

**ETUDE ET MISE EN ŒUVRE D'UNE
SOLUTION SDN POUR LES HÔPITAUX DE
LA VILLE DE MOUYONDZI AU CONGO**



RÉSUMÉ

L'apparition du SDN a permis de faciliter la gestion des réseaux en offrant la possibilité de définir leurs politiques sous forme de programmes de contrôle. Ceci décharge les administrateurs du devoir de configurer chaque équipement à part afin d'implanter une politique donnée. Le programme de contrôle s'exécute sur un contrôleur qui se charge de le compiler pour générer les configurations nécessaires aux équipements afin de mettre en œuvre les politiques désirées. De nos jours, les contrôleurs permettent de spécifier les aspects de gestion des réseaux sous forme de requis de haut niveau.

Ce mémoire porte sur l'étude et la mise en œuvre d'une solution d'automatisation et contrôle des réseaux informatiques des hôpitaux de la ville de Mouyondzi. Nous considérons deux aspects, à savoir la configuration de flux et de règles, et la gestion de VXLAN. Les applications déployées sur les réseaux peuvent exiger la garantie ou la limitation de la bande passante afin de garder des performances bien déterminées. Finalement, les règles génériques permettent de définir des politiques globales en associant des actions à des types de paquets bien déterminés. Étant donné que notre but est de générer les configurations nécessaires pour l'implantation des aspects de gestion spécifiés sous forme de codes, nous avons défini plusieurs méthodes qui prennent en compte les détails du réseau. La mise en œuvre des aspects de la QoS, des files d'attente et les règles a été faite moyennant un logiciel interagissant avec l'API du contrôleur OpenDayLight. D'autre part, le déploiement du réseau de superposition VXLAN s'est réalisé en utilisant des Open vSwitch et gérés depuis le contrôleur Floodlight.

Ainsi, la gestion centralisée du réseau depuis un contrôleur permettrait d'atteindre un niveau de performances optimale au sein du réseau de tout le système hospitalier de la ville pour une meilleure prise en charge des patients et une diminution considérable des risques.

Mots clés : SDN, VXLAN, Contrôleur, OpenDayLight, Openflow, Open vSwitch, Floodlight



ABSTRACT

The emergence of software-defined networks has made it easier to manage networks by offering the possibility of defining their policies in the form of control programs. This relieves the administrators of the duty to configure each piece of equipment separately to implement a given policy. The control program runs on a controller that takes care of compiling it to generate the necessary configurations for the equipment to implement the desired policies. Nowadays, controllers allow to specify network management aspects in the form of high-level requirements.

This dissertation focuses on the study and implementation of an automation and control solution for computer networks in hospitals in the city of Mouyondzi. We consider two aspects, namely the configuration of flows and rules, and the management of VXLAN. Applications deployed on networks may require warranty or limitation bandwidth to maintain well-determined performance. Finally, the generic rules make it possible to define global policies by associating actions to well-defined packet types. Since our goal is to generate the configurations necessary for the implementation of the management aspects specified in the form of codes, we have defined several methods which consider the details of the network. The implementation of aspects of QoS, queues and rules were made using software interacting with the OpenDayLight controller API. On the other hand, the deployment of the VXLAN overlay network was carried out using Open vSwitches and managed from the Floodlight controller.

Thus, the centralized management of the network from a controller would make it possible to achieve an optimal level of performance within the network of the entire hospital system of the city for better patient care and a considerable reduction in risks.



DEDICACES

Je dédie ce présent document à :

Mes parents TALANTSI Rodrigue Elvis et GNEME Françoise qui ont œuvré pour ma réussite, de par leurs précieux conseils, leurs soutiens, moral et financier, infaillibles mais aussi et surtout pour leur amour inconditionnel.

Mes frères et sœurs pour leurs exemples de persévérance et de générosité, et pour leur soutien qui m'a permis de suivre une bonne formation.

Mes oncles et tantes pour leur encouragement dans mes études.

Tous mes amis et collègues, ils constituent un environnement favorable qui m'a permis de m'épanouir et d'avancer dans mes études.



REMERCIEMENTS

Mes remerciements s'adressent principalement au Dieu tout puissant créateur du ciel et de la terre qui m'a accordé la santé, le courage et la force nécessaire pour mener ce travail à terme.

L'achèvement de ce travail n'est pas seulement le fruit d'un effort personnel mais aussi celui de l'appui et des conseils de plusieurs volontés envers lesquelles je tiens à exprimer ma profonde gratitude.

Qu'il me soit donc permis de remercier particulièrement :

Mon encadrant Monsieur Samuel OUYA, pour sa patience et sa disponibilité tout au long de la réalisation de ce projet.

Je remercie l'Ecole Supérieure de Management et de Technologie, l'administration et le corps professoral de m'avoir permis de suivre une bonne formation.

Mes remerciements vont également aux membres du jury pour avoir accepté d'évaluer et d'apporter leur appréciation sur ce travail.



TABLE DES MATIERES

RESUME.....	I
ABSTRACT	II
DEDICACES.....	III
REMERCIEMENTS	IV
TABLE DES MATIERES.....	V
LISTE DES FIGURES	XI
LISTE DES TABLEAUX	XIII
LISTE DES SIGLES ET ABREVIATIONS.....	XIV
INTRODUCTION GENERALE.....	1
CHAPITRE 1 : PRESENTATION GENERALE.....	3
Introduction	3
1. Contexte justificatif de l'intérêt du sujet.....	3
1.2. Problématique.....	4
1.3. Objectifs de travail	4
1.3.1. Objectif final	4
1.3.2. Objectifs stratégiques.....	5
1.4. Démarche méthodologique.....	5
1.5. Plan.....	5
1.6. Résultats attendus	5
1.7. Présentation de Mouyondzi	6
Démographie de Mouyondzi	7
Infrastructures sanitaires.....	7
Collecte de données	7
a. Analyse de la situation informatique de l'hôpital général et des centres de soins intégrés.....	8



b. Evaluation des besoins informatiques de l'hôpital général et des centres de soins intégrés.....	9
Conclusion.....	10
CHAPITRE 2 : GENERALITES SUR LES RESEAUX DEFINIS PAR LOGICIELS	11
Introduction	11
2.1. Les motivations du SDN.....	11
2.1.1. Rappels sur les réseaux traditionnels.....	11
2.1.2. La Virtualisation	13
2.1.2.1. Principe de la virtualisation	13
2.1.2.2. Intérêts de la virtualisation	14
2.1.2.3. Inconvénients de la virtualisation	15
2.1.2.4. Les types de virtualisation	16
2.1.3. Cloud Computing	16
2.1.3.1. Caractéristiques du Cloud Computing.....	16
2.1.3.2. Services du Cloud Computing.....	17
2.1.4. La conteneurisation.....	18
2.1.4.1. Les outils de conteneurisation	18
a. LXC.....	18
b. Docker.....	19
c. OpenVZ.....	19
d. CoreOS RKT (Rocket).....	19
2.1.4.2. Standard de conteneurisation OCI.....	20
2.1.4.3. Orchestration des conteneurs.....	20
2.2. Concept des réseaux définis par logiciels.....	21
2.1.1. Définition.....	21
2.1.2. Avantages	21



2.3. Architecture du SDN	21
2.3.1. La couche infrastructure :	22
2.3.2. La couche contrôle.....	23
2.3.3. La couche application.....	23
2.3.4. Les interfaces de communication	23
2.3.4.1. Interfaces Sud	23
2.3.4.2. Interfaces Nord	24
2.3.4.3. Interfaces Est/Ouest.....	24
2.2.5. Orchestrateur	25
2.4. Principe de fonctionnement du SDN	26
2.5. Protocole OpenFlow	26
2.5.1. Définition d'OpenFlow	26
2.5.2. Architecture OpenFlow	27
2.5.2.1. Tables de flux	27
a. Champs de correspondance.....	27
b. Compteurs	28
c. Actions	30
2.5.3. Messages OpenFlow.....	31
2.5.3.1. Messages symétriques	31
2.5.3.2. Messages Contrôleur-switch	32
2.5.3.3. Messages asynchrones.....	33
2.6. Les Commutateurs SDN	34
2.6.1. Commutateur SDN logiciel	34
2.6.2. Commutateur SDN matériel	35
2.7. Quelques cas d'utilisation de réseaux définis par logiciels	35
2.7.1. Cas du Big Data.....	35
2.7.2. Cas du Data Center	36



2.7.3. Cas de l'IoT	38
2.7.4. Cas d'un réseau d'entreprise.....	38
2.7.5. Cas du réseau mobile 5G	39
2.8. Les Solutions SDN existantes	40
2.8.1. Présentation de quelques solutions SDN propriétaires.....	40
2.8.1.1. Contrail Network	40
2.8.1.2. Cisco DNA Center	42
2.8.1.3. HPE VAN SDN	43
2.8.1.4. Orion.....	44
2.8.2. Présentation de quelques contrôleurs SDN open source	45
2.8.2.1. Open Network Operating System (ONOS).....	45
2.8.2.2. Floodlight.....	46
2.8.2.3 OpenDayLight	47
2.8.2.4. Rvu controller	49
2.8.3. Tableau Comparatif des contrôleurs SDN Open Source	51
2.8.4. Choix de la solution SDN à implémenter	52
Conclusion.....	52
CHAPITRE 3 : IMPLEMENTATION DE LA SOLUTION DE RESEAU A DEFINITION LOGICIELLE.....	54
Introduction	54
3.1. Présentation du réseau à mettre en place.....	54
3.1.1. Présentation d'Open vSwitch et de l'émulateur Mininet-Wifi	57
3.1.1.1. Open vSwitch.....	57
a. Architecture OvS	57
b. Fonctionnement de OvS.....	58
3.1.1.2. Mininet	59
a. Fonctionnement	60



3.1.1.3. Mininet-Wifi.....	61
a. Architecture et composants	62
b. Fonctionnement	62
c. Emulation de support sans fil.....	64
3.1.2. Serveur OpenDayLight.....	65
3.1.2.1. Architecture OpenDayLight	65
3.1.2.2. Fonctionnement de ODL	67
3.1.3. Contrôleur Floodlight	67
3.1.3.1. Architecture de Floodlight	68
a. Fonctionnement de Floodlight	68
3.2. Mise en place de la solution SDN	69
3.2.1. Implémentation réseau virtuel SD-LAN.....	70
3.2.1.1. Topologie du réseau virtuel SDN	70
a. Prise en main du contrôleur OpenDayLight.....	70
b. Création de la topologie sous Mininet	72
3.2.1.2. Planification du réseau.....	75
a. Traffic Shaping.....	75
b. DiffServ	79
3.2.1.3. Installation de flux avec OpenFlow Manager.....	82
a. OpenFlow Manager	82
b. Installation de OFM.....	83
c. Configuration des flux.....	84
3.2.2. Déploiement d'un réseau de superposition de type VXLAN	85
3.2.2.1. Définition VXLAN	85
3.2.2.2. Principe de fonctionnement	85
3.2.2.3. Proposition de la topologie VXLAN	87
a. Déploiement du contrôleur Floodlight	88



b. Configuration réseau VXLAN.....	89
c. Gestion du réseau VXLAN	90
Conclusion	92
CONCLUSION GÉNÉRALE	93
BIBLIOGRAPHIE	110



LISTE DES FIGURES

Figure 1. 1: Carte géographique	6
Figure 2. 2: Architecture réseaux traditionnels.....	12
Figure 2. 3: Architecture de la virtualisation.....	14
Figure 2. 4: Architecture standard du SDN	22
Figure 2. 5: Communication via interfaces Sud	24
Figure 2. 6: Communication via interfaces Est/Ouest.....	25
Figure 2. 7: Architecture du SDN avec Orchestrateur.....	25
Figure 2. 8: Architecture OpenFlow (Kreutz et al, 2015)	27
Figure 2. 9: Entrée de flux	27
Figure 2. 10: Champs de correspondance OpenFlow	28
Figure 2. 11: Modèle de flux OpenFlow	31
Figure 3. 1: Architecture réseau du système hospitalier de Mouyondzi.....	56
Figure 3. 2 : Architecture OvS	58
Figure 3. 3: Instance réseau virtuel sur Mininet	61
Figure 3. 4: Architecture Mininet-Wifi	62
Figure 3. 5: Illustration simple du fonctionnement de Mininet-Wifi	63
Figure 3. 6: Architecture OpenDayLight version Oxygène	66
Figure 3. 7: Architecture Floodlight.....	68
Figure 3. 8: Installation Openjdk.....	70
Figure 3. 9: Installation ODL	71
Figure 3. 10: Extraction du fichier zip.....	71
Figure 3. 11: Vue du répertoire karaf-0.8.4.....	71
Figure 3. 12: Démarrage de ODL.....	71
Figure 3. 13: Installation des fonctionnalités GUI	72
Figure 3. 14: Connexion GUI.....	72
Figure 3. 15: Fichier de création de la topologie.....	73
Figure 3. 16: Topologie générée.....	74
Figure 3. 17: Topologie créée.....	74
Figure 3. 18: Vue de la topologie sous ODL.....	74
Figure 3. 19: Installation iperf3 serveur	76



Figure 3. 20: Installation iperf3 client	76
Figure 3. 21: Démarrage iperf3 serveur.....	76
Figure 3. 22: Démarrage iperf3 serveur.....	77
Figure 3. 23: Configuration QoS1 et File d'attente 1	77
Figure 3. 24: Extrait code mappage du flux	78
Figure 3. 25: Envoie du code avec Postman.....	78
Figure 3. 26: Test iperf avant QoS	79
Figure 3. 27: Test iperf après application QoS	79
Figure 3. 28: Envoie de la QoS via l'API.....	81
Figure 3. 29: Résultats des valeurs DSCP appliquées	81
Figure 3. 30: Architecture OpenFlow Manager.....	82
Figure 3. 31: Vue topologie réseau via GUI OFM	83
Figure 3. 32: Liste des flux sur OFM	84
Figure 3. 33: Programmation du flux de h1 à h3.....	85
Figure 3. 34: Ping de vérification des flux entre configurés	85
Figure 3. 35: Exemple topologie VXLAN	86
Figure 3. 36: Topologie VXLAN	88
Figure 3. 37: GUI Floodlight.....	88
Figure 3. 38: Installation d'OpenvSwitch	89
Figure 3. 39: Création d'un pont et connexion au contrôleur SDN.....	89
Figure 3. 40: Création des tunnels	89
Figure 3. 41: Configuration adresse IP OVS.....	90
Figure 3. 42: Test communication entre OVS3 et OVS2	90
Figure 3. 43 : Interface WebUI Floodlight	90
Figure 3. 44: Topologie réseau VXLAN via Web UI	91
Figure 3. 45: Création règle ACL.....	91
Figure 3. 46: Vue ACL créée.....	92
Figure 3. 47: Test communication bloqué	92



LISTE DES TABLEAUX

Tableau 1. 1 : Tableau des besoins informatiques.....	10
Tableau 3. 1 : Tableau des versions de OpenDayLight.....	66
Tableau 3. 2: Tableau récapitulatif des hôtes créés	75



LISTE DES SIGLES ET ABBREVIATIONS

API : Application Programming Interface
AWS : Amazon Web Services
BGP : Border Gateway Protocol
CAPEX : Capital Expenditure
CPU : Central Processing Unit
DHCP : Dynamic Host Configuration Protocol
DNS : Domain Name System
gRPC : high performance Remote Procedure Call
HTTP : Hypertext Transfer Protocol
I2RS : Interface to Routing System
IBM : International Business Machines
ICMP : Internet Control Message Protocol
IPv4 : Internet Protocol version 4
IPv6 : Internet Protocol version 6
IT : Information Tehcnology
LAN : Local Area Network
MAC OS : Macintosh Operating System
MD-SAL : Model-driven Service Abstraction Layer
MPLS : MultiProtocol Label Switching
NETCONF : NETwork CONFiguration protocol
NFV : Network Functions Virtualization
ONAP : Open Network Automation Platform
ONF : Open Network Foundation
OpenTSDB : Open Time Series Database
OPEX : Operating Expenses
OPNFV : Open Platform for NFV
OSGi : Open Services Gateway initiative
OSI : Open Systems Interconnection
PCE : Path Computation Engine
PCEP : Path Computation Element communication Protocol
PoE : Power over Ethernet



QoS : Quality of Service
REST : Representational State Transfer
SAL : Service Abstraction Layer
SDDC : Software-Defined Data Center
SDN : Software Defined Network
SDS : Software-Defined Storage
SFP : Small Form-factor Pluggable
SLA : Service Level Agreements
SNMP : Simple Network Management Protocol
STP : Shielded Twisted Pair
TCP : Transmission Control Protocol
TIC : Technologies de l'Information et de la Communication
TL1 : Transaction Language 1
TLS : Transport Layer Security
VLAN : Virtual Local Area Network
VM : Virtual Machine
VPN : Virtual Private Network
WAN : Wide Area Network
WLAN : Wireless Local Area Network
YANG : Yet Another Next Generation



INTRODUCTION GENERALE

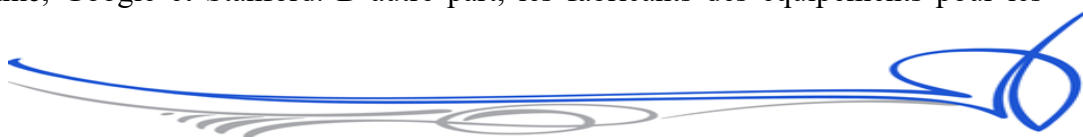
Durant ces dernières années, les réseaux informatiques classiques ont connu de grands défis qui résultent principalement des applications modernes déployées sur ces réseaux.

En effet nous assistons à un nouveau tournant dans le monde des réseaux, avec l'arrivée du Cloud, sa puissance de stockage et de calcul. Un constat immédiat de cette innovation est fait c'est la centralisation des contrôles et l'introduction d'un paradoxe : des commandes centralisées pour des systèmes complètement distribués. Par la puissance du Cloud, il est possible de contrôler chaque utilisateur, chaque application, chaque objet de l'Internet. Ces évolutions nécessitent généralement une mise à jour des infrastructures concernées. Toutefois, la mise à jour des réseaux classiques est réalisée en configurant manuellement chaque équipement ce qui peut produire plusieurs erreurs et incohérences. De plus, ces configurations prennent beaucoup de temps ce qui ne permet pas de faire face à l'expansion rapide du monde des technologies de l'information et de la communication.

C'est dans ce contexte qu'est apparu le paradigme des réseaux définis par logiciels (SDN) qui permet principalement de s'adapter à la nature dynamique des applications susmentionnées et de découpler les équipements physiques et les éléments de contrôle du réseau.

Les réseaux définis par logiciels (SDN) ont pour but principal de centraliser la logique déterminant les politiques de gestion d'un réseau dans une ou plusieurs unités appelées contrôleurs. Ces contrôleurs communiquent avec le reste des équipements du réseau via des interfaces ouvertes. De ce fait, la définition d'une politique de gestion d'un réseau revient à écrire des programmes et les déployer dans les contrôleurs. Dans la plupart des cas, ces programmes seront compilés, en tenant compte de la topologie et des ressources disponibles, afin de générer les configurations nécessaires à chaque équipement du réseau pour mettre en œuvre les politiques désirées. Ce nouveau paradigme offre également un meilleur contrôle de la QoS dans les réseaux grâce à sa nature centralisée.

L'architecture SDN a été adoptée par plusieurs grandes entreprises et universités comme, Google et Stanford. D'autre part, les fabricants des équipements pour les



réseaux tels que Cisco, HP et Juniper offrent désormais des solutions SDN qui permettent de gérer les centres de données.

Le travail qui a été défini pour ce projet concerne « l'étude et la mise en œuvre d'une solution SDN pour les hôpitaux de la ville de Mouyondzi au Congo ».

Afin de réaliser au mieux cette étude et mise en œuvre, le présent mémoire est organisé comme suit :

- Le premier chapitre est consacré à la présentation générale.
- Le deuxième chapitre est dédié au concept de réseaux définis par logiciels (SDN).
- Et enfin l'implémentation et l'exploitation de notre solution de réseaux définis par logiciels (SDN) feront l'objet du troisième et dernier chapitre.



CHAPITRE 1 : PRESENTATION GENERALE

Introduction

Avant de se lancer dans la réalisation de tout projet, il est nécessaire de savoir avec précision ce que l'on veut faire et pourquoi veut-on le faire. Cela présume l'existence des besoins réels que les objectifs du projet devront satisfaire.

Dans ce chapitre, à partir d'un contexte justificatif et d'une problématique précise sur laquelle notre travail est basé, nous énumérerons les objectifs du projet et ferons la présentation de la situation technologique du système hospitalier de la ville de Mouyondzi.

