il y a de grandes perspectives dans la technologie SDN. Elle est en train de continuer à évoluer et de gagner en popularité. Ce qui permet de transformer les réseaux traditionnels actuels en réseaux dynamiques et programmables. Avec autant d'avantages et de potentialités industrielles, le SDN deviendra le nouveau standard des futurs réseaux.

Mots clés : MPLS ; SDN ; OpenFlow ; VPN-L3 ; Virtuel.

ABSTRACT

MPLS (Multiprotocol Label Switching) is a network technology used to efficiently route data packets across a network.

SDN (Software Defined Network) is an emerging technology that separates the control plane from the data plane of a network device, offering flexibility and hardware-independent scalability.

In this thesis, we evaluated an IP/MPLS infrastructure as well as an OpenFlow (SDN) infrastructure in a virtual environment to compare their operation and performance.

This work has allowed us to discover significant prospects in SDN technology. It is currently evolving and gaining popularity, enabling the transformation of current traditional networks into dynamic and programmable networks. With its numerous advantages and industrial potentialities, SDN will become the new standard for future networks.

Keywords: MPLS, SDN, OpenFlow, virtual.

ملخص

MPLS (تبديل التسمية متعددة البروتوكولات) هي تكنولوجيا شبكة تستخدم لتوجيه حزم البيانات بكفاءة عبر شبكة.

SDN(شبكة محددة بالبرمجيات) هي تكنولوجيا متطورة تفصل مستوى التحكم عن مستوى البيانات في جهاز الشبكة، مما يوفر المرونة وقدرة التوسعية المستقلة عن الأجهزة.

في هذا الأطروحة، قمنا بتقييم بنية تحتية لبروتوكول (IP/MPLS) وبنية تحتية لبروتوكول OpenFlow باستخدام بيئة افتراضية لمقارنة عملها وأدائها.

هذا العمل سمح لنا باكتشاف آفاق كبيرة في تكنولوجيا SDN. إنها تتطور حاليًا وتكتسب شعبية، مما يمكّن تحويل الشبكات التقليدية الحالية إلى شبكات ديناميكية وقابلة للبرمجة. مع وجود هذا العدد الكبير من المزايا والإمكانيات الصناعية، ستصبح SDNالمعيار الجديد للشبكات المستقبلية.

الكلمات الرئيسية: IP/MPLS ; SDN ; OpenFlow