1. Contexte justificatif de l'intérêt du sujet

L'administration des soins et le traitement des patients pour la guérison de ces derniers font l'objet du métier de la médecine et donc l'activité principale du secteur médical dans un pays. Confrontés à un grand nombre de patients souffrants de différentes maladies, les hôpitaux et tout le corps médical de la ville de Mouyondzi au Congo ont de plus en plus du mal à traiter les informations des patients. On note de nombreuses difficultés de communication, d'accès aux applications en réseau et de partage de données (données sensibles) entre les hôpitaux de Mouyondzi faute d'infrastructure réseaux et de gestion des ressources réseau.

A cette ère des technologies de l'information et de la communication, le domaine de la santé présente une très grande variété de problèmes qui peuvent être résolus à l'aide de techniques informatiques. De ce fait les réformes, initiées par le nouveau ministre de la Santé et de la Population M. Gilbert MOKOKI, en cours depuis l'année 2022 visent non seulement à répondre aux besoins purement médicaux mais aussi à numériser tout le secteur de la santé au Congo.

C'est pourquoi dans cette étude, nous voulons mettre place une plateforme SDN en partant des installations déjà existantes et interconnecter les réseaux informatiques des deux (2) centres de soins intégrés et celui de l'hôpital général de la ville de Mouyondzi située dans le département de la Bouenza en république du Congo, afin d'augmenter l'efficacité globale dans le partage et traitement de données pour une meilleure fourniture de soins de santé à la population de cette ville.



1.2. Problématique

Le système d'information des hôpitaux de Mouyondzi fait face à une quasi-absence d'infrastructure réseaux. Il n'y a presque pas de système de communication et de partage de données entre les hôpitaux.

Le manque de communication et de gestion de ressources réseau pour les applications (E-LGH par exemple) qui feront faire partie du SI des hôpitaux, empêchent le système de santé et les établissements de pratiques médicales de partager les données de patients et de gérer les situations de transition des soins dans les dossiers de santé électronique.

Pour garantir un meilleur traitement dans la prise en charge des patients, les professionnels de la santé devraient documenter leur travail d'une manière spécifique. Ils devraient confirmer qu'ils ont administré un traitement aux patients, puis conserver et enregistrer les antécédents médicaux des patients à l'aide des formulaires électroniques qui seraient disponibles sur le réseau. Et avec des liens d'interconnexion entre réseaux de différents hôpitaux ces formulaires pourront être, avec les autorisations requises, consultés et utilisés pour traiter plus rapidement les patients étant transférés d'un hôpital à un autre dans la ville.

Ainsi la mise en place d'une solution de réseau à définition logicielle des réseaux informatiques des hôpitaux à Mouyondzi permettra non seulement la communication entre les hôpitaux mais aussi une gestion centralisée des réseaux de chaque hôpital en gérant la QoS pour l'acheminement du flux de données sensibles des différents hôpitaux et en spécifier des politiques de gestion moyennant des règles pour les applications qui requièrent une garantie de la bande passante pour garder de bonnes performances.

1.3.Objectifs de travail

1.3.1. Objectif final

L'objectif principal de ce travail est de mettre en place une solution SDN afin de permettre la gestion des ressources et la communication entre les hôpitaux de la ville de Mouyondzi. D'où le besoin de mettre en place une solution de contrôle centralisé des réseaux informatiques dans les hôpitaux dans la ville de Mouyondzi.



1.3.2. Objectifs stratégiques

L'atteinte de l'objectif final de notre projet passe par des objectifs spécifiques qui sont :

- ✚ Former tout le corps médical de Mouyondzi aux outils informatiques ;
- ✚ Procéder à un état des lieux des réseaux informatiques dans les hôpitaux de la ville ;
- ✚ Procéder à une analyse de faisabilité économique afin de faire des propositions susceptibles d'aboutir à la définition d'un cadre institutionnel de mesures d'assistance aux promoteurs de projets dans le secteur de la santé publique et des TICs ;
- ✚ Interconnecter les réseaux locaux des hôpitaux en un réseau unique à l'échelle de la ville.

1.4. Démarche méthodologique

La démarche méthodologique de notre travail est basée sur une étude de l'existant du point de vue de l'environnement et des installations déjà présentes pour en évaluer la fonctionnalité, la capacité, la qualité et la sécurité. Une analyse des besoins (équipements, autorisations, contrats, ...) pour la réalisation du projet sera également faite et en fin l'implémentation de la solution.

1.5. Plan

Le plan de ce document s'articule autour de trois chapitres. Le premier chapitre est réservé à la présentation générale justifiant l'intérêt du projet tout en dégagant la problématique et les objectifs fixés, mais aussi en présentant le lieu d'où le besoin de la réalisation du projet se manifeste.

Une présentation générale des concepts sur les réseaux définis par logiciels (SDN) fera l'objet du deuxième chapitre.

Et le dernier chapitre présentera l'implémentation de la solution SDN pour les hôpitaux de Mouyondzi.

1.6. Résultats attendus

Au terme de ce travail, la solution de réseaux définis par logiciels (SDN) permettra de :

- ✚ Centraliser la gestion des équipements des hôpitaux de la ville ;



- ✚ Partager des données entre tous les participants du système de soins de santé de la ville ;
- ✚ Rendre plus rapide et fiable la prise de décision de médecins sur les cas d'urgence ;
- ✚ Accéder plus rapidement aux données et applications pour un traitement plus efficace des patients ;
- ✚ Gérer plus efficacement les ressources réseau entre les hôpitaux.

Ainsi le système hospitalier de la ville de Mouyondzi pourra bénéficier d'une réelle infrastructure réseaux.

1.7. Présentation de Mouyondzi

Mouyondzi est une localité de class H (hydrographique) crée en 1911, elle est située dans le département de la Bouenza au sud de la République Du Congo. Ayant pour code de région Africa/Middle East, Mouyondzi est situé à 174 mètres d'altitude.

Les coordonnées géographiques sont 4°2'11" S et 11°49'30" E en DMS (degrés, minutes, secondes) ou -4.03639 et 11.825 (en degrés décimaux). Mouyondzi est limité par le district de TSIAKI au Nord, le district de KINGOUE au Nord – Est, le district de MABOMBO à l'Ouest et le district de YAMBA, au Sud. Le peuple de Mouyondzi est multi ethnique.

Le fuseau horaire pour Mouyondzi est UTC/GMT+1, le lever du soleil est à 08 :10 et le coucher du soleil est à 20 :17 heure locale (Africa/Brazzaville UTC/GMT+1). [8]



Démographie de Mouyondzi

Le district de Mouyondzi, actuellement dénommé Sous- Préfecture de Mouyondzi, est situé au Sud de la République du Congo, au nord du département de la Bouenza, sur une superficie de 950 km² [8]. Sa population est composée essentiellement des Bembé, Lali, Téké, et Minkengué. On compte environ 36.815 habitants dont 11.000 au centre de Mouyondzi. Les langues parlées à Mouyondzi sont : Bembe, Kinkengué, Lali, Munu-Kutuba, Téké.

On trouve à Mouyondzi un climat Bas-Congolais caractérisé par quatre saisons régulières dont TOMBO (petite saison de pluie d'Octobre à Décembre), MWANGA (petite saison sèche de Janvier à Février) NDOLO (grande saison de pluie de Mars à Juin) et KISIHU (grande saison sèche de Juillet à Septembre) favorisant la pratique des cultures maraîchères. L'économie locale dépend essentiellement des activités agro-pastorales et commerciales intenses.

Infrastructures sanitaires

Pour garantir la bonne santé aux habitants de Mouyondzi les infrastructures mises en place sont :

- ✚ 1 hôpital datant de la période coloniale nommé Robert Makele ex Moukala ;
- ✚ 2 centres de soins intégrés (CSI) ;
- ✚ 8 Centres de formations sanitaires privées ;
- ✚ 1 morgue publique ;

Collecte de données

Elle s'est faite, d'une part, à travers une enquête auprès du personnel médical des hôpitaux de la ville et de personnes ayant un ou plusieurs membres de leur famille hospitalisés. D'autre part, en procédant par observation nous avons pu obtenir des informations sur la situation numérique des infrastructures sanitaires de Mouyondzi.

Les données collectées sont présentées de la manière suivante :

- ✚ Le support d'enregistrement et traitement de données des patients dans les centres de soins intégrés et à l'hôpital ;
- ✚ La situation électrique et informatique de chaque centre de soins intégrés et de l'hôpital général ;



- La communication entre les centres de soins intégrés, mais aussi entre ceux-ci et l'hôpital général ;
- Le moyen de partage de données entre les infrastructures médicales de Mouyondzi et le Centre Hospitalier Universitaire à Brazzaville pour le transfert de patients sur les cas d'urgence ;

a. Analyse de la situation informatique de l'hôpital général et des centres de soins intégrés

L'hôpital Robert Makele dispose d'un très petit réseau informatique de type poste à poste. On compte environ cinq postes de travail reliés par un switch Linksys non-administrable.

Ces ordinateurs sont utilisés par les médecins et les infirmières pour saisir des rapports et faire des ordonnances, et pour la facturation sur ce qui concerne les frais de consultations et de traitement. Quant aux trois centres de soins intégrés, il n'y en a aucun qui dispose d'un réseau informatique.

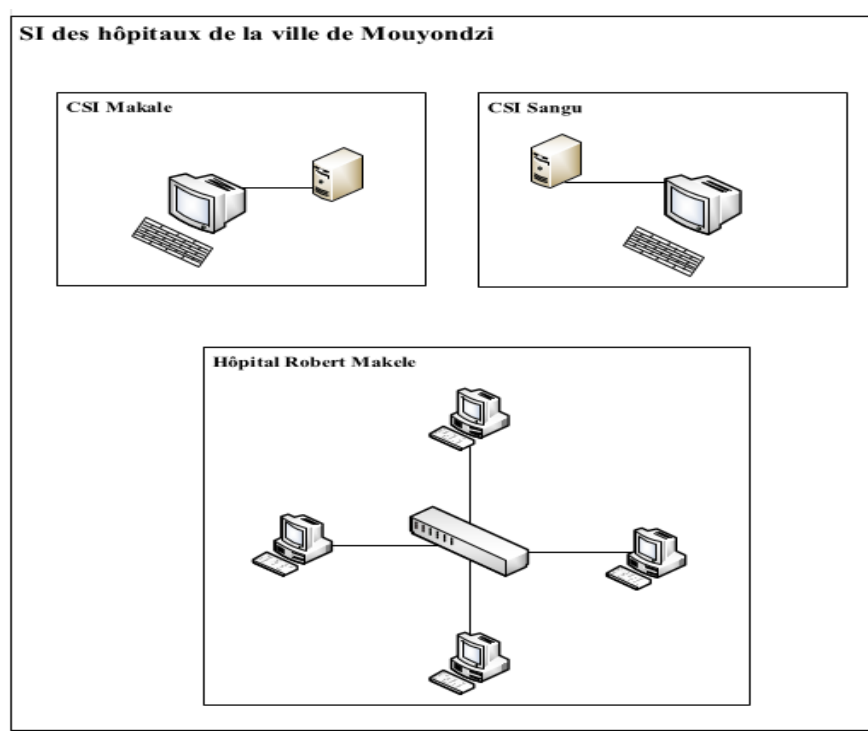


Figure 1. 2: Réseau actuel du système hospitalier de Mouyondzi

Toutefois avec les reformes qui sont apportées par le ministre de la Santé et de la Population M. Gilbert MOKOKI, pour l'optimisation de la gestion des établissements sanitaires au Congo, la situation numérique du système sanitaire de



Mouyondzi se verra améliorée au cours de l'année 2022. On peut déjà noter que la nouvelle application E-LGH (logiciel de gestion hospitalière développé par des jeunes congolais) et le logiciel Sage sont prévus dans le projet de numérisation de tout le système hospitalier du pays. [6]

b. Evaluation des besoins informatiques de l'hôpital général et des centres de soins intégrés

Les besoins informatiques du système hospitalier peuvent être déterminés comme étant une évaluation de la QoS, la bande passante, et du routage du trafic entre les réseaux des centres de soins intégrés et celui de l'hôpital général. A côté de ces besoins informatiques.

Ainsi, en se référant aux données collectées et à l'analyse faite de la situation informatique actuelle des centres de soins intégrés et de l'hôpital général, le bilan des matériels a été dressé.

- ✓ Un switch 48 port PoE 100/1000 Gigabit manageable pour chaque site ;
- ✓ La connectique (câble Ethernet Cat 6 STP) ;
- ✓ Un firewall avec un WLAN 802.11ac intégré, au moins 8 ports cuivre Gigabit intégrés + 1 port SFP ;
- ✓ Un routeur/modem pour l'accès internet fournis par l'opérateur Congo Telecom à installer à l'hôpital général ;
- ✓ Une liaison (fibre ou radio sur 300m maximum) entre les centres de soins intégrés et une autre liaison entre les centres de soins intégrés et l'hôpital général ;
- ✓ Un serveur sur lequel sera installé la solution SDN à mettre en place.

La gestion de QoS, de la bande passante, du routage du trafic entre les sites du système hospitalier et les règles de sécurité avec le firewall ainsi que l'accès internet seront réalisés depuis les locaux de l'hôpital général. Il conviendra donc de trouver des compétences en réseaux, systèmes et télécoms, pour assurer la gestion et l'exploitation de l'infrastructure qui sera mise en place.

Le total des besoins pour la mise en place de notre infrastructure nous conduit à un budget estimé à 5 538 215 FCFA illustré dans le tableau ci-dessous :



Tableau 1. 1 : Tableau des besoins informatiques

Quantité	Equipement	Prix Unitaire
3	NETGEAR (GS752TP) Smart Switch Ethernet PoE+ web manageable professionnel 52 Ports Gigabit (10/100/1000)	714 \$ soit 396.270 FCFA
3	Rouleau câble Ethernet blindé Cat 6 STP 305 mètres	35.000 FCFA
1	Firewall Sophos XG 126 Security Appliance	989 \$ soit 548.895 FCFA
1	Routeur / Modem Internet Congo Télécoms	35.000 FCFA
3	Liaisons Fibre, interconnexion entre les sites	148.740 FCFA
1	SERVEUR HPE ProLiant DL380 Gen10 4210R 1P 32G 8SFF sans disque[P56961-B21]	3.225000 FCFA
Total		5.538 215 FCFA

Conclusion

Ce chapitre achevé nous montre à travers son contexte justificatif, la problématique technologique du système sanitaire de Mouyondzi. Une fois, l'objectif général du projet fixé, la collecte de données faite, et la détermination des besoins effectués, il convient de procéder à la présentation des concepts généraux des réseaux SDN qui nous permettront de mieux réaliser l'implémentation de notre solution à définition logicielle.



CHAPITRE 2 : GENERALITES SUR LES RESEAUX DEFINIS PAR LOGICIELS

Introduction

Tout au long de ce chapitre, nous présentons les notions nécessaires à la compréhension de la suite du présent mémoire. Nous commençons par une présentation générale du concept des réseaux définis par logiciels (SDN). Par la suite, nous nous étalerons sur les principes de fonctionnement du SDN, d'OpenFlow et de leurs applications les plus significatives.

2.1. Les motivations du SDN

2.1.1. Rappels sur les réseaux traditionnels

Un réseau informatique est un ensemble d'équipements reliés entre eux pour échanger des informations. Outre des moyens informatiques, la mise en œuvre d'un réseau suppose des infrastructures telles que des liaisons physiques (câbles, ondes hertziennes...) et des équipements de transmission et d'interconnexion (carte réseau, routeur, switch...) Un réseau informatique nécessite également la mise en œuvre de protocoles de communication permettant de définir de façon standardisée la manière dont les informations sont échangées entre les équipements du réseau. Les différents types de réseaux ont généralement les points suivants en commun :

- Serveurs : ordinateurs ou applications qui offrent des services aux clients via le réseau.
- Clients : ordinateurs ou applications qui accèdent aux ressources et services fournies par un serveur sur un réseau.
- Support de connexion : conditionne la façon dont les ordinateurs sont reliés entre eux dans un réseau.
- Ressources partagées : fichiers, imprimantes ou autres éléments utilisés par les usagers du réseau.

Il est possible selon la taille et la portée du réseau informatique de différencier et de catégoriser les réseaux. Voici ci-dessous les principales catégories de réseaux informatiques :



- Personal Area Network (PAN) : utilisés sur des faibles distances (quelques mètres) pour interconnecter des équipements personnels comme les téléphones, ordinateurs portables, tablettes.
- Local Area Network (LAN) : réseaux locaux ou réseaux intra-entreprises, utilisés pour des distances de plusieurs centaines de mètres. Ils interconnectent les équipements informatiques d'une même entreprise. Les débits de ces réseaux vont aujourd'hui de quelques mégabits à plusieurs centaines de mégabits par seconde.
- Metropolitan Area Network (MAN) : réseaux métropolitains, c'est l'interconnexion des entreprises sur un réseau spécialisé à haut débit. Ces réseaux interconnectent plusieurs sites dans une même ville ou des réseaux locaux situés dans des bâtiments différents.
- Regional Area Network (RAN) : ont pour objectif de couvrir une large surface géographique. Dans le cas des réseaux sans fil, les RAN peuvent avoir une cinquantaine de kilomètres de rayon, ce qui permet, à partir d'une seule antenne, de connecter un très grand nombre d'utilisateurs.
- Wide Area Network (WAN) : réseaux étendus, ils interconnectent des sites et des réseaux à l'échelle d'un pays ou plusieurs continents et ils peuvent être terrestres ou satellitaires. [5]

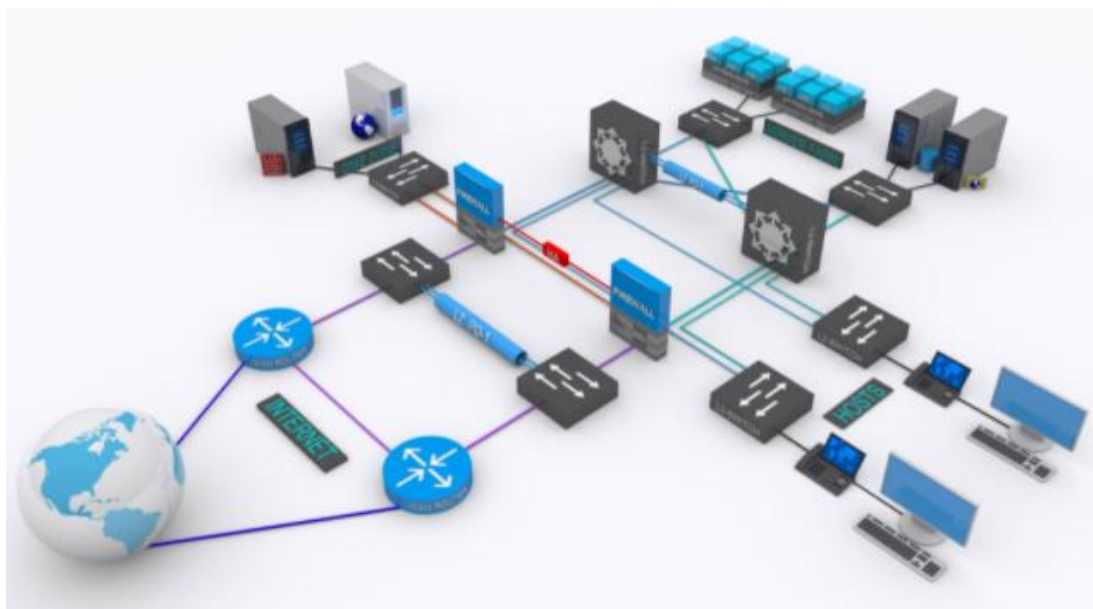


Figure 2. 1: Architecture réseaux traditionnels



2.1.2. La Virtualisation

La virtualisation est la capacité de simuler une plateforme matérielle, telle qu'un serveur, un périphérique de stockage ou une ressource réseau, dans un logiciel. Toutes les fonctionnalités séparées du matériel et simulées comme une instance « instance virtuelle », avec la capacité de fonctionner comme le ferait une solution matérielle traditionnelle. Une solution virtualisée est généralement beaucoup plus portable, évolutive et économique qu'une solution matérielle traditionnelle.

La virtualisation correspond à l'ensemble des techniques matérielles et/ou logiciels qui permettent de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines distinctes.

2.1.2.1. Principe de la virtualisation

Le principe de la virtualisation consiste à faire abstraction des infrastructures matérielles. En d'autres termes, le procédé de la virtualisation dématérialise les environnements systèmes et applicatifs afin de les faire fonctionner sur un seul et même serveur physique. Les instances se partagent ainsi les ressources du serveur comme la mémoire ou encore l'espace disque. La virtualisation informatique s'applique à différents niveaux comme le stockage de données ou encore le réseau, et cela, aussi bien, sur des serveurs que sur des postes de travail. [7]

Le principe de la virtualisation repose sur 3 piliers majeurs que sont le système hôte, l'hyperviseur et le système invité. La combinaison de l'ensemble de ces ingrédients permet de créer une virtualisation.

Sur le serveur unique utilisé, un système d'exploitation, qui est appelé également système hôte, est installé. Il représente l'OS principal pour accueillir les autres systèmes d'exploitation. Un logiciel de virtualisation dénommé hyperviseur est alors installé sur le système hôte. Son rôle est de pouvoir créer des environnements sur lesquels d'autres systèmes d'exploitation seront hébergés. Ces derniers sont appelés systèmes invités. Chaque environnement, appelé machine virtuelle, fonctionne de manière indépendante, mais peut disposer des capacités du serveur physique en termes de ressources hardware. Les machines virtuelles



ou VM (Virtual Machine) bénéficient, donc, chacune d'un accès à la mémoire, au processeur ou encore à l'espace disque.

Pour être utile de manière opérationnelle, la virtualisation doit respecter deux principes fondamentaux :

- **Le cloisonnement** : chaque système d'exploitation a un fonctionnement indépendant, et ne peut interférer avec les autres en aucune manière.
- **La transparence** : le fait de fonctionner en mode virtualisé ne change rien au fonctionnement du système d'exploitation et a fortiori des applications.

La transparence implique la compatibilité : toutes les applications peuvent tourner sur un système virtualisé, et leur fonctionnement n'est en rien modifié.

Pour ce qui est du cloisonnement, il existe bien sûr une interférence passive liée à la concurrence dans le partage des ressources. Mais nous verrons que ce partage peut être parfaitement contrôlé. [12]

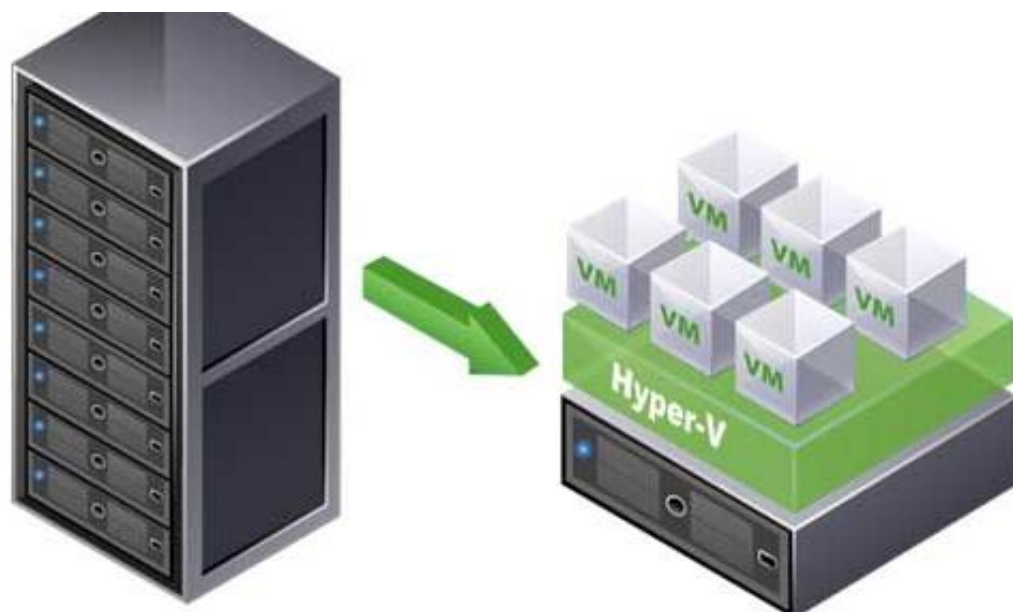


Figure 2. 2: Architecture de la virtualisation

2.1.2.2. Intérêts de la virtualisation

Les intérêts de la virtualisation sont :

- Isolation des différents utilisateurs simultanés d'une même machine (utilisation de type site central) ;



- Utilisation optimale des ressources d'un parc de machines (répartition des machines virtuelles sur les machines physiques en fonction des charges respectives) ;
- Installation, déploiement et migration facile des machines virtuelles d'une machine physique à une autre, notamment dans le contexte d'une mise en production à partir d'un environnement de qualification ou de préproduction, livraison facilitée ;
- Economie sur le matériel par mutualisation (consommation électrique, entretien physique, surveillance, support, compatibilité matérielle, etc.)
- Installation, tests, développements, cassage et possibilité de recommencer sans casser le système d'exploitation hôte ;
- Sécurisation et/ou isolation d'un réseau (cassage des systèmes d'exploitation virtuels, mais pas des systèmes d'exploitation hôtes qui sont invisibles pour l'attaquant, tests d'architectures applicatives et réseau) ;
- Allocation dynamique de la puissance de calcul en fonction des besoins de chaque application à un instant donné ;
- Diminution des risques liés au dimensionnement des serveurs lors de la définition de l'architecture d'une application, l'ajout de puissance (nouveau serveur, etc.) étant alors transparent.

2.1.2.3. Inconvénients de la virtualisation

Comme toute solution informatique, la virtualisation présente des contraintes :

- ✓ Le matériel doit être dimensionné au besoin (Processeurs, Disques, Mémoires.) ;
- ✓ Certaines solutions de virtualisation requièrent de la comptabilité avec le matériel ;
- ✓ Toute la sécurité de l'infrastructure virtuelle dépend de l'hyperviseur, ce qui en fait un point de défaillance unique ;
- ✓ Faibles performances ;
- ✓ Toutes les machines virtuelles invitées dépendent de la machine physique hôte.



2.1.2.4. Les types de virtualisation

Il existe différents types de virtualisation, visant à répondre à des besoins divers :

- ✓ La virtualisation de serveurs ;
- ✓ La virtualisation de l'espace de stockage ;
- ✓ La virtualisation d'application ;
- ✓ La virtualisation du réseau.

2.1.3. Cloud Computing

Le Cloud Computing, en français l'informatique en nuage (ou encore l'infonuagique au Canada), correspond à l'accès à des services informatiques (serveurs, stockage, mise en réseau, logiciels) via Internet à partir d'un fournisseur. Ceci est réalisé grâce au concept de virtualisation, qui résume la couche matérielle physique et la partage entre plusieurs hôtes, connues sous le nom de machines virtuelles (VMs). Le système responsable du mécanisme de l'extraction des ressources physiques et du contrôle de leur accès est connu sous le nom d'hyperviseur.

En général, le Cloud peut être classé comme privé, publique, hybride ou communautaire. Dans le Cloud Computing privé, l'administrateur VM a accès à l'hyperviseur, ce qui lui permet de gérer la disposition des VMs et de contrôler l'accès aux ressources physiques.

Dans le Cloud Computing public, hybride ou communautaire, l'administrateur VM n'a normalement pas accès à l'hyperviseur. Ainsi, l'administrateur ne peut pas déterminer le statut de la ressource physique VM, car le matériel est partagé entre plusieurs locataires.

2.1.3.1. Caractéristiques du Cloud Computing

Les principales caractéristiques du Cloud sont :

- Le service à la demande : un utilisateur peut allouer des ressources sans interaction humaine avec le fournisseur.
- L'accès au réseau élargi : les ressources sont accessibles via le réseau par des systèmes hétérogènes.



- Partage de ressources : les ressources sont dynamiquement affectées, libérées puis réaffectées à différents utilisateurs. L'utilisateur n'a pas besoin de connaître la localisation (pays, région, centre de données) des ressources.
- L'élasticité : les ressources peuvent être augmentées ou diminuées facilement, voir automatiquement, pour qu'elles s'adaptent aux besoins en temps réel. Les capacités offertes paraissent infinies, et l'utilisateur peut en consommer davantage, à tout moment sans se soucier d'une éventuelle limite.
- Les mesures : le système contrôle et optimise l'usage des ressources en mesurant régulièrement leur consommation. Ces mesures sont indiquées de manière transparente aux utilisateurs et au fournisseur pour le calcul de la facturation. [7]

2.1.3.2. Services du Cloud Computing

Du point de vue économique, le Cloud Computing est essentiellement une offre commerciale d'abonnement économique à des services externes. Selon le National Institute of Standards and Technology aux États-Unis, il existe trois principales catégories de services qui sont proposées en Cloud Computing : *IaaS*, *PaaS* et *SaaS*.

- *IaaS* (Infrastructure as a Service) : infrastructure en tant que service, en français, c'est le service de plus bas niveau. Il consiste à offrir un accès à un parc informatique virtualisé. Des machines virtuelles sur lesquelles le consommateur peut installer un système d'exploitation et des applications. S'apparentant aux services d'hébergement, l'*IaaS* dispense le consommateur de l'achat de matériel informatique.
- *PaaS* (Platform as a Service) : dans ce type de service, situé juste au-dessus du précédent, le système d'exploitation et les outils d'infrastructure sont sous la responsabilité du fournisseur. Le consommateur a le contrôle des applications et peut ajouter ses propres outils.
- *SaaS* (Software as a Service) : en français logiciel en tant que service. Dans ce type de service, des applications sont mises à la disposition des consommateurs. Les applications peuvent être manipulées à l'aide



d'un navigateur Web ou installées de façon locative sur un PC, et le consommateur n'a pas à se soucier d'effectuer des mises à jour, d'ajouter des patches de sécurité et d'assurer la disponibilité du service.

2.1.4. La conteneurisation

La conteneurisation est une méthode qui permet de virtualiser, dans un conteneur, les ressources matérielles (processeur, mémoire vive, etc.) systèmes de fichiers, réseau, nécessaires à l'exécution d'une application.

Dans un conteneur sont aussi stockées toutes les dépendances des applications : fichiers, bibliothèques, etc. Pour déplacer les applications virtuelles d'un système d'exploitation à un autre, le conteneur se connecte à leur noyau (kernel), ce qui permet aux différents composants matériels et logiciels de communiquer entre eux. Contrairement à la virtualisation de serveurs et à une machine virtuelle, le conteneur n'intègre pas de noyau, il s'appuie directement sur le noyau de l'ordinateur sur lequel il est déployé. [2]

Les conteneurs implémentent en général des méthodes pour :

- ✓ Créer un conteneur vide ;
- ✓ Ajouter des objets au conteneur ;
- ✓ Supprimer un ou plusieurs objets du conteneur ;
- ✓ Supprimer tous les objets du conteneur ;
- ✓ Accéder aux objets du conteneur ;
- ✓ Accéder au nombre d'objets du conteneur.

2.1.4.1. Les outils de conteneurisation

a. LXC

LXC (Linux Containers) est une plateforme de conteneurs open source qui promet une utilisation simple, ce qui est plutôt rare pour un système de conteneurs, et une expérience utilisateur intuitive et moderne. Pour cela, la plateforme offre différents outils, langues, modèles et bibliothèques. En outre, l'environnement de virtualisation peut s'installer et s'utiliser sur toutes les distributions courantes de Linux.



Dans la pratique, LXC assure un développement plus rapide des applications. La technique de conteneurs aide notamment pour le portage, la configuration et l'isolation. Lors de la diffusion de données en temps réel, les conteneurs jouent aussi un rôle majeur en mettant à disposition l'évolutivité nécessaire aux applications. Les Linux Containers s'adaptent à l'infrastructure ce qui les rend hautement indépendants et permet donc de les utiliser aussi bien en local que sur un Cloud ou dans un environnement hybride.

b. Docker

Docker offre également des temps de démarrage réduits, ce qui améliore l'utilisation des ressources. Cependant, au fur et à mesure que les conteneurs continuent d'évoluer, les problèmes de sécurité augmentent également.

c. OpenVZ

Basé sur des conteneurs linux, openVZ est un système de virtualisation permettant de créer plusieurs conteneurs isolés et sécurisés sur un seul serveur physique tout en garantissant que les applications n'entrent pas en conflit. Chaque conteneur fonctionne et s'exécute exactement comme un système autonome. Un conteneur peut être redémarré indépendamment et avoir un accès root, des utilisateurs, des adresses IP, de la mémoire, des processus, des fichiers, des applications, des bibliothèques système et des fichiers de configuration.

OpenVZ est un logiciel open source gratuit proposé par la société Virtuozzo, disponible sous GNU GPL.

d. CoreOS RKT (Rocket)

RKT est un moteur de conteneur d'applications développé pour les environnements de production cloud natifs modernes. Le projet RKT, introduit par CoreOS en décembre 2014, est disponible pour la plupart des principales distributions Linux et chaque version de RKT construit des packages rpm/deb autonomes que les utilisateurs peuvent installer.

RKT est basé sur l'application Container Image (ACI) telle que définie par la spécification App Container. Une ACI est une archive tar composée d'un système de



fichiers racine, qui contient tous les fichiers nécessaires à l'exécution d'une application, et d'un manifeste d'image.

Il présente une approche Pod-native, un environnement d'exécution enfichable et une surface bien définie qui le rend idéal pour l'intégration avec d'autres systèmes.

2.1.4.2. Standard de conteneurisation OCI

L'Open Container Initiative (OCI) est un projet collaboratif hébergé par la Fondation Linux conçu pour établir des normes communes pour les conteneurs. L'Open Container Initiative réunit un nombre important d'entreprises de premier plan dont : Red Hat, Amazon Web Services, Docker, CoreOS, Microsoft, VMware, EMC, Nutanix, Goldman Sachs, IBM et Google. Ce projet a été réalisé dans le but de garantir un standard ouvert et formel pour les conteneurs et les futures plates-formes de conteneurisation qui préservent la nature flexible et ouverte des conteneurs. Depuis le don du format de conteneur, du code d'exécution et des spécifications fait par Docker au projet, il existe aujourd'hui deux spécifications en cours de développement et en cours d'utilisation : la spécification d'exécution (runtime-spec) et la spécification d'image (image-spec).

2.1.4.3. Orchestration des conteneurs

En raison du nombre de plus en plus croissant de conteneurs pouvant être utilisés pour effectuer des tâches spécifiques (par exemple micro-services) le besoin d'une automatisation de ces conteneurs s'est fait ressentir, conduisant au développement d'outils d'orchestration de ceux-ci.

De ce fait l'orchestration de conteneurs consiste en l'automatisation d'une grande partie de l'effort opérationnel requis pour exécuter des charges de travail et des services conteneurisés. Cela inclut un large éventail de choses dont les équipes logicielles ont besoin pour gérer le cycle de vie d'un conteneur, y compris le provisionnement, le déploiement, la mise à l'échelle (à la hausse et à la baisse), la mise en réseau, l'équilibrage de charge et plus encore.

Il existe de nombreux outils d'orchestration de conteneurs disponibles sur le marché parmi lesquels on peut citer : Kubernetes, OpenShift, Docker Swarm, Ranchers, etc.



2.2. Concept des réseaux définis par logiciels

2.1.1. Définition

Le **Software-Defined Network** (SDN) est un modèle d'architecture réseau qui permet aux administrateurs de gérer les services de réseaux par abstractions de fonctionnalités. Il consiste en un nouveau paradigme permettant de définir le comportement des équipements du réseau moyennant un logiciel de contrôle. [9]

2.1.2. Avantages

Les infrastructures réseaux courantes utilisent généralement des équipements (commutateurs, routeurs. . .) dont la configuration dépend des fabricants. Une fois déployées, il s'avère difficile de faire évoluer ces infrastructures en adoptant de nouveaux protocoles ou en ajoutant et supprimant des équipements. De ce fait émerge l'idée des réseaux définis par logiciels (SDN). Le SDN offre de nombreux avantages par rapport aux réseaux traditionnels, notamment :

- ✚ Contrôle renforcé avec plus de souplesse et un débit accéléré ;
- ✚ Infrastructure réseau personnalisable ;
- ✚ Sécurité renforcée ;
- ✚ Délais plus courts de mise en production de nouvelles applications ;
- ✚ Economies CAPEX (coûts d'investissement), et OPEX (les coûts récurrents) à long terme pour les opérateurs ou fournisseurs de services ;

2.3. Architecture du SDN

De façon globale, les réseaux définis par logiciels se présentent en une architecture en trois (03) couches et d'interfaces de communication. Cette architecture que l'on présente de façon simpliste s'avère en réalité complexe et structurée. Elle découple la partie de contrôle de la partie transmission des équipements d'interconnexions afin d'ouvrir les équipements réseaux aux innovations et au développement de nouveaux services. [3]

Les couches constituant de l'architecture du SDN sont :

- ✚ La couche infrastructure ;
- ✚ La couche contrôle ;
- ✚ La couche application ;



Ces trois couches communiquent à l'aide d'interfaces de programmation d'applications (API) respectives en direction du nord et du sud. Par exemple, les applications communiquent avec le contrôleur via son interface en direction du nord, tandis que le contrôleur et les commutateurs communiquent via des interfaces en direction sud, tel que OpenFlow – bien que d'autres protocoles existent.

Dans les architectures de fournisseur de services ou opérateurs, la fonction de programmation de bout en bout est assurée par un orchestrateur. Ce dernier reçoit un ordre depuis une application et réalise ensuite la suite d'actions nécessaires pour mener à bien la tâche demandée. Pour réaliser sa mission, l'orchestrateur va pouvoir s'appuyer pour chaque élément de la chaîne sur le/les contrôleurs déployés (Durand, 2015). [1]

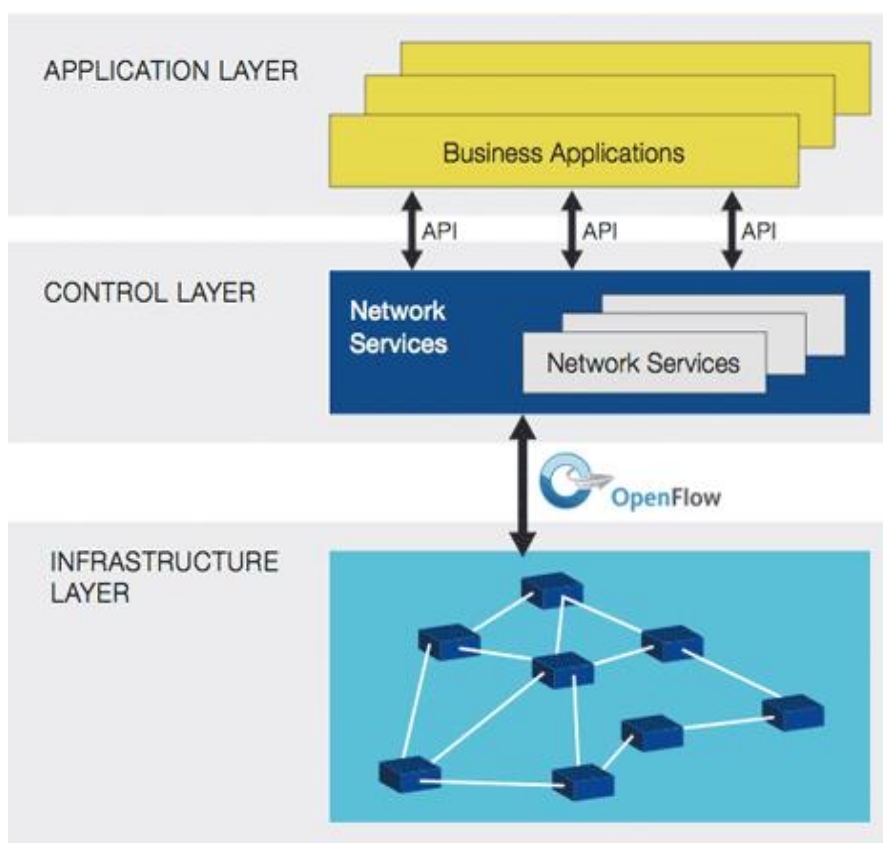


Figure 2. 3: Architecture standard du SDN

2.3.1. La couche infrastructure :

La couche d'infrastructure représente l'ensemble des ressources matérielles réseau (Switch, Routeur, Pare-feu, ...) et les connexions entre elles. Les équipements



contenant les plans de données du réseau reçoivent des informations du contrôleur sur l'endroit où déplacer les données.

2.3.2. La couche contrôle

Elle représente le logiciel contrôleur centralisé qui agit comme cerveau du réseau défini par logiciel. Ce contrôleur réside sur un serveur et gère les règles et le flux de trafic sur le réseau. Le contrôleur utilise les informations provenant des applications pour décider de l'acheminement d'un paquet de données.

2.3.3. La couche application

La couche applicative contient les applications ou fonctions réseaux typiques utilisées par les organisations. Ces applications ou fonctions communiquent des demandes de ressources ou des informations sur l'ensemble du réseau.

La couche d'application couvre un éventail d'applications pour répondre aux différentes demandes des clients telles que l'automatisation du réseau, la flexibilité et la programmabilité. Certains des domaines des applications SDN incluent l'ingénierie du trafic, la virtualisation du réseau, la surveillance et l'analyse du réseau, la découverte de services réseau et le contrôle d'accès.

2.3.4. Les interfaces de communication

Il existe principalement trois types d'interfaces ou API permettant au contrôleur d'interagir avec les autres couches du réseau : interfaces Sud, Nord et Est/Ouest.

Elles se voient attribuer la notation Nord/Sud/Est/Ouest en fonction de la position de la couche avec laquelle communique le contrôleur dans la hiérarchie de l'architecture. Pour communiquer avec une couche basse, l'interface utilisée est une interface Sud. Inversement pour les couches hautes c'est une interface Nord. La communication avec d'autres contrôleurs se fait via les interfaces Est/Ouest.

2.3.4.1. Interfaces Sud

Ce sont les interfaces (Southbound) qui permettent le processus de communication entre le contrôleur et les switches/routeurs et autres éléments de la couche infrastructure réseau. C'est à travers cette interface et notamment le protocole OpenFlow dans le cas du standard Open SDN (Open Networking Foundation, 2012), que le contrôleur injecte les différentes politiques aux équipements, et récupère les



informations permettant aux applications de construire une vue globale du réseau (Goransson et al, 2016). [10]

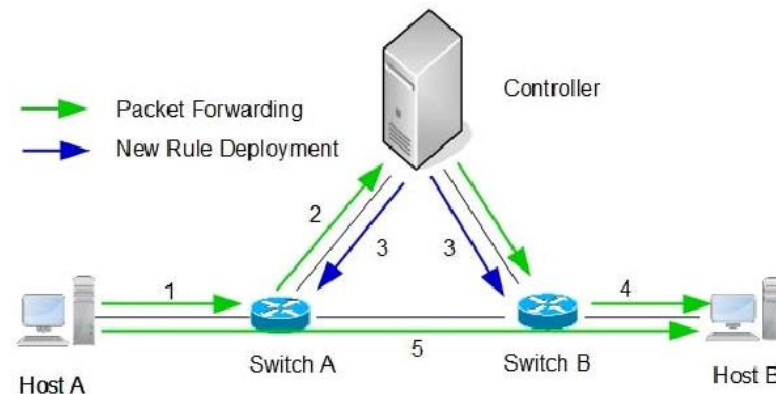


Figure 2. 4: Communication via interfaces Sud

Excepté le standard OpenFlow, plusieurs protocoles faisant office d'interface Sud ont été développés et sont utilisés dans certaines implémentations dites SDN à base d'API :

- ✚ Path Computation Element communication Protocol (PCEP) ;
- ✚ Interface to Routing System (I2RS) ;
- ✚ Simple Network Management Protocol (SNMP) ;
- ✚ NETwork CONFiguration protocol (NETCONF) ;
- ✚ BGP flow-spec ;

2.3.4.2. Interfaces Nord

Les interfaces Nord servent à programmer les équipements de transmission, en exploitant l'abstraction du réseau fourni par le plan de contrôle. Il est noté que contrairement à la Southbound API qui a été standardisé, l'interface nord reste encore une question ouverte. Bien que la nécessité d'une telle interface standardisée constitue un débat considérable au sein de l'industrie, l'avantage d'une API nord ouverte est aussi important, une API nord ouverte permette plus d'innovation et d'expérimentation. Plusieurs implémentations de cette interface existent, chacune de ces implémentations offre des fonctionnalités bien différents. Le RESTful considéré comme l'API nord le plus répandue dans les réseaux SDN.

2.3.4.3. Interfaces Est/Ouest

Les interfaces Est/Ouest sont des interfaces qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs pour synchroniser l'état



du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible. [12]

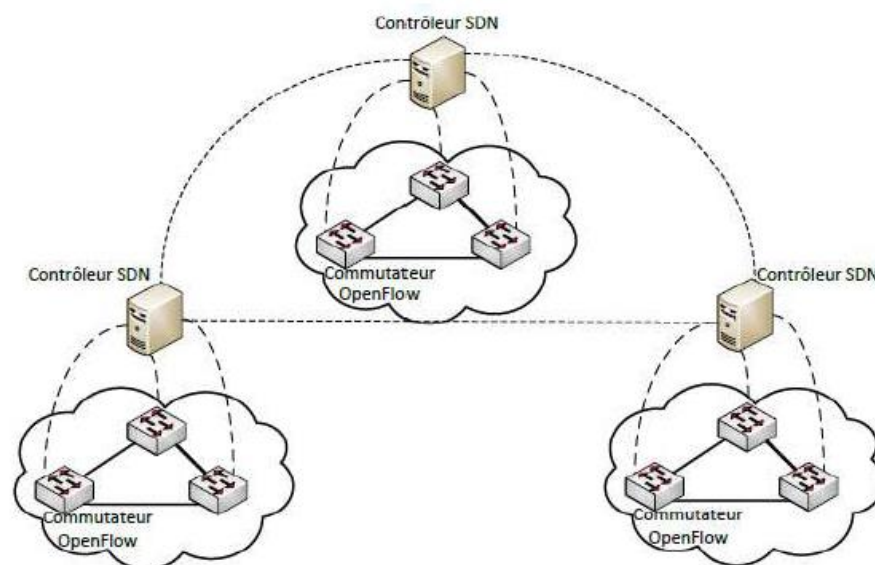


Figure 2. 5: Communication via interfaces Est/Ouest

2.2.5. Orchestrateur

L'orchestrateur vient répondre à la nécessité de pouvoir programmer le réseau de bout en bout. Si l'application est déployée dans un data center, les usagers peuvent être connectés en Wifi, connecté au LAN, derrière le WAN et des firewalls ou autres dispositifs réseau, ceux-ci sont souvent des domaines indépendants formant une chaîne d'exécution, ce qui rend la tâche pratiquement impossible pour le contrôleur (Durand, 2015). L'orchestrateur de réseau définie par logiciel (SDN) programme les comportements automatisés dans un réseau en s'appuyant sur les contrôleurs pour prendre en charge les applications et les services. [12]

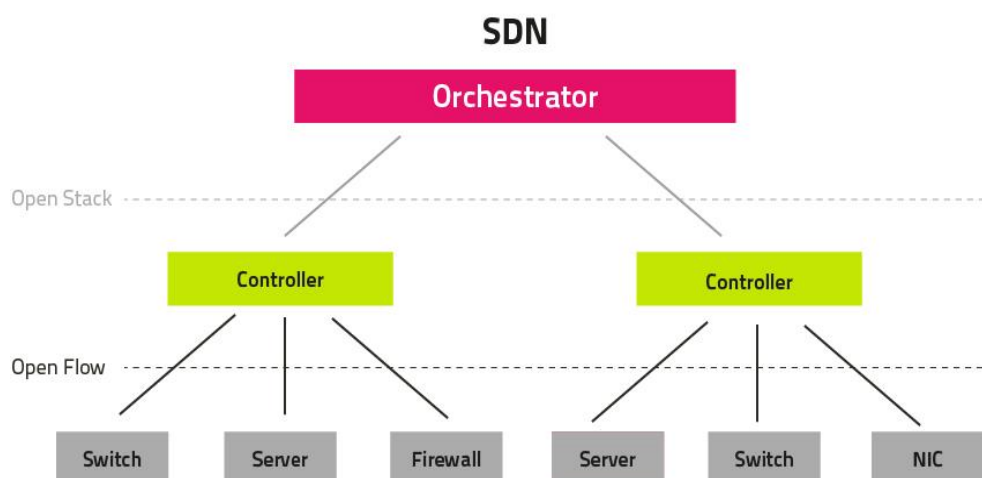


Figure 2. 6: Architecture du SDN avec Orchestrateur



2.4. Principe de fonctionnement du SDN

Le principe de fonctionnement du SDN, comme pour tout composant virtualiser, consiste à découpler le logiciel du matériel. Le SDN sépare les deux plans des périphériques réseau, en déplaçant le plan de contrôle qui détermine où envoyer le trafic vers le logiciel, et en laissant le plan de données qui transmet effectivement le trafic dans le matériel. Ainsi, les administrateurs réseau qui utilisent le Software-Defined Network peuvent programmer et contrôler le réseau dans son intégralité depuis une console unique, au lieu de procéder terminal par terminal.

Les trois éléments constituant l'architecture du SDN peuvent être situés sur un même site ou sur des sites physiques différents. En réalité, les périphériques de réseau physique ou virtuel font circuler les données sur le réseau. Dans certains cas, des commutateurs virtuels, qui peuvent être intégrés dans le logiciel ou le matériel, reprennent en charge les commutateurs physiques et consolident leurs fonctions au sein d'un seul commutateur intelligent. Le commutateur vérifie l'intégrité des paquets de données et des machines virtuelles de destination, et déplace les paquets. [13]

2.5. Protocole OpenFlow

2.5.1. Définition d'OpenFlow

OpenFlow est le protocole utilisé pour la communication entre la couche transmission et la couche de contrôle dans architecture Software-Defined Network (SDN). Il a été initialement proposé et implémenté par l'université de Stanford, et standardisé par la suite par l'ONF, sa dernière version est 1.5.1. Cependant, la version 1.6 est disponible depuis septembre 2016, mais accessible uniquement aux membres de l'ONF. Ce protocole est constitué d'instructions qui permettent de programmer le plan de contrôle d'un équipement réseau.

Le protocole prend en compte les fonctions communes supportées par les différentes tables des switchs Ethernet et routeurs conçus par les constructeurs, ce qui permet l'utilisation des tables de flux par tous les dispositifs réseau. OpenFlow est utilisé comme une interface sud dans les réseaux définies par architectures SDN (Canceres, 2016).



2.5.2. Architecture OpenFlow

L'architecture OpenFlow est l'implémentation réelle des réseaux SDN, cette architecture est basée principalement sur trois composantes dont le plan de données (switches OpenFlow), le plan de contrôle (contrôleurs OpenFlow) et une chaîne sécurisée qui permet aux commutateurs de se connecter au plan de contrôle. [4]

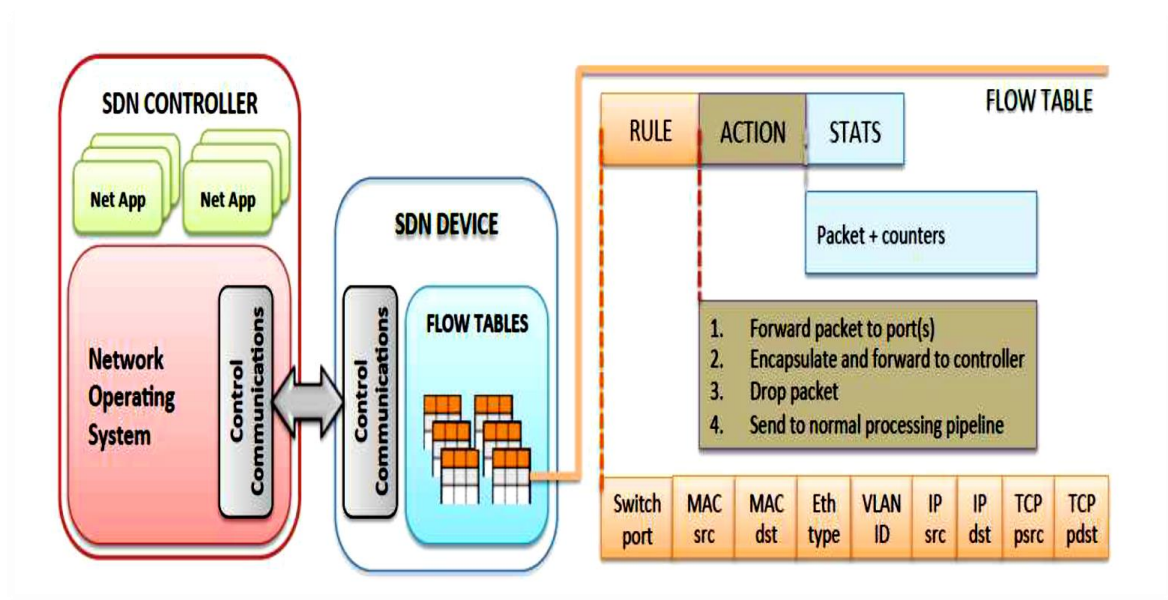


Figure 2. 7: Architecture OpenFlow (Kreutz et al, 2015)

2.5.2.1. Tables de flux

Un commutateur OpenFlow associe les paquets à une ou plusieurs tables de flux. Une table de flux contient des entrées de flux, et les paquets sont mis en correspondance sur la base de la priorité correspondante des entrées de flux.

Une entrée de flux est composée de :

- ✚ Champs de correspondance (Match fields) ;
- ✚ Compteurs ;
- ✚ Actions ;



Figure 2. 8: Entrée de flux

a. Champs de correspondance

Les champs de correspondance ou règles de correspondance de l'entrée de flux contiennent le port d'entrée, les en-têtes de paquet et les métadonnées liées au paquet.



Ils sont utilisés pour l'identification des flux. Cette identification peut se faire à base d'une multitude d'informations contenues dans l'entête du paquet, allant de la couche 1 à la couche 4 du modèle ISO. La figure suivante montre quelques-uns des champs proposés par le protocole, et depuis les versions les plus récentes d'autres possibilités d'identification ont été rajoutés comme les champs IPv6 et les étiquettes MPLS (Benamrane, 2017).

In port	VLAN ID	L2			L3			L4		and mask
		SA	DA	Type	SA	DA	Prot	Src	Dst	

Figure 2. 9: Champs de correspondance OpenFlow

Ingress Port : port d'entrée sur lequel est reçu le paquet.

Source Address (SA L2) : adresse Ethernet source (48).

Destination address (DA L2) : adresse Ethernet destination (48).

Ethernet frame type (Type) : Type de trame Ethernet : Ethernet II, LLC ou NSAP.

VLAN ID : Identificateur de VLAN dans l'en-tête VLAN si présent.

IPv4 source address (SA L3) : adresse IPv4 source du paquet (32).

IPv4 destination address (DA L3) : adresse IPv4 destination du paquet (32).

IP protocol (Prot) : Protocole contenu dans IP, exemple OSPF, TCP, UDP, etc.

Transport source port /ICMP type (Src) : Port source (couche transport).

Transport destination port /ICMP code (Dst) : Port destination (couche transport).

Mask : Masque de sous réseau.

b. Compteurs

Les compteurs sont réservés à la collecte des statistiques de flux. Ils enregistrent le nombre de paquets et d'octets reçus de chaque flux, et le temps écoulé depuis le dernier transfert de flux. Les compteurs peuvent être maintenus pour chaque table de flux entrée de flux, port, file d'attente. La liste ci-dessus décrit les compteurs :

- Par Flow Table :

Reference Count (active entries) : Nombre d'entrées actives dans la table.

Packet Lookups : Nombre de paquets soumis à la table.



Packet Matches : Nombre de paquets pour lesquels une des entrées de la table a pu s'appliquer.

- **Par Flow entry :**

Received Packets : Nombre de paquets reçus par l'entrée pour lesquels la recherche de correspondance a réussi.

Received Bytes : Nombre d'octets de paquets reçus par l'entrée pour lesquels la recherche de correspondance a réussi.

Duration (seconds) : Durée en secondes pendant laquelle l'entrée a été active.

Duration (nanoseconds) : Durée en nanosecondes pendant laquelle l'entrée a été active au-delà de la durée en secondes.

- **Par Port :**

Received Packets : Nombre de paquets reçus sur ce port.

Transmitted Packets : Nombre de paquets transmis par ce port.

Received Bytes : Nombre d'octets reçus par ce port.

Transmitted Bytes : Nombre d'octets transmis par ce port.

Receive Drops : Nombre de paquets reçus et rejetés par ce port.

Transmit Drops : Nombre de paquet rejetés en transmission.

Receive Errors : Nombre de paquets reçus en erreur.

Transmit Errors : Nombre de paquet transmis en erreur.

Receive Frame Alignment Errors : Nombre de paquet reçus avec erreur d'alignement.

Receive Overrun Errors : Nombre de paquets reçus et rejetés faute de mémoire dans les files d'attente en entrée du port.

Receive CRC Errors : Nombre de paquets reçus avec un CRC erroné.

Collisions : Nombre de paquets rejetés à la suite d'une collision.

- **Par Queue (file d'attente) :**

Transmit Packets : Nombre de paquets transmis sur cette file d'attente.

Transmit Bytes : Nombre d'octets transmis sur cette file d'attente

Transmit Overrun Errors : Nombre de paquet transmis sur cette file d'attente et rejetés faute de mémoire sur la file.



Lorsqu'un compteur arrive à sa valeur maximum, il repasse à 0 sans autre indication. Si un compteur n'est pas disponible, sa valeur doit être positionnée à -1.

c. Actions

À chaque entrée de la table de flux est associé un ensemble d'actions à exécuter sur les paquets, avant de les envoyer vers un port de sortie. Les actions doivent être exécutées dans le même ordre dans lequel elles sont présentées dans la table. Si une entrée ne contient aucune action, le paquet qui lui correspond est supprimé. Un switch OpenFlow peut ne pas supporter tous les types d'actions. Cependant il devrait supporter l'acheminement de paquets aux ports physiques ainsi qu'aux ports virtuels suivants :

- **ALL** : Est utilisé pour inonder un paquet sur tous les ports du commutateur à l'exception du port d'entrée ; ceci permet d'offrir une capacité broadcast rudimentaire au commutateur OpenFlow.
- **CONTROLLER** : Pour encapsuler le paquet et l'envoyer au contrôleur (Message PACKET_IN du commutateur au contrôleur). Cette action peut être indiquée dans l'entrée de flux. Aussi, si aucune entrée ne correspond au paquet à acheminer, le commutateur émet par défaut ce paquet au contrôleur.
- **LOCAL** : Pour l'envoi le paquet à la CPU locale (contrôleur local dans le commutateur).

Un commutateur dispose d'un contrôleur local vers lequel le paquet peut être envoyé pour la décision de son acheminement.

- **TABLE** : S'applique uniquement aux paquets que le contrôleur émet au commutateur.

Ces paquets arrivent via le message PACKET_OUT du contrôleur, incluant la liste d'actions. Cette liste d'actions va généralement contenir une action de sortie, qui spécifie un numéro de port. Le contrôleur peut vouloir directement spécifier le port de sortie pour ce paquet de données, ou vouloir que le port soit déterminé par le



traitement de paquet OpenFlow normal ; dans ce dernier cas, il stipule TABLE comme port de sortie.

- **IN PORT** : Envoyer le paquet sur le port d'entrée. Cette action est nécessaire lorsque l'émetteur et le récepteur du paquet sont joignables via le même port du commutateur (Exemple d'un émetteur et récepteur présents sur le même hotspot Wi-Fi). [4]

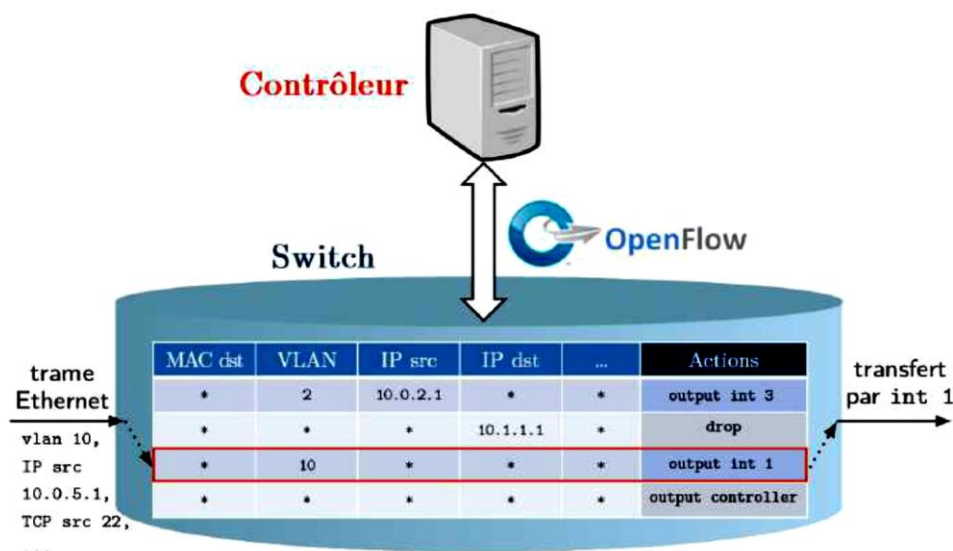


Figure 2. 10: Modèle de flux OpenFlow

2.5.3. Messages OpenFlow

Les messages OpenFlow sont envoyés et reçus entre le contrôleur et les chemins de données (instances ou appareils OpenFlow). Ces messages entre le contrôleur et le switch sont transmis via un canal sécurisé via une connexion TLS sur TCP. Le switch initie la connexion TLS lorsqu'il connaît l'adresse IP du contrôleur. Chaque message entre le switch et le contrôleur commence avec l'en-tête OpenFlow. Cet en-tête spécifie le numéro de OpenFlow, le type de message, la longueur de message, et l'identificateur de transaction du message. Il existe trois catégories de message : symetric, controller-switch et async.

2.5.3.1. Messages symétriques

Les messages symétriques peuvent être émis indifféremment par le contrôleur ou le switch sans avoir été sollicité par l'autre entité. Ces messages sont :

- **HELLO** : Ces messages sont échangés pour déterminer le numéro de version OpenFlow le plus élevé supporté par le switch et le contrôleur après



l'établissement du canal sécurisé. Le protocole spécifie que la plus petite des deux versions doit être utilisée pour la communication contrôleur - commutateur sur le canal sécurisé.

- **ECHO** : Ce sont des messages qui sont utilisés par n'importe quelle entité (contrôleur, commutateur) pendant le fonctionnement du canal pour s'assurer que la connexion est toujours en vie et afin de mesurer la latence ainsi que le débit courant de la connexion. Chaque message ECHO_REQUEST doit être acquitté par un message ECHO_REPLY.
- **VENDOR** : Ces messages sont disponibles pour des améliorations et pour une expérimentation spécifique au vendeur. [2]

2.5.3.2. Messages Contrôleur-switch

Les messages contrôleur-commutateur représentent la catégorie la plus importante de messages OpenFlow. Ils peuvent être représentés en cinq sous-catégories : *switch configuration*, *command from controller*, *statistics*, *queue configuration*, et *barrier*.

- **Switch configuration** : Ils consistent en un message unidirectionnel et deux paires de messages requête-réponse. Le contrôleur émet le message unidirectionnel SET_CONFIG afin de positionner les paramètres de configuration du commutateur. Le contrôleur utilise la paire de message FEATURES_REQUEST et FEATURES_REPLY afin d'interroger le commutateur au sujet des fonctionnalités (notamment optionnelles) qu'il supporte. La paire de message GET_CONFIG_REQUEST et GET_CONFIG_REPLY est utilisée afin d'obtenir la configuration du commutateur.
- **Command from controller** : Les messages sont au nombre de 3. PACKET_OUT est analogue à PACKET_IN mentionné précédemment. Le contrôleur utilise PACKET_OUT afin d'émettre des paquets de données au commutateur pour leur acheminement via le plan usager (Plan de données). Le contrôleur modifie les entrées de flux existantes dans le commutateur via le message



FLOW_MOD. Le PORT_MOD est utilisé pour modifier l'état d'un port OpenFlow.

- **Statistics** : Des statistiques sont obtenues du commutateur par le contrôleur via la paire de message STATS_REQUEST et STATS_REPLY.
- **Queue configuration** : La configuration de files d'attente associées à un port n'est pas spécifiée par OpenFlow.

Un autre protocole de configuration doit être utilisé pour ce faire. Toutefois le contrôleur peut interroger le switch via QUEUE_GET_CONFIG_REQUEST acquitté par QUEUE_GET_CONFIG_REPLY pour apprendre quelle est la configuration de files d'attente associées à un port afin de pourvoir acheminer des paquets sur des file d'attente spécifiques et ainsi fournir à ces paquets un niveau de QoS désiré.

- **BARRIER_REQUEST** : Ce message est utilisé par le contrôleur pour s'assurer que tous les messages OpenFlow émis par le contrôleur et qui ont précédé cette requête ont été reçus et traités par le switch. La confirmation est retournée par le switch via la réponse BARRIER_REPLY. [1]

2.5.3.3. Messages asynchrones

Des messages asynchrones sont envoyés à la fois par le contrôleur et le commutateur lorsqu'il y a un changement d'état dans le système. Comme les messages symétriques, les messages asynchrones doivent également être envoyés sans aucune sollicitation entre le commutateur et le contrôleur. Le commutateur doit pouvoir envoyer les messages asynchrones suivants au contrôleur :

- **Packet-in** : Pour tous les paquets qui n'ont pas d'entrée de flux correspondante ou si un paquet correspond à une entrée avec une action d'envoi au contrôleur, un message de paquet entrant est envoyé au contrôleur. Il transfère le contrôle du paquet au contrôleur.
- **Flow-removed** : Ce message informe le contrôleur de la suppression d'une entrée de flux d'une table de flux. Le message de modification de flux



spécifie également si le commutateur doit envoyer un message de suppression de flux au contrôleur lorsque le flux expire.

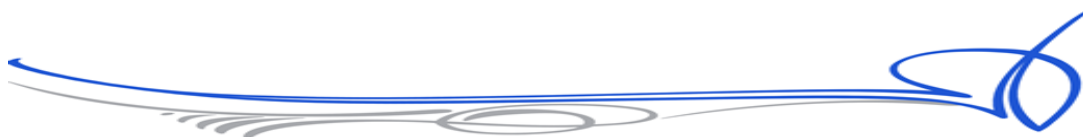
- **Port-status** : Il informe le contrôleur d'un changement sur le port. Ces événements incluent des changements dans l'état du port (par exemple, désactivé par l'utilisateur) ou un changement dans l'état du port tel que spécifié par 802.1D (Spanning Tree). Le commutateur est censé envoyer l'état du port au contrôleur via les messages de mise à jour du port.
- **Table-status** : Ce message permet au switch d'informer le contrôleur du changement d'état de la table de flux.
- **Controller-role status** : C'est un message par lequel le commutateur informe le contrôleur du changement de rôle.
- **Request-forward** : C'est un message de demande de transfert. Il permet au commutateur de transférer les messages de demande d'un contrôleur à d'autres contrôleurs. [4]

2.6. Les Commutateurs SDN

Un commutateur SDN est un programme logiciel ou un périphérique matériel qui transfère des paquets dans un environnement SDN. Il doit prendre en charge les protocoles SDN. Le protocole SDN le plus ancien et le plus connu est OpenFlow. Par conséquent, un commutateur SDN est également souvent appelé commutateur OpenFlow.

2.6.1. Commutateur SDN logiciel

Le plus souvent utilisé dans les environnements virtuels, un commutateur SDN logiciel est un programme qui implémente les tables de flux, les entrées et les champs de correspondance dans ses structures de données. Ce commutateur est plus lent et moins efficace car il ne bénéficie pas d'une accélération hardware, mais présente l'avantage d'être plus flexible, plus riche dans les actions disponibles, et supporte un nombre d'entrées beaucoup plus important. Ce type de switch permet de faire des économies du fait de sa simplicité de mise en place. Les principaux fabricants spécialisés dans la conception de switchs SDN logiciels sont : Nicira pour Open vSwitch et Big Switch pour Indigo Virtual Switch. [2]



2.6.2. Commutateur SDN matériel

Ce modèle de commutateur offre de meilleures performances et peut être utilisé dans les data center et au cœur du réseau. Les commutateurs matériels permettent d'obtenir jusqu'à 80 % de retard en moins et 100 % de probabilité de perte de paquets en moins par rapport aux commutateurs logiciels. Les commutateurs matériels impliquent un contrôleur SDN pour la prise de décision et les gains en termes de délais de transfert sont faibles. Pour transcrire les différentes entrées de flux et leurs composantes dans ces commutateurs l'utilisation de hardware spécialisé a été opté. Ces commutateurs intègrent une table CAM (Content-Addressable Memories) pour le traitement des trames et assurent le traitement de la couche 3 avec des TCAMs (Ternary Content-addressable Memories). [1]

2.7. Quelques cas d'utilisation de réseaux définis par logiciels

Le concept des réseaux définis par les logiciels (Software-Defined Network ou SDN) est apparu après l'avènement de l'internet et les nouvelles technologies de l'information pour répondre à de nombreux besoins et apporter des solutions aux difficultés rencontrées dans des cas d'utilisation très variés.

2.7.1. Cas du Big Data

Le Big Data est défini comme étant des ensembles de données très variées, arrivant dans des volumes de plus en plus importants et avec une vitesse plus élevée. Ces ensembles de données sont si volumineux qu'un logiciel de traitement de données traditionnel ne peut tout simplement pas les gérer.

La majorité des données constitutives du Big Data sont des données non structurées qui, contrairement aux données structurées qui sont parfaitement adaptées aux schémas de base de données classiques, sont beaucoup plus difficiles à gérer. Les calculs analytiques effectués sur le Big Data sont basés principalement sur les opérations de fractionnement des données en plusieurs nœuds de serveur, d'analyse de chaque bloc de données en parallèle et de fusion des résultats. De ce fait un réseau intelligent évoluant de manière adéquate à chaque phase du calcul répondant aux exigences de bande passante du transfert de données lors des phases de fractionnement et de fusion, et améliorant la vitesse de traitement est requis.



Le SDN offre donc d'énormes possibilités pour construire le réseau adaptatif intelligent requis pour l'analyse du Big Data. En raison du découplage du plan de contrôle et du plan de données, le SDN propose une interface programmatique bien définie qui permet à l'intelligence logicielle de programmer des réseaux hautement personnalisables, extensibles et agiles, afin de répondre aux exigences du Big Data à la demande. Le SDN permet de dimensionner le réseau à la demande de manière à ce que les machines virtuelles de calcul communiquent entre elles de façon optimale. Le principal obstacle auquel se heurte le Big Data, application massivement parallèle comme des temps de traitement trop longs désormais pris en compte. Les vitesses de traitement lentes sont dues à la durée d'attente des volumes massifs de données dans une application Big Data lors des opérations de fragmentation-regroupement avant de pouvoir commencer le traitement. Grâce au SDN, le réseau peut créer des voies de communication sécurisées à la demande et réaliser une extension de capacité lors des opérations de fragmentation-regroupement, réduisant ainsi de façon significative le temps d'attente et par conséquent le temps de traitement. [2]

Cette intelligence logicielle, qui consiste ni plus ni moins en la compréhension de ce que le réseau peut apporter à l'application, peut être mise à profit de manière très précise et efficace pour les applications de Big Data. Cela s'explique de deux façons, la première est l'existence de modèles de calcul et de communication bien définis, tels que le modèle SplitMerge ou MapReduce de Hadoop et la seconde est l'existence d'une structure de gestion centralisée qui permet d'exploiter l'information au niveau applicatif, comme Hadoop Scheduler ou Hbase Master.

Avec l'aide du contrôleur SDN, qui a une vue globale du réseau sous-jacent (état, utilisation, etc.), l'intelligence logicielle peut traduire les besoins de l'application de façon précise en programmant le réseau à la demande.

2.7.2. Cas du Data Center

Un data center, ou centre de données en français, désigne un lieu physique où sont regroupés différents équipements informatiques, tels que des ordinateurs, des serveurs, etc. Sa fonction principale consiste à stocker des informations utiles au bon fonctionnement d'une organisation.



Le prolongement du concept SDN au cœur des centres de données sous la dénomination SDDC (Software-Defined Data Center) est devenu rapidement une évidence ces dernières années, avec l'exploitation commerciale qui en résulte.

Dans le modèle SDDC, toute l'infrastructure du centre de données est virtualisée pour permettre à la fois d'automatiser certaines fonctions et procurer des ressources ou services à la demande (sur le principe du mode « as-a-Service » popularisé dans l'univers du cloud).

Le pilotage de l'ensemble est assuré par un système d'orchestration ou méta-contrôleur central qui fixe et applique des règles, comme dans le modèle OpenFlow.

Le niveau de contrôle dans le SDDC est dissocié de celui des applications et du traitement des données, selon un découpage logique avec des « pools » virtuels et non plus des entités d'équipements physiques. Le dispositif de contrôle ainsi centralisé, est personnalisable et programmable, donc en grande partie automatisable.

Dans un centre de données défini par logiciel, tous les éléments de l'infrastructure réseau, unités de calculs (CPU/serveurs avec leurs VM), unités de stockage des données (SDS, Software-Defined Storage), et dispositifs de sécurité (application automatique des règles de sécurité, mise en quarantaine de terminaux suspects, etc.) sont virtualisés et définis par logiciel. Ils deviennent accessibles en tant que services, disponibles à la demande.

Un centre de données défini par logiciel partage une grande partie des concepts implémentés dans un Cloud. Vu qu'il est censé à terme englober toute l'infrastructure IT, certains considèrent que le SDDC peut constituer la plateforme globale, virtuelle de l'ensemble des Clouds (privés, publics et hybrides). Ce qui suppose qu'il puisse être étendu à plusieurs sites, donc plusieurs centres de données interdépendants. Cela rejoint les concepts d'architecture mis en application par les géants de la haute technologie tels qu'Amazon (AWS), Google, IBM ou Microsoft.

Enfin, le SDDC vise à réduire les dépenses d'investissement mais surtout les dépenses opérationnelles, tout en améliorant l'efficacité, l'agilité, le contrôle et la flexibilité de l'ensemble avec une sécurité renforcée grâce à un niveau de contrôle plus centralisée.



2.7.3. Cas de l'IoT

Défini comme étant l'interconnexion entre l'Internet et des objets, des lieux et des environnements physiques, l'Internet des objets (en anglais (the) Internet of Things ou *IoT*) croise les réseaux à définition logicielle au carrefour de l'épuisement des VPN, des défis de la disponibilité et des ressources réseau limitées. Le SDN contribuera à l'expansion des appareils IoT, renforcera l'efficacité du partage des ressources réseau et améliorera les accords de niveau de service (SLA, Service-Level Agreements) de l'IoT.

L'impact que les appareils de l'Internet des objets auront sur les réseaux commence tout juste à se faire sentir. Quand l'Internet des objets arrivera à maturité, ses besoins en matière d'interconnexion inverseront les tendances existantes, c'est-à-dire de faibles volumes de données avec un nombre très élevé de connexions. La souplesse de configuration qu'offre le SDN peut permettre aux opérateurs réseau comme aux entreprises d'allouer plus facilement des ressources pour supporter cette évolution. La virtualisation des fonctions réseau (NFV, Network Functions Virtualization) renforce l'évolutivité des fonctions de sécurité et de traitement nécessaires pour faire face à cette nouvelle vague de données.

Pour finir, le déploiement du SDN pour les appareils de l'Internet des Objets permettrait de partager les ressources de manière plus efficace et fiable tout en palliant l'épuisement des VPN. [10]

2.7.4. Cas d'un réseau d'entreprise

Les réseaux définis par logiciel ou SDN sont, depuis 2017, devenus la technologie à adopter. Même si, pour être clair, l'intérêt que voient les entreprises dans cette technologie ne réside pas dans le déploiement du SDN en soi, mais plutôt dans les capacités de transformation que rend possible le SDN.

La transformation numérique et le niveau d'exigence des clients obligent les entreprises à faire évoluer leurs business modèles faisant ainsi peser plus de pression sur la technologie réseau héritée. Le réseau SDN (Software-Defined Network), avec sa capacité à orchestrer les réseaux de manière intelligente pour permettre aux applications d'avoir accès aux ressources à la demande selon le modèle de facturation à l'utilisation, a le



potentiel d'offrir le niveau d'agilité dont les entreprises ont besoin pour survivre dans le monde numérique.

De nombreuses entreprises cherchent actuellement à accélérer l'adoption des réseaux SDN. Une étude de Verizon, menée par Longitude, société du groupe Financial Times, auprès de 165 dirigeants IT seniors de grandes multinationales, met en évidence les raisons de ce sentiment d'urgence.

Le SDN est largement considéré comme une clé de transformation réseau et de croissance de l'activité. Le SDN permet aux entreprises de construire un réseau plus sophistiqué sans ajouter de complexité, si bien que les entreprises peuvent mieux gérer leur capacité, déployer des logiciels plus facilement et administrer leurs réseaux plus efficacement. Autrement dit, cette technologie aide les entreprises à faire face aux aléas du marché, et ainsi à répondre de manière plus réactive aux besoins des utilisateurs et clients et à mieux saisir les opportunités de marché.

2.7.5. Cas du réseau mobile 5G

Parmi les différentes architectures et technologies qui structurent la 5ème génération de téléphonie mobile (5G), le SDN (Software-Defined Network) constitue une brique essentielle. Il est l'heure de recentrer les services sur l'utilisateur, comme le réalisent déjà certains opérateurs télécoms sur le fixe en utilisant SDN et NFV (Network Functions Virtualization) pour proposer des offres « on demand ».

Avec les très hauts débits de la 5G, l'infrastructure pourra supporter des milliards d'appareils. Mais les modèles de trafic seront moins prévisibles. Il faut donc pousser le niveau d'intelligence de l'infrastructure. Le SDN s'avère être une technique très prometteuse pour ces réseaux 5G, soutenue par la virtualisation des fonctions réseau ou NFV. Car, avec la 5G, l'administration et l'exploitation du réseau seront conditionnées par les applications en cours d'utilisation.

La 5G utilise le concept de tranche réseau (Network Slicing) pour gérer ses différents types services. Le Network Slicing est une architecture qui permet le découpage en plusieurs tranches de réseaux logiques virtualisés et indépendants sur la même infrastructure de réseau physique. Le SDN est la clé permettant d'allouer les ressources du réseau à la demande. Cette flexibilité favorise la création des tranches



de réseau spécifiques à chaque usage. C'est le principe de virtualisation du cœur de réseau qui est fondamental dans la 5G.

Le SDN offrirait donc à la 5G une flexibilité unique pour développer des services innovants et ouvrir le réseau mobile à de nouveaux usages, tel que la voiture autonome, l'*e-health*, l'industrie 4.0, ou bien la réalité augmentée. Tous ces services ont des besoins particuliers et il faut un réseau capable de connecter toutes ces ressources, qui seront certainement virtuelles. [10]

2.8. Les Solutions SDN existantes

Avec l'évolution du SDN, beaucoup d'équipementiers de réseau et des particuliers proposent une multitude de solutions SDN. Parmi cette multitude, on distingue des solutions propriétaires donc payantes et des solutions open sources ou gratuites. Loin de dresser une liste exhaustive de ses différentes solutions, nous passons en revue les principales.

2.8.1. Présentation de quelques solutions SDN propriétaires

2.8.1.1. Contrail Network

Le Contrail Network de Juniper Networks est un produit d'automatisation de réseau cloud ouvert qui utilise la technologie de réseau défini par logiciel (SDN) pour orchestrer la création de réseaux virtuels à haute évolutivité. Il fournit une politique et un contrôle réseau dynamique de bout en bout pour n'importe quelle charge de travail et n'importe quel déploiement, à partir d'une interface utilisateur unique.

Après l'achat de Contrail Systems en 2012, Juniper l'a ouvert un an plus tard sous le nom d'OpenContrail puis a transféré le projet à la fondation Linux en 2017. Et l'année suivante il a été renommé Tungsten Fabric.

Bien que conçu pour le réseau cloud, la capacité de mise en réseau définie par logiciel (SDN) du projet de virtualisation de réseau open source permet à Tungsten Fabric d'être déployé dans n'importe quel environnement.

La plateforme Contrail est architecturée autour de deux principaux composants :

- ✓ Le contrôleur Contrail ;
- ✓ Le vRouter Contrail ;



Le contrôleur est le plan de contrôle et de gestion qui gère et configure le vRouter, en collectant et en présentant les analyses.

Le Contrail vRouter est le plan de transfert qui fournit des services de couche 2 et de couche 3, ainsi que des fonctionnalités de pare-feu distribué, tout en implémentant des politiques entre les réseaux virtuels.

Le fonctionnement de Contrail Network élimine le besoin de supprimer et de remplacer quoique ce soit en prenant en charge de nombreux protocoles basés sur des normes, permettant l'interopérabilité dans une infrastructure physique multifournisseur. Il faut juste, pour Contrail Network, que le réseau sous-jacent soit en connectivité IP. A travers cette connectivité la virtualisation de réseau robuste exploite la norme L3VPN pour les superpositions IP L3, la norme EVPN pour les superpositions L2 et une multitude de normes d'encapsulation de données telles que MPLS sur protocole d'encapsulation de routage générique (MPLSoGRE), MPLS sur datagramme utilisateur Protocol (MPLSoUDP), Virtual Extensible LAN (VXLAN), etc. Les services de fonctions réseau virtualisées ou physiques sont mis en un chaînage dynamique qui simplifie la création, le déploiement et la gestion de services réseau différenciés.

Pour répondre au besoin de programmabilité et d'automatisation une traduction, est réalisée, des flux de travail abstraits de haut niveau en règles spécifiques pour automatiser le provisionnement des charges de travail. Par exemple, il est possible de demander la connectivité de la machine virtuelle sans entrer dans les détails des éléments sous-jacents tels que les ports, les VLAN, les sous-réseaux, les commutateurs, les routeurs, etc. De plus, dans un modèle unifié pour la configuration, le fonctionnement et l'analyse sont exposés via les API REST, ainsi que des bibliothèques dans divers langages de programmation tels que Python, Go, Javascript et Java, pour n'en nommer que quelques-uns. Les informations, telles que les latences la gigue, d'analyse exposée via les API REST peut être visualisées pour simplifier les opérations et la prise de décision grâce à une planification proactive et à des diagnostics prédictifs. [13]



2.8.1.2. Cisco DNA Center

Cisco DNA Center est un logiciel central de gestion et d'automatisation. Il est utilisé comme contrôleur pour Cisco DNA (Digital Network Architecture). Il fait office de contrôleur réseau et un tableau de bord de gestion pour les accès SD (Software Defined), les réseaux basés sur l'intention et les réseaux traditionnels existants. Cisco DNA Center est la récente plate-forme de gestion de réseau de Cisco pour les réseaux d'entreprise.

Cisco DNA Center fournit un tableau de bord unique pour chaque tâche de gestion fondamentale afin de simplifier l'exploitation du réseau. Grâce à cette plate-forme, le service informatique peut réagir aux changements et aux défis plus rapidement et plus intelligemment.

Cisco DNA Center comporte quatre sections générales alignées sur les workflows informatiques décrivant son fonctionnement :

- ✓ La conception : permettant la configuration cohérente par périphérique et par site pouvant être placée dans une image dorée. Cette image peut être utilisée pour la provisionnement automatique de nouveaux périphériques réseau.
- ✓ La politique : intention de l'entreprise en stratégies de réseau (contrôle d'accès, le routage du trafic et la qualité de service), de manière cohérente sur l'ensemble de l'infrastructure filaire et sans fil. Cette politique est assurée par Cisco Software-Defined Access (SD-Access), Cisco Identity Services Engine (ISE) et Cisco AI Network Analytics et Cisco Group-Based Policy Analytics.
- ✓ Le provisionnement : une fois que les stratégies sont créées le provisionnement est une tâche simple (glisser-déposer). Les profils (appelés balises de groupe évolutives ou « SGT » en anglais Scalable Group Tags) dans la liste d'inventaire de Cisco DNA Center se voient attribuer une stratégie, et cette stratégie suivra toujours l'identité. Les nouveaux périphériques ajoutés au réseau sont attribués à un SGT en fonction de leur identité, ce qui facilite grandement les configurations de bureaux distants.
- ✓ L'assurance : l'utilisation de l'IA/ML permet à chaque point du réseau de devenir un capteur, envoyant une télémétrie en continu sur les



performances des applications et la connectivité des utilisateurs en temps réel. Le tableau de bord affiche de manière clair et simple la santé détaillée du réseau et signale les problèmes. Ensuite, la correction guidée automatise la résolution pour maintenir les performances du réseau à son niveau optimal, offrant ainsi une optimisation proactive de votre réseau avec moins de temps consacré aux tâches de dépannage. [15]

2.8.1.3. HPE VAN SDN

Virtual Application Networks (VAN) SDN de HP est un logiciel qui fournit un point de contrôle unifié dans un réseau SDN, simplifiant la gestion, le provisionnement et l'orchestration. Le contrôleur VAN SDN est le bloc de construction de l'écosystème HPE Open SDN (HPE SDN App Store et kit de développement logiciel SDN), permettant aux développeurs tiers de fournir des solutions innovantes pour lier dynamiquement les besoins de l'entreprise à l'infrastructure réseau.

Les composants du contrôleur HP VAN SDN sont décrits comme suit :

- ✓ Un gestionnaire d'applications du contrôleur ;
- ✓ Un journal d'audit des événements liés aux activités ;
- ✓ Un journal des alertes d'informations sur les événements qui affectent le fonctionnement du contrôleur ;
- ✓ Le Distributed Coordination Framework qui est l'une des fonctionnalités de haute disponibilité du contrôleur ;
- ✓ Les pilotes de périphériques modélisant les capacités des périphériques et fournissant des API pour interagir avec différents types de périphériques ;
- ✓ Un contrôleur OpenFlow (également appelé contrôleur principal) qui gère les connexions des appareils OpenFlow et fournit aux couches supérieures du logiciel les moyens d'interagir avec ces appareils.

HPE VAN SDN inclut un ensemble d'applications par défaut de service de réseau principal qui sont installées avec le contrôleur. Ces applications sont installées, mises à niveau, et désinstallées grâce au gestionnaire. Parmi les applications intégrées, le contrôleur utilise les applications Topology Manager, Topology Viewer, OpenFlow Link Discovery et OpenFlow Node Discovery pour la



découverte, la collecte et l'affichage des informations sur le réseau OpenFlow. Les informations collectées sont enregistrées par un journal d'alertes et un journal d'audit. Le journal d'alertes conserve les informations sur les événements affectant le fonctionnement du contrôleur et les interactions avec les appareils du réseau. Le journal d'audit, quant à lui se charge d'enregistrer les événements liés aux activités, aux opérations et aux changements de configuration.

Le contrôleur HPE VAN SDN fournit un cadre pour sauvegarder et restaurer son état dans un fichier de sauvegarde. Le fichier de sauvegarde peut être copié et stocké pour une utilisation ultérieure. Le fichier de sauvegarde stocké peut être téléchargé sur le contrôleur. Ce fichier répond en partie à la haute disponibilité du contrôleur fournit par le Framework coordinateur distribuée. [16]

2.8.1.4. Orion

Orion est une plate-forme distribuée de réseau défini par logiciel de Google. Elle est la seconde génération de contrôleur SDN de Google après Onix. Orion a été conçu autour d'une architecture modulaire de micro-services avec une base de données centrale de publication-abonnement pour permettre un système de contrôle de réseau distribué, mais étroitement couplé, défini par logiciel. Orion permet une gestion et un contrôle basés sur l'intention. Il est hautement évolutif et se prête aux hiérarchies de contrôle globales.

Les composants d'Orion sont :

- ✓ Le Core Orion, le domaine au sein duquel on trouve une interface Southbound OpenFlow, trois (3) gestionnaires (pour le flux, la topologie et la configuration) et une base d'information réseau (NIB) ;
- ✓ Le Framework d'application Orion qui est la base de toutes les applications d'Orion ;
- ✓ Une interface Northbound pour la communication avec les applications ;
- ✓ Le moteur de routage Orion, application chargée du contrôle de routage intra-domaine.

Pour l'évolutivité et l'isolation des pannes, l'architecture d'Orion divise le réseau en domaines où chaque domaine est une instance du contrôleur Orion. Le plan de données se compose de commutateurs SDN en bas. Orion utilise une interface OpenFlow pour le contrôle vers le plan de données. Le plan de données est



représenté sous forme de topologie (nœud, port, lien, interface, etc.) par le gestionnaire de topologie du Core Orion. La gestion flux dans le plan de données et vers le plan de contrôle est assurée par le gestionnaire de flux.

Le plan de contrôle est composé du Core Orion au centre et d'applications SDN en haut. Le Core Orion contrôle le réseau via des processus distribués. Pour communiquer avec les applications, le Core Orion utilise l'interface Northbound fournit par la NIB. La NIB (Network Information Base) est le magasin de données centralisées en mémoire. Les applications d'Orion sont placées dans un cadre garantissant des fonctionnalités requises pour le déploiement et la haute disponibilité. Ce cadre, fournit par le Framework, isole les bugs d'applications les uns des autres.

Pour finir, le routage intra-domaine du contrôleur Orion est assuré par l'application de routage RE (Routing Engine). Ce moteur de routage fournit des mécanismes de routage communs tels que le L3 MPF (Multi-Path Forwarding), l'équilibrage de charge et l'encapsulation de données. [17]

2.8.2. Présentation de quelques contrôleurs SDN open source

2.8.2.1. Open Network Operating System (ONOS)

Le projet ONOS (Open Network Operating System) est une communauté open source hébergée par The Linux Foundation. L'objectif du projet est de créer un système d'exploitation de réseau défini par logiciel (SDN) pour les fournisseurs de services de communication, conçu pour l'évolutivité, les hautes performances et la haute disponibilité.

ONOS prend en charge à la fois la configuration et le contrôle en temps réel du réseau, éliminant ainsi le besoin d'exécuter des protocoles de routage et de contrôle de commutation à l'intérieur de la structure du réseau.

- ✓ Un ensemble d'applications qui agissent comme un contrôleur SDN extensible, modulaire et distribué ;
- ✓ Une gestion, configuration et déploiement simplifiés de nouveaux logiciels, matériels et services ;
- ✓ Une architecture évolutive pour fournir la résilience et l'évolutivité requises pour répondre aux rigueurs des environnements de production des opérateurs.

Les éléments constitutifs de l'architecture d'ONOS sont :



- ✓ Noyau distribué (Distributed Core) ;
- ✓ API d'interface sud (Southbound Core API) ;
- ✓ API d'interface nord (Northbound Core API).

Même si le noyau ONOS est souvent décrit comme un bloc monolithique, il s'agit en réalité d'un assemblage de sous-systèmes individuels, chacun responsable du suivi de sa propre classe d'état du réseau. Ce diagramme offre un inventaire de haut niveau de ces sous-systèmes. La plupart d'entre eux fournissent des services qui peuvent être utilisés par d'autres sous-systèmes et qui forment collectivement l'API Northbound de base d'ONOS.

ONOS prend en charge une longue liste d'interfaces vers le sud, notamment OpenFlow, P4, NETCONF, TL1, SNMP, BGP, RESTCONF et PCEP. Vers le nord ONOS offre le plus grand ensemble d'interfaces vers le nord avec les API gRPC et RESTful.

Avec son interface Web, ONOS fournit une interface visuelle au contrôleur du système d'exploitation en réseau ouvert (ou groupe de contrôleurs).

ONOS met en œuvre un cadre basé sur l'intention intégré. En faisant abstraction d'un service réseau en un ensemble de critères auxquels un flux doit répondre, la génération de la configuration OpenFlow (ou P4) sous-jacente est gérée en interne, et le système client spécifiant uniquement ce que le résultat fonctionnel doit être. [14]

2.8.2.2. Floodlight

FloodLight est un contrôleur SDN développé par une communauté ouverte de développeurs, dont beaucoup de Big Switch Networks, qui utilise le protocole OpenFlow pour orchestrer les flux de trafic dans un environnement de réseau défini par logiciel (SDN).

Le contrôleur Floodlight est principalement avantageux pour les développeurs, car il leur offre la possibilité d'adapter facilement des logiciels et de développer des applications et est écrit en Java. Sont incluses des interfaces de programme d'application de transfert d'état représentatif (API REST) qui facilitent la programmation de l'interface avec le produit, et le site Web Floodlight propose des exemples de codage qui aident les développeurs à créer le produit.



Testé avec des commutateurs compatibles OpenFlow physiques et virtuels, le contrôleur Floodlight peut fonctionner dans une variété d'environnements et peut coïncider avec ce que les entreprises ont déjà à leur disposition.

La plateforme de Floodlight comprend divers modules, tels que :

- ✓ La gestion de la topologie ;
- ✓ La gestion des périphériques/stations d'extrémité ;
- ✓ Le calcul des chemins/routes ;
- ✓ L'infrastructure pour l'accès Web (gestion) ;
- ✓ Le magasin de compteurs (compteurs OpenFlow) ;
- ✓ Un système de stockage d'état.

Comme de nombreux autres contrôleurs, lorsqu'on exécute Floodlight, les opérations vers le nord et vers le sud du contrôleur deviennent actives. Autrement dit, quand Floodlight est exécuté, le contrôleur et l'ensemble des applications de module configurées s'exécutent également. Les API REST vers le nord exposées par tous les modules en cours d'exécution deviennent disponibles via le port REST spécifié. Toute application peut interagir (récupérer des informations et invoquer des services) avec le contrôleur en envoyant des commandes HTTP REST. D'autre part, en direction sud, le module fournisseur de Floodlight commencera à écouter sur le port TCP spécifié par OpenFlow les connexions des commutateurs OpenFlow. [24]

2.8.2.3 OpenDayLight

Le projet OpenDayLight est né du mouvement SDN, avec un accent clair sur la programmabilité du réseau. Il a été conçu dès le départ comme une base pour des solutions commerciales qui répondent à une variété de cas d'utilisation dans les environnements de réseau existants.

Les composantes d'OpenDayLight sont :

- ✓ Des plugins vers le sud ;
- ✓ Un ensemble services de base qui peuvent être utilisés au moyen de la couche d'abstraction de service basée sur OSGi pour aider les composants à entrer et sortir du contrôleur pendant que le contrôleur est en cours d'exécution ;
- ✓ Des interfaces vers le nord (par exemple REST/NETCONF) ;



Le cœur de la plate-forme OpenDaylight est la couche d'abstraction de service pilotée par le modèle (MD-SAL). Dans OpenDaylight, les périphériques réseaux sous-jacents et les applications réseau sont tous représentés sous forme d'objets, ou de modèles, dont les interactions sont traitées dans la SAL.

Le SAL est un mécanisme de mise en relation pour l'échange et l'adaptation de données entre des modèles de YANG représentant des dispositifs et d'applications réseau. Les modèles de YANG fournissent des descriptions généralisées des capacités d'un appareil ou d'une application sans exigence des détails de mise en œuvre spécifiques. En fonction du rôle au sein du SAL dans une interaction donnée, un modèle « producteur » implémente une API et fournit les données de l'API alors qu'un modèle « consommateur » utilise l'API et consomme les données de l'API. Les interfaces « Northbound » et « Southbound » fournissent une vue d'ingénieur réseau sur le SAL.

ODL a été conçue pour offrir aux utilisateurs en aval et aux fournisseurs de solutions une flexibilité maximale dans la construction d'un contrôleur adapté à leurs besoins. La modularité de la plate-forme ODL permet à n'importe qui dans l'écosystème ODL de profiter des services créés par d'autres, d'apporter leurs ajouts et de partager leur travail. ODL prend en charge un large ensemble de protocoles dans n'importe quelle plate-forme SDN - OpenFlow, OVSDB, NETCONF, BGP et bien d'autres améliorant la programmabilité des réseaux modernes et répondant à plus de besoins des utilisateurs.

Pour des raisons de sécurité, d'évolutivité, de stabilité et de performances ou « S3P » la communauté ODL fournit des améliorations continues à tous ses projets. Des jeux de test et d'intégration menés en continu fournissent aux développeurs des résultats en temps réel pour voir comment les changements affectent S3P (Sécurité, Scalability, Stability and Performance). La sécurité étant un domaine d'intérêt clé pour ODL, la plate-forme fournit un cadre pour l'authentification, l'autorisation et la comptabilité (AAA), ainsi que la découverte et la sécurisation automatiques des périphériques et contrôleurs réseau. [23]



2.8.2.4. Ryu controller

Ryu est un contrôleur de réseau défini par logiciel (SDN) ouvert basé sur des composants conçus pour augmenter l'agilité du réseau en facilitant la gestion et l'adaptation de la gestion du trafic. Il est entièrement implémenté en Python et pris en charge par les laboratoires de NTT.

Ryu fournit des composants logiciels avec des APIs bien définies qui permettent aux développeurs de créer facilement de nouvelles applications de gestion et de contrôle de réseau. Cette approche par composants aide les organisations à personnaliser les déploiements pour répondre à leurs besoins spécifiques. [2]

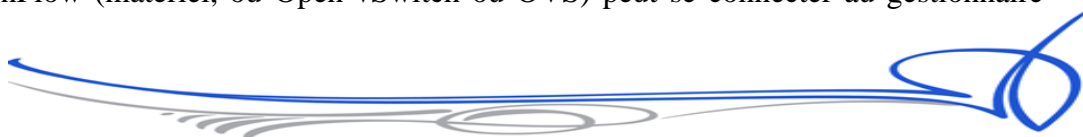
Comme tout contrôleur SDN, Ryu peut également créer et envoyer un message OpenFlow, écouter des événements asynchrones tels que `flow_removed`, et analyser et gérer les paquets entrants. La plateforme du contrôleur Ryu est principalement composée de :

- ✓ Une collection de bibliothèques ;
- ✓ Un composant de control OpenFlow ;
- ✓ Gestionnaire de processus de base ;
- ✓ Une interface Northbound ;
- ✓ Une interface Southbound OpenFlow ;
- ✓ Un ensemble d'applications fonctionnant de façon distribuée.

Ryu possède une collection impressionnante de bibliothèques, allant de la prise en charge de plusieurs protocoles vers le sud tels que OpenFlow, le protocole de configuration réseau (NETCONF) à diverses opérations de traitement de paquets réseau. Il est aussi inclus un plug-in Openstack Neutron qui prend en charge à la fois la superposition basée sur GRE et les configurations VLAN au niveau de la couche API.

L'un des composants clés de l'architecture Ryu est le contrôleur OpenFlow, qui est responsable de la gestion des commutateurs OpenFlow utilisés pour configurer les flux et gérer les événements. Ce contrôleur OpenFlow est l'une des sources d'événements internes de l'architecture Ryu.

Lors du démarrage de Ryu le gestionnaire de processus de base est le principale composant exécuté. Lorsqu'il est exécuté, Ryu écoute à l'adresse IP spécifiée (ex : 0.0.0.0) et au port spécifié (6633 par défaut) ensuite n'importe quel commutateur OpenFlow (matériel, ou Open vSwitch ou OVS) peut se connecter au gestionnaire



Ryu. Le composant de processus de base de l'architecture comprend la gestion des événements, la messagerie, la gestion de l'état en mémoire, etc. Les composants de messagerie Ryu peuvent être développés dans d'autres langages.

Ryu fonctionne en mode distribué avec plusieurs applications telles qu'un simple switch, un routeur, un pare-feu, un tunnel GRE, un VLAN, etc. Les applications Ryu sont des entités à un seul thread, qui implémentent diverses fonctionnalités. [18]



2.8.3. Tableau Comparatif des contrôleurs SDN Open Source

Contrôleurs SDN	Langage de prog	Doc	Modularité	GUI	Plateforme	OS supporté	Southbound API(s)	Northbound API(s)	Support OpenStack
ONOS	Java	Bonne	Elevée	Basé sur web	Distribué	Linux, Windows, MAC OS.	OpenFlow, P4, NETCONF, TL1, SNMP, BGP, PCEP RESTCONF.	API RESTful, gRPC API.	Java
OpenDaylight	Java	Bonne	Elevée	Basé sur Java Web/version	Distribué	Linux, Windows, MAC OS.	OpenFlow, P4, PCEP, NETCONF, SNMP, BGP, RESTCONF.	API RESTful, gRPC API, OSGi.	Java
Floodlight	Java	Complète	Moyenne	Basé sur Java/web	Distribué	Linux, Windows, MAC OS.	OpenFlow	API RESTful	Java, Python
Ryu	Python	Bonne	Moyenne	Basé sur Python	Framework	Linux, Windows, MAC OS.	OpenFlow, NETCONF, OF-Config	API RESTful	Python



Les contrôleurs SDN figurants dans le tableau ci-dessus sont tous gratuits, ainsi le choix de notre solution sera porté, en se basant sur l'étude comparative menée, sur celui qui répond le mieux à nos besoins.

2.8.4. Choix de la solution SDN à implémenter

L'étude comparative réalisée place les contrôleurs ONOS et OpenDayLight au coude à coude. Les deux solutions ont :

- ✓ Une documentation complète ;
- ✓ Une installation relativement très simple ;
- ✓ Une grande flexibilité à l'ajout de fonctionnalités externes (modularité) ;
- ✓ Un large choix de protocoles d'interfaces Southbound et Northbound.

Toutefois, ONOS et OpenDayLight restent deux solutions SDN adaptées à des infrastructures plus ou moins différentes.

ONOS est un choix approprié pour les opérateurs de télécommunications. En effet, ONOS prend en charge plusieurs API(s) nord et sud afin que les fournisseurs de services de communication n'aient pas à écrire leur propre protocole pour configurer leurs appareils. L'évolutivité d'ONOS le rend hautement disponible et résistant aux pannes, ce qui améliore l'expérience utilisateur du client de l'opérateur. [14]

Comparé à Floodlight, OpenDayLight est l'un des contrôleurs SDN open source les plus répandus avec des API étendues vers le nord et vers le sud. OpenDayLight a été intégré à d'autres solutions de virtualisation et de gestion SDN/NFV open source telles que OpenStack, Kubernetes, OPNFV et ONAP.

Par conséquent, la résilience, l'évolutivité et l'architecture modulaire d'ODL en fait un choix approprié pour différents cas d'utilisation. C'est pourquoi nous portons notre premier choix de solution de réseau à définition logicielle sur OpenDayLight et en deuxième lieu sur Floodlight.

Conclusion

Ce chapitre nous a présenté de manière claire et explicite les généralités sur le SDN dans sa globalité. Nous avons pu comprendre les fonctions de contrôle de



réseau permettant de centraliser la gestion et la programmation du réseau. Toujours dans le même ordre d'idées, une analyse comparative a été menée pour sélectionner deux contrôleurs SDN pouvant correspondre aux exigences de notre projet.

Pour conclure, les avantages des réseaux à définition logicielle aussi bien sur le fonctionnement que sur les coûts d'investissement et d'exploitation font d'eux le concept réseau à adopter pour une infrastructure réseau optimale et automatisée.



CHAPITRE 3 : IMPLEMENTATION DE LA SOLUTION DE RESEAU A DEFINITION LOGICIELLE

Introduction

Les chapitres précédents ont établi le contexte du travail présenté dans ce projet. Dans ce nouveau chapitre nous entrons dans l'aspect pratique du projet qui consiste à présenter la plateforme, installer et configurer les différents outils et composants nécessaires à l'implémentation pour notre réseau SDN.

3.1. Présentation du réseau à mettre en place

Nous exploiterons la nature portable et orientée logiciel du SDN pour la mise en œuvre du réseau dans un environnement virtuel avec OpenDayLight, Floodlight, l'émulateur Mininet et OpenvSwitch, le tout sous l'hyperviseur VMware.

Toutefois, la mise en place réelle du réseau du système hospitalier de Mouyondzi devra être composé de :

- Commutateurs

Des commutateurs physiques et virtuels seront utilisés pour connecter les différents postes et serveurs du réseau repartis en trois VLANs, dont un pour les postes clients, un pour les serveurs et un dernier pour l'administration du réseau.

- Routeurs

Les routeurs à mettre en place assureront l'acheminement des paquets entre les sites du système hospitalier de la ville et le routage du flux Internet.

- Un pare-feu

Il assurera le filtrage simple de paquets en analysant les en-têtes de chaque paquet de données (datagramme) échangé entre les réseaux externes et notre réseau.

Une politique de sécurité avec les listes de contrôle d'accès sera mise en place sur ce firewall afin de :

- Protéger le réseau interne des attaques venant de l'extérieur tout en autorisant l'accès au réseau local ;



- Permettre aux utilisateurs d'accéder aux services extérieurs ;
- Permettre l'accès interactif aux serveurs uniquement depuis les postes du service d'administration ;
- Tracer et/ou stocker les paquets répondant à certaines des règles établies ;

- **Serveurs pour les contrôleurs**

Nous avons un serveur Ubuntu sur lequel sera installé le contrôleur Opendaylight pour les tâches d'administration et un deuxième serveur Floodlight pour la gestion de tunnels VTEP pour un réseau de superposition de type VXLAN.



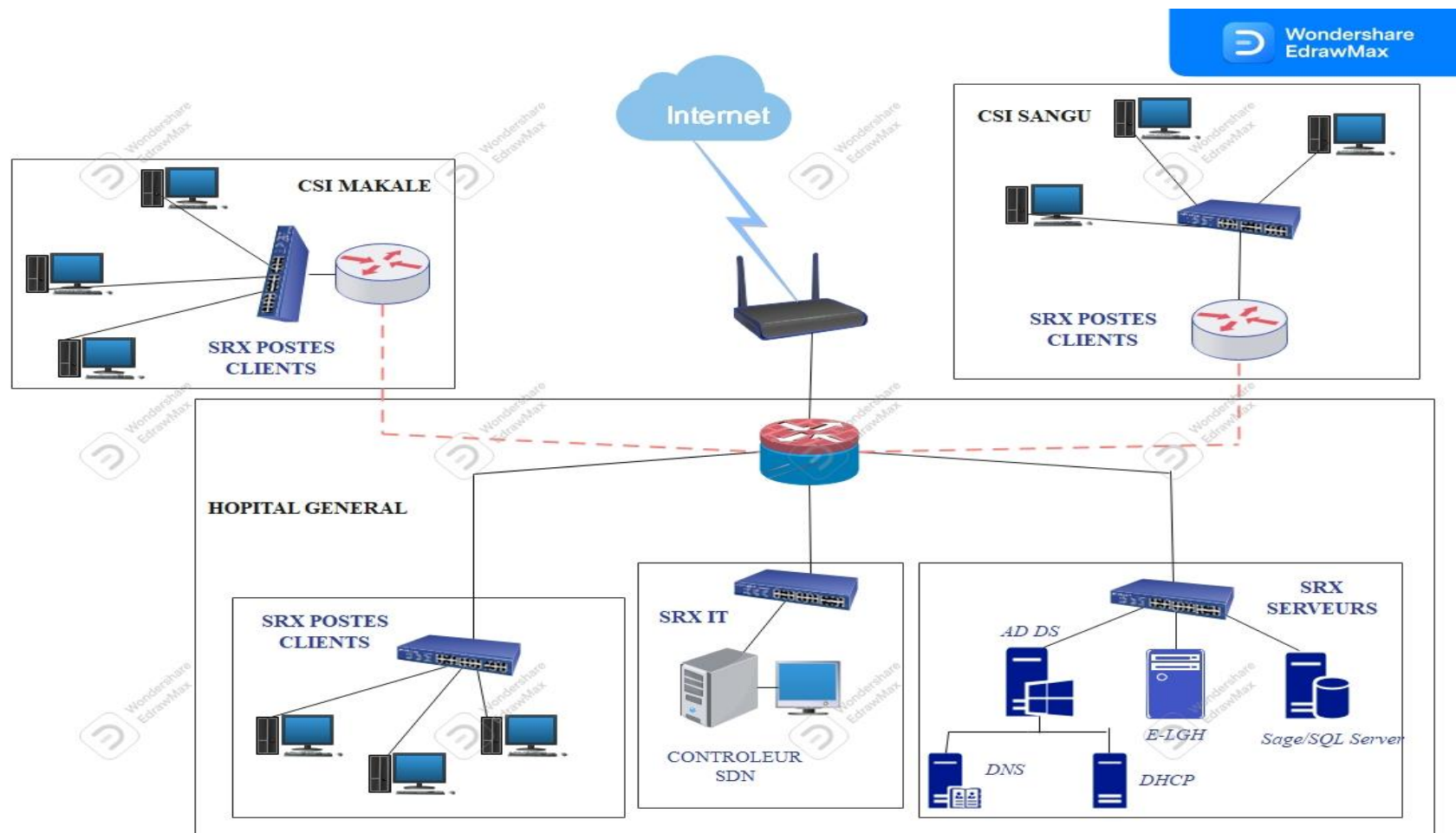


Figure 3. 1: Architecture réseau du système hospitalier de Mouyondzi



3.1.1. Présentation d'Open vSwitch et de l'émulateur Mininet-Wifi

Dans le cadre de ce projet, nous travaillerons avec l'émulateur Mininet et OpenvSwitch. Il reste donc important de présenter Mininet que nous utiliserons un peu plus mais aussi le commutateur virtuel Open vSwitch qui est outil assez utilisé dans le réseau à définition logicielle.

3.1.1.1.Open vSwitch

Open vSwitch est un commutateur virtuel multicouche de qualité production sous licence open source Apache 2.0. Il est conçu pour permettre une automatisation massive du réseau via une extension programmatique, tout en prenant en charge les interfaces et protocoles de gestion standard (par exemple, NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag).

a. Architecture OvS

Deux principales parties composent la plateforme Open vSwitch :

- ✓ Le Module du noyau ;
- ✓ L'ensemble d'outils utilisateur ;

Il existe un certain nombre d'outils CLI qui s'interfacent avec les différents composants :

- **ovs-vswitchd** : un démon qui implémente le commutateur, ainsi qu'un module compagnon du noyau Linux pour la commutation basée sur les flux ;
- **ovsdb-server** : un serveur de base de données léger que ovs-vswitchd interroge pour obtenir sa configuration ;
- **ovs-dpctl** : un outil pour configurer le module du noyau du commutateur ;
- **ovs-vsctl** : un utilitaire pour interroger et mettre à jour la configuration de ovs-vswitchd ;
- **ovs-appctl** : un utilitaire qui envoie des commandes aux démons Open vSwitch en cours d'exécution ;
- **ovs-ofctl** : un utilitaire pour interroger et contrôler les commutateurs et contrôleurs OpenFlow ;
- **ovs-pki** : un utilitaire pour créer et gérer l'infrastructure à clé publique pour les commutateurs OpenFlow ;



- **ovs-testcontroller** : un simple contrôleur OpenFlow qui peut être utile pour les tests (mais pas pour la production).

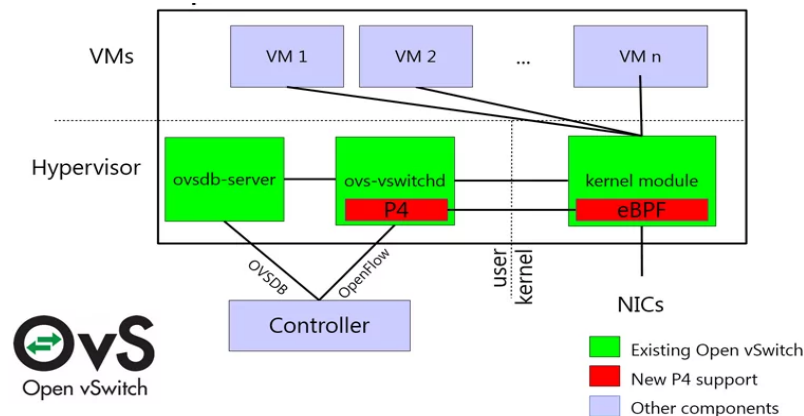


Figure 3. 2 : Architecture OvS

b. Fonctionnement de OvS

Le fonctionnement d'Open vSwitch répond à quatre (4) principales exigences dont la mobilité de l'état, la dynamique du réseau, la maintenance de balises logiques et l'intégration matérielle.

Open vSwitch prend en charge à la fois la configuration et la migration de l'état du réseau lent (configuration) et rapide entre les instances. Par exemple, si une machine virtuelle migre entre des hôtes finaux, il est possible non seulement de migrer la configuration associée (règles SPAN, ACL, QoS) mais tout l'état du réseau en direct (par exemple, l'état existant qui peut être difficile à reconstruire).

Les environnements virtuels sont souvent caractérisés par des taux de changement élevés, Open vSwitch vient gérer certaines fonctionnalités permettant à un système de contrôle de réseau de réagir et de s'adapter à mesure que l'environnement change. Cela inclut une prise en charge simple de la comptabilité et de la visibilité, telle que NetFlow, IPFIX et sFlow. Une base de données d'état du réseau (OVSDb) avec des déclencheurs à distance est pris en charge par Open vSwitch. OpenFlow est également utilisé par Open vSwitch comme méthode d'exportation de l'accès à distance pour contrôler le trafic.

Plusieurs méthodes pour spécifier et maintenir les règles de balisage, qui sont toutes accessibles à un processus distant pour l'orchestration, sont incluses dans le commutateur virtuel. Un stockage sous une forme optimisée des règles de marquage est réalisé afin qu'elles n'aient pas à être couplées à un périphérique réseau lourd.



Cela permet de configurer, de modifier et de migrer des milliers de règles de balisage ou de remappage d'adresses sans oublier l'implémentation GRE capable de gérer des milliers de tunnels GRE simultanés.

Open vSwitch dispose d'un chemin de données dans le noyau conçu pour décharger le traitement des paquets vers des chipsets matériels hébergés dans un châssis de commutateur matériel classique ou dans une carte réseau d'hôte final. Cela permet au chemin de contrôle Open vSwitch de contrôler aussi bien une implémentation logicielle pure qu'un commutateur matériel. [19]

3.1.1.2. Mininet

Mininet est un émulateur de réseau, écrit presque entièrement Python, qui crée un réseau d'hôtes virtuels, de commutateurs, de contrôleurs et de liens. Les hôtes Mininet exécutent un logiciel réseau Linux standard et ses commutateurs prennent en charge OpenFlow pour un routage personnalisé très flexible et une mise en réseau définie par logiciel.

Mininet prend en charge la recherche, le développement, l'apprentissage, le prototypage, les tests, le débogage et toute autre tâche qui pourrait bénéficier d'un réseau expérimental complet sur un ordinateur portable ou un autre PC.

Mininet fournit un banc d'essai réseau simple et peu coûteux pour le développement d'applications OpenFlow. Il permet à plusieurs développeurs de travailler simultanément et indépendamment sur la même topologie. Mininet prend en charge les tests de régression au niveau du système, qui sont reproductibles et facilement empaquetés. Les tests de topologie complexes sont également possibles sans avoir besoin de câbler un réseau physique. Une CLI est inclut prenant en charge la topologie et OpenFlow, pour le débogage ou l'exécution de tests à l'échelle du réseau. La personnalisation de topologie aussi bien prise ne charge que les topologies paramétrées et le tout est utilisable directement sans programmation, mais Mininet dispose également d'une API Python simple et extensible pour la création et l'expérimentation de réseaux.



a. Fonctionnement

Presque tous les systèmes d'exploitation virtualisent les ressources informatiques à l'aide d'une abstraction de processus. Mininet utilise la virtualisation basée sur les processus pour exécuter de nombreux hôtes et commutateurs (nous avons démarré avec succès jusqu'à 4096) sur un seul noyau de système d'exploitation. Depuis la version 2.2.26, Linux prend en charge les espaces de noms réseau, une fonctionnalité de virtualisation légère qui fournit aux processus individuels des interfaces réseau, des tables de routage et des tables ARP distinctes. L'architecture complète du conteneur Linux ajoute **Chroot()**jails, les espaces de noms de processus et d'utilisateurs, ainsi que les limites de CPU et de mémoire pour fournir une virtualisation complète au niveau du système d'exploitation, mais Mininet ne nécessite pas ces fonctionnalités supplémentaires. Mininet peut créer des commutateurs OpenFlow du noyau ou de l'espace utilisateur, des contrôleurs pour contrôler les commutateurs et des hôtes pour communiquer sur le réseau simulé. Mininet connecte les commutateurs et les hôtes à l'aide de **veth** paires Ethernet virtuelles. Alors que Mininet dépend actuellement du noyau Linux, à l'avenir, il pourrait prendre en charge d'autres systèmes d'exploitation avec une virtualisation basée sur les processus, tels que les conteneurs Solaris ou les jails FreeBSD.

Par rapport aux approches basées sur la virtualisation complète du système, Mininet :

- Démarre plus rapide : secondes au lieu de minutes ;
- Fournit une échelle plus grande : des centaines d'hôtes et de commutateurs contre un seul chiffre ;
- Fournit plus de bande passante : généralement 2 Gbit/s de bande passante totale sur un matériel modeste ;
- S'installe facilement : une VM prépackagée est disponible qui s'exécute sur VMware ou VirtualBox pour Mac/Win/Linux avec les outils OpenFlow v1.0 déjà installés. [22]

Par rapport aux simulateurs, Mininet :

- Exécute un code réel non modifié, y compris le code d'application, le code du noyau du système d'exploitation et le code du plan de contrôle



(à la fois le code du contrôleur OpenFlow et le code Open vSwitch) ;

- Se connecte facilement aux réseaux réels ;
- Offre des performances interactives.

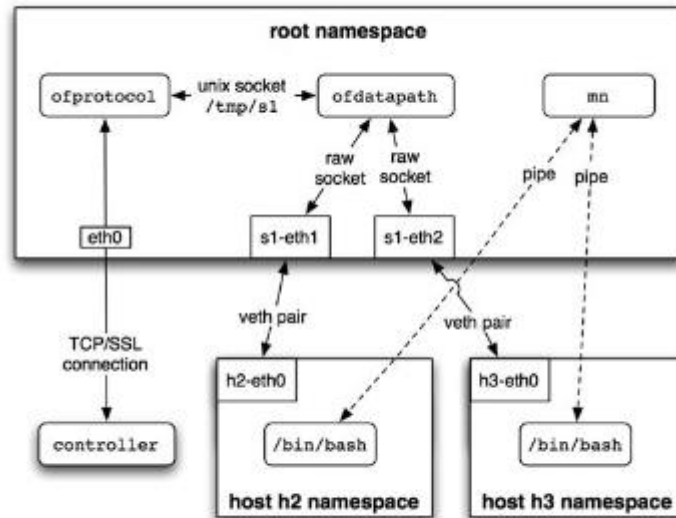


Figure 3. 3: Instance réseau virtuel sur Mininet

3.1.1.3. Mininet-Wifi

Mininet-Wifi est un fork de l'émulateur de réseau Mininet SDN et étend les fonctionnalités de Mininet avec l'ajout des stations Wifi virtualisées et des points d'accès basés sur les pilotes sans fil Linux standard et le pilote de simulation sans fil 80211_hwsim. Mininet prend en charge la recherche, le développement, l'apprentissage, le prototypage, les tests, le débogage et toute autre tâche qui pourrait requérir d'un réseau expérimental complet sur un ordinateur portable ou un autre PC. Cela signifie nouvelles classes ont été ajoutées afin de prendre en charge l'ajout de ces dispositifs sans fil dans un scénario de réseau Mininet et d'émuler les attributs d'une station mobile tels que la position et le mouvement par rapport aux points d'accès.

Mininet-Wifi étend la base de code Mininet en ajoutant ou en modifiant des classes et des scripts. Ainsi, Mininet-Wifi ajoute de nouvelles fonctionnalités et prend toujours en charge toutes les capacités d'émulation SDN normales de l'émulateur de réseau Mininet standard.



a. Architecture et composants

Les principaux composants qui font partie du développement de Mininet-Wifi sont illustrés dans la figure ci-dessous. Dans l'espace noyau, le module `mac80211_hwsim` est chargé de créer des interfaces Wi-Fi virtuelles, importantes pour les stations et les points d'accès. Toujours dans l'espace noyau, MLME (Media Access Control Sublayer Management Entity) est réalisé du côté des stations, tandis que dans l'espace utilisateur, l'`hostapd` est responsable de cette tâche du côté AP.

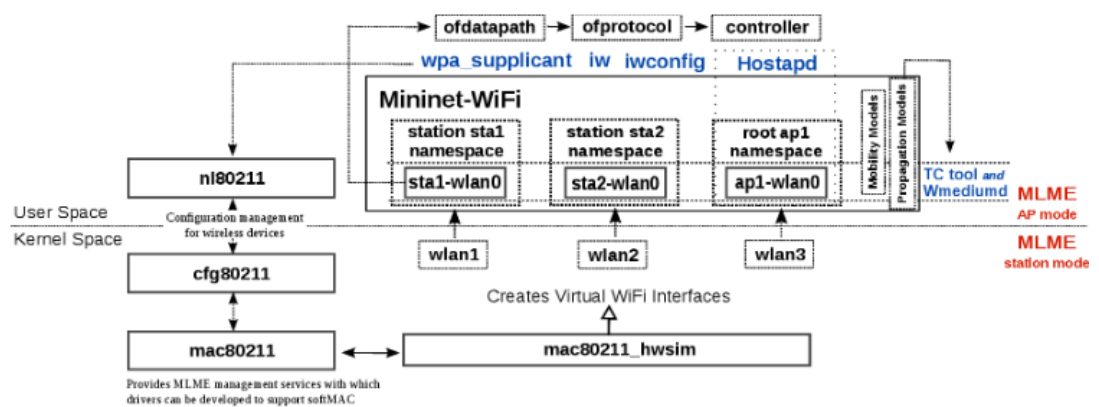


Figure 3. 4: Architecture Mininet-Wifi

Mininet-Wifi utilise également quelques utilitaires tels que `iw`, `iwconfig` et `wpa_supplicant`. Les deux premiers sont utilisés pour la configuration de l'interface et pour obtenir des informations des interfaces sans fil et le dernier est utilisé avec `Hostapd`, afin de prendre en charge le WPA (Wi-Fi Protected Access), entre autres. En plus d'eux, un autre utilitaire fondamental est TC (Traffic Control). Le TC est un programme utilitaire en espace utilisateur utilisé pour configurer le planificateur de paquets du noyau Linux, responsable du contrôle du débit, du retard, de la latence et de la perte, en appliquant ces attributs dans les interfaces virtuelles des stations et des points d'accès, représentant avec une plus grande fidélité le comportement du réel monde.

b. Fonctionnement

Le fonctionnement de Mininet-Wifi peut s'illustrer par la figure ci-dessous avec les composants et les connexions dans une topologie simple de deux stations (ou hôtes) créées, où les composants nouvellement implémentés (surlignés en gris) sont présentés le long des blocs de construction Mininet d'origine. Bien que les stations



soient équipées d'une interface sans fil par défaut, elles peuvent également se connecter à des points d'accès via des liaisons filaires (paires veth).

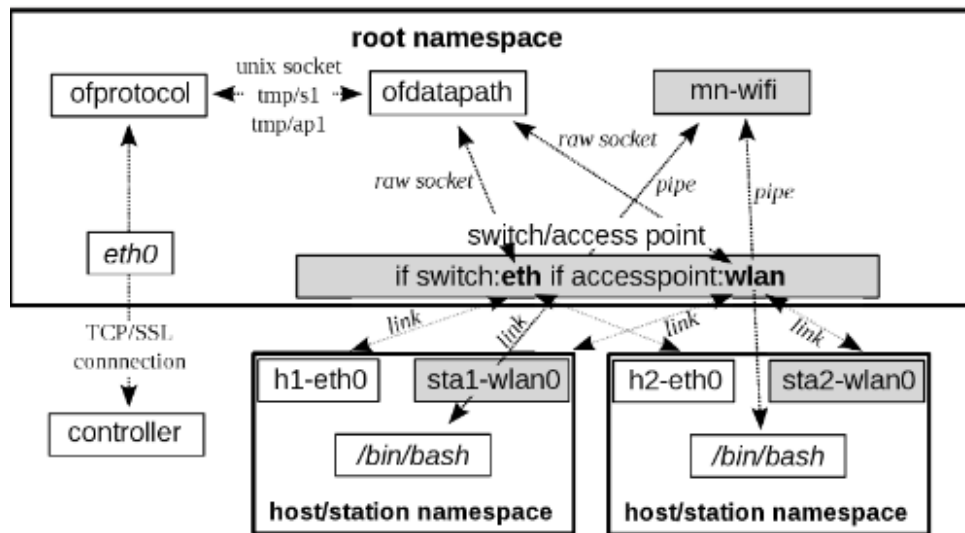


Figure 3. 5: Illustration simple du fonctionnement de Mininet-Wifi

On peut remarquer que les espaces de noms réseau du système d'exploitation Linux sont interconnectés via des paires Ethernet virtuelles (veth). Les interfaces sans fil pour virtualiser les appareils Wifi fonctionnent en mode maître pour les points d'accès et en mode géré pour les stations.

Les stations et les points d'accès utilisent `cfg80211` pour communiquer avec le pilote de périphérique sans fil, une API de configuration Linux 802.11 qui assure la communication entre les stations et `mac80211`. Ce framework communique à son tour directement avec le pilote de périphérique Wifi via un socket netlink (ou plus précisément `nl80211`) qui est utilisé pour configurer le périphérique `cfg80211` et également pour la communication noyau-espace utilisateur. [23]

🚦 **Les Stations** : Ce sont des appareils qui se connectent à un point d'accès par authentification et association. Dans la figure ci-dessus, chaque station possède une carte sans fil (`staX-wlan0` où X doit être remplacé par le numéro de chaque station). Étant donné que les hôtes Mininet traditionnels sont connectés à un point d'accès, les stations peuvent communiquer avec ces hôtes.



✚ **Les points d'accès** : Ce sont des appareils chargés de la gestion des stations associées. Ces appareils sont virtualisés via le démon **hostapd** et utilisent des interfaces sans fil virtuelles pour les points d'accès et les serveurs d'authentification. Mininet-Wifi inclut actuellement la prise en charge des implémentations de référence de l'espace utilisateur et Open vSwitch dans les modes noyau et espace utilisateur. Mininet-Wifi prenait en charge l'implémentation de référence du noyau OpenFlow 0.8.9 (**-ap kernel**), mais celle-ci est désormais obsolète et a été largement remplacée par Open vSwitch. Les options de ligne de commande sont **-ap user** (identique à **UserAP**) et **-ap ovsk** (identique à **OVSAP** ou **OVSKernelAP**) pour la référence utilisateur et les aps du noyau Open vSwitch, respectivement. Il est également possible d'installer le **CPqD BOFUSSap** en utilisant la commande **install.sh -3f** et il remplacera le commutateur de référence de Stanford, c'est-à-dire **-ap user** et **UserAP**.

c. Emulation de support sans fil

La simulation du support sans fil s'appuie sur deux approches :

- Contrôle du Trafic

Tc est le programme utilitaire de l'espace utilisateur utilisé pour configurer le planificateur de paquets du noyau Linux. Utilisé pour configurer le contrôle du trafic dans le noyau Linux, le contrôle du trafic comprend quatre (4) éléments dont la mise en forme, la planification, le maintien de l'ordre et la suppression. Ces quatre (4) propriétés ont été utilisées pour appliquer des valeurs de bande passante, de perte, de latence et de retard dans Mininet-Wifi. TC a été la première approche adoptée dans Mininet-Wifi pour simuler le support sans fil.

- Wmediumd

Le module de noyau mac80211_hwsim utilise le même support virtuel pour tous les nœuds sans fil. Cela signifie que tous les nœuds sont en interne à portée les uns des autres et qu'ils peuvent être découverts lors d'une analyse sans fil sur les interfaces virtuelles. Mininet-Wifi simule leur position et leurs portées sans fil en attribuant des stations à d'autres stations ou points d'accès et en révoquant ces associations sans fil. Si les interfaces sans fil doivent être isolées les unes des autres (par exemple dans des réseaux ad hoc ou maillés), une solution telle que wmediumd



est nécessaire. Il utilise une sorte de répartiteur pour autoriser ou refuser le transfert de paquets d'une interface à une autre.

Par rapport au TC Wmediumd s'est avéré être la meilleure approche pour la simulation du support sans fil. Certains avantages incluent :

- ✓ Isolation des interfaces sans fil les unes des autres ;
- ✓ Wmediumd implémente un algorithme de backoff alors que TC s'appuie uniquement sur la discipline de file d'attente FIFO ;
- ✓ Prise de décision quand l'association doit être évoquée en fonction du niveau du signal ;
- ✓ Les valeurs de bande passante, de perte, de latence et de retard sont appliquées en s'appuyant sur une matrice. Cette matrice implémente une option pour déterminer le PER (taux d'erreur de paquet) avec la matrice externe définie dans IEEE 802.11ax. La matrice est définie à l'annexe 3 de la méthodologie d'évaluation 11-14-0571-12 TGax ;
- ✓ Wmediumd est fortement recommandé pour les réseaux maillés adhoc et sans fil. [23]

3.1.2. Serveur OpenDayLight

Comme il a été décrit dans le chapitre précédent, ODL (OpenDayLight) est une plate-forme ouverte modulaire permettant de personnaliser et automatiser des réseaux de toute taille et échelle.

3.1.2.1. Architecture OpenDayLight

La plateforme d'OpenDayLight est structurée autour des plugins vers le sud, d'interfaces vers le nord (par exemple REST/NETCONF) et d'un ensemble services de base qui peuvent être utilisés au moyen de la couche d'abstraction de service basée sur OSGi pour aider les composants à entrer et sortir du contrôleur pendant que le contrôleur est en cours d'exécution. Chacun des composants est isolé en tant que fonctionnalité Karaf, pour garantir que les nouveaux travaux n'interfèrent pas avec le code mature et testé. OpenDaylight utilise OSGi et Maven pour créer un package qui gère ces fonctionnalités Karaf et leurs interactions.



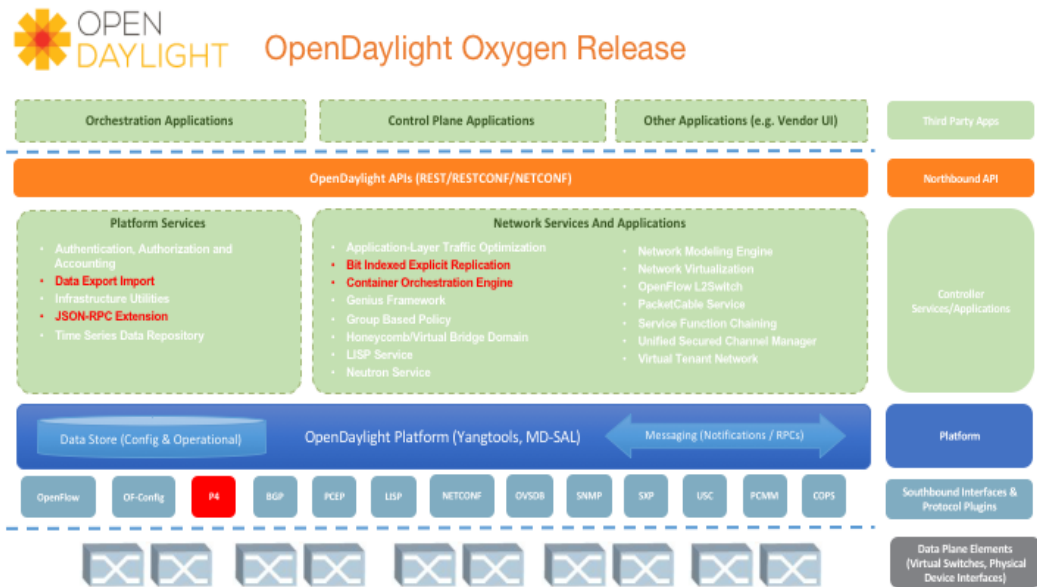


Figure 3. 6: Architecture OpenDayLight version Oxygène

La nomenclature de chacune des versions est faite suivant les numéros atomiques des éléments chimiques.

Tableau 3. 2 : Tableau des versions de OpenDayLight

Version d'OpenDayLight	Date de sortie
Sulfure (16)	Mars-22
Phosphore (15)	Sept-21
Silicium (14)	Mars-2021
Aluminium (13)	Sept-20
Magnésium (12)	Mars-20
Sodium (11)	Sept-19
Néon (10)	Mars-19
Fluor (9)	Août-18
Oxygène (8)	Mars-18
Azote (7)	Sept-17
Carbone (6)	Juin-17
Bore (5)	Nov-16
Béryllium (4)	Fevr-16
Lithium (3)	Juin-15
Hélium (2)	Oct-14
Hydrogène (1)	Févr-14

Nous choisissons la version 8 pour son interface GUI et sa compatibilité avec plusieurs systèmes open source.

3.1.2.2. Fonctionnement de ODL

ODL est principalement basé sur son approche de modèle, ce qui implique qu'une vue globale en mémoire du réseau est nécessaire pour effectuer des calculs logiques. Le package du contrôleur repose sur quatre composants clés (odlparent, contrôleur, MD-SAL et yangtools). Chacun des composants ODL est isolé en tant que fonctionnalité Karaf, pour s'assurer que les nouveaux travaux n'interfèrent pas avec le code déjà validé. OpenDaylight utilise OSGi et Maven pour créer un package qui gère ces fonctionnalités Karaf et leurs interactions. ODL utilise la distribution Karaf pour démarrer. Apache Karaf est un petit environnement d'exécution basé sur OSGi qui fournit un conteneur léger sur lequel divers composants et applications peuvent être déployés. Karaf permet à l'utilisateur de choisir les protocoles et les services que le contrôleur prendra en charge. L'environnement Karaf génère une carte des dépendances et maintient le démarrage rapide et le déploiement de nouvelles fonctionnalités, selon la configuration faite. Karaf ODL présente une sorte de console permettant de faire la configuration des composants, par exemple l'interface utilisateur ou encore OpenFlow nécessaire pour la communication entre le contrôleur et le plan de données. Certaines fonctionnalités sont proactives, ce qui signifie qu'OpenDaylight, en contact avec d'autres éléments du réseau, commence à apporter des modifications au réseau même sans y être invité par les utilisateurs, afin de satisfaire aux conditions initiales attendues par leur cas d'utilisation. Une telle activité d'une fonctionnalité peut à son tour affecter le comportement d'une autre fonctionnalité. Dans certains cas, il existe des fonctionnalités qui offrent différentes implémentations du même service, elles peuvent ne pas s'initialiser correctement (par exemple, ne pas lier un port déjà lié par l'autre fonctionnalité). Il faut donc veiller à vérifier les ports et n'activer que les fonctionnalités dont on a besoin. [24]

3.1.3. Contrôleur Floodlight

Le contrôleur Floodlight Open SDN est un contrôleur OpenFlow basé sur Java, sous licence Apache et conçu pour fonctionner avec les outils **JDK** standard et **ant**.

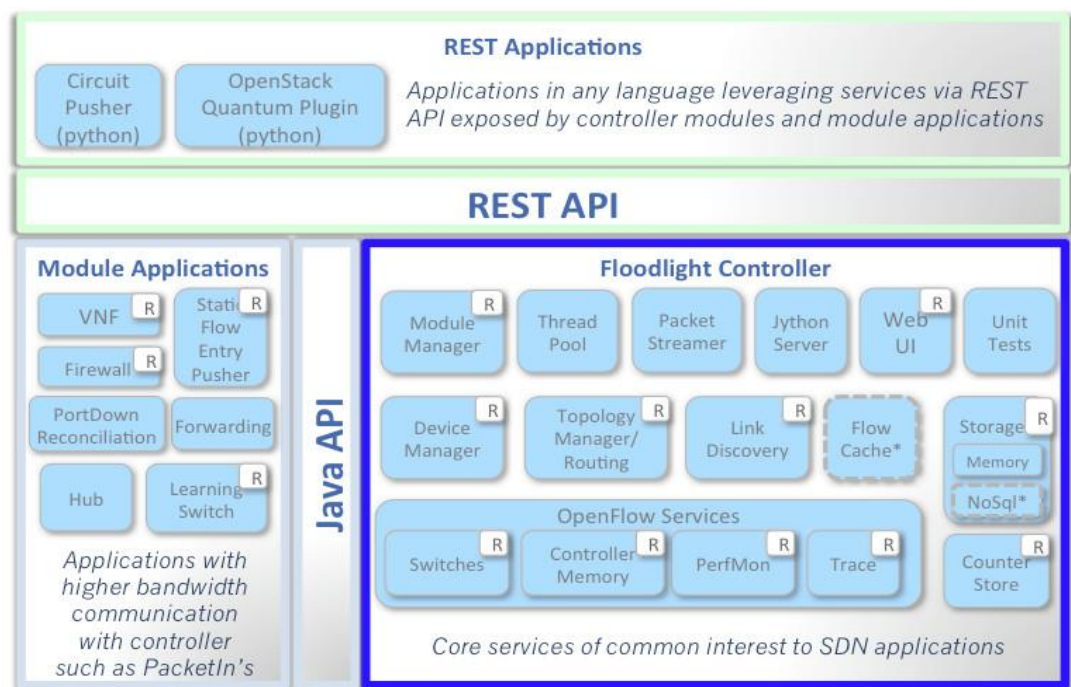


3.1.3.1. Architecture de Floodlight

Floodlight n'est pas seulement un contrôleur OpenFlow. A part la fonction de contrôleur OpenFlow, un ensemble d'applications sont construites sur le contrôleur Floodlight.

Le contrôleur Floodlight intègre un ensemble de fonctionnalités communes pour contrôler et interroger un réseau OpenFlow, tandis que les applications qui le surmontent disposent de différentes fonctionnalités pour répondre aux différents besoins des utilisateurs sur le réseau.

Les figures ci-dessous illustrent la relation entre le contrôleur Floodlight, les applications créées en tant que modules Java compilés avec Floodlight et les applications créées via l'API REST Floodlight :



* Interfaces defined only & not implemented: FlowCache, NoSql

Figure 3. 7: Architecture Floodlight

a. Fonctionnement de Floodlight

Floodlight adopte une architecture modulaire pour implémenter ses fonctionnalités de contrôleur et certaines applications. Module Loading System décrit l'interface Java IFloodlightModule qui réalise ce Framework.

Sur le plan fonctionnel, Floodlight se compose de modules de contrôleur qui implémentent des services de réseau central qu'un réseau défini par logiciel exposerait aux applications, et de modules d'application qui implémentent des



solutions à des fins différentes. Les modules peuvent tirer parti des fonctionnalités et des fonctionnalités exposées par d'autres modules grâce à l'utilisation de services Floodlight.

Les modules du contrôleur implémentent des fonctions qui sont d'usage commun à une majorité d'applications, telles que :

- La découverte et exposition des états et événements du réseau (topologie, appareils, flux) ;
- Permettre la communication du contrôleur avec les commutateurs réseau (c'est-à-dire le protocole OpenFlow) ;
- Gestion des modules Floodlight et des ressources partagées telles que le stockage, les threads, les tests ;
- Une interface utilisateur Web et un serveur de débogage (Jython).

Les modules Floodlight qui souhaitent exposer des fonctionnalités à d'autres modules Floodlight ou à l'API REST doivent d'abord mettre en œuvre un service Floodlight.

Floodlight met à disposition une page web de référence pour tous les services Floodlight existants pouvant être exploités à partir d'un module Floodlight. Pour utiliser l'un de ces services, il faudrait d'abord spécifier le service en tant que dépendance dans la fonction `getModuleDependencies()` du module utilisé. [26]

3.2. Mise en place de la solution SDN

Dans ce mémoire nous mettrons en œuvre, dans un premier temps, un réseau virtuel SDN LAN avec notre contrôleur et ensuite nous déploierons un réseau de superposition de type VXLAN.

Cas 1 : Mise en place d'un réseau virtuel SDN

Ce déploiement se fera dans un environnement virtuel. Cette option est d'autant plus intéressante pour notre projet vu la richesse des switchs virtuels en possibilités comparés à leurs équivalents hardware. La couche infrastructure se composera de commutateurs Open vSwitch émulés par Mininet.



Cas 2 : Déploiement d'un réseau de superposition de type VXLAN

Il s'agit dans ce cas de créer et de gérer des tunnels VXLAN dans un réseau de superposition avec le contrôleur Floodlight.

3.2.1. Implémentation réseau virtuel SD-LAN

Notre réseau SD-LAN, mis en place sur l'hyperviseur VMWare Workstation, est composé de :

- Une machine virtuelle **Ubuntu 20.4 amd 64 bits** sur laquelle est installée la version OpenDaylight version **Oxygen** et activé l'application **OpenFlow Manager**.
- Une machine virtuelle **Ubuntu serveur 16.04.12 amd 64 bits Mininet** que nous avons importé depuis : <https://github.com/mininet/mininet/releases/> . Toutes les autres versions de l'émulateur y sont disponibles sous formes de VM préparées, il suffit d'importer le fichier OVF sur l'hyperviseur. La version qui a été installée dans notre cas est la **2.3.0**

3.2.1.1. Topologie du réseau virtuel SDN

Cette topologie correspond au réseau local que l'on aimerait installer au sein de l'hôpital général de Mouyondzi. Pour notre exemple, nous avons une architecture réseau composée de six (6) switches d'accès et 8 hôtes.

a. Prise en main du contrôleur OpenDayLight

Sur une VM Ubuntu de version **20.04.1**, nous avons installé OpenDayLight. L'installation de ODL requiert d'un prérequis dont **java jre** et la version de java qu'il faut pour la version de notre contrôleur est Openjdk 8 :

```
levymab@ubuntu:~$ sudo apt-get -y install openjdk-8-jre
[sudo] password for levymab:
Sorry, try again.
[sudo] password for levymab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-8-jre is already the newest version (8u312-b07-0ubuntu1~20.04).
```

Figure 3. 8: Installation Openjdk

Après avoir installé Openjdk, on le configure sur serveur Ubuntu. Cette configuration est disponible en annexe du document.



Une fois notre installation et configuration de java terminées, on passe au téléchargement du contrôleur :

```
levymab@ubuntu:~$ curl -XGET -O https://nexus.opendaylight.org/content/repositories/opendaylight.release/org.opendaylight/integration/karaf/0.8.4/karaf-0.8.4.zip
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
							Speed

Figure 3. 9: Installation ODL

Pour télécharger ODL comme nous l'avons fait sur la figure 3.9, il faut avoir au préalable installé Curl sur la machine. Maintenant nous passons à l'installation du contrôleur en faisant l'extraction du fichier **zip** téléchargé :

```
levymab@ubuntu:~$ sudo unzip karaf-0.8.4.zip
Archive: karaf-0.8.4.zip
```

Figure 3. 10: Extraction du fichier zip

Cela crée un dossier nommé **karaf-0.8.0** qui contient le logiciel et les plugins OpenDaylight :

```
levymab@ubuntu:~$ ls
Desktop  Downloads  karaf-0.8.4.zip  Pictures  Templates
Documents  karaf-0.8.4  Music           Public    Videos
```

Figure 3. 11: Vue du répertoire karaf-0.8.4

Nous avons terminé l'installation du contrôleur, démarrons-le maintenant :

```
levymab@ubuntu:~/karaf-0.8.4$ ./bin/karaf
Apache Karaf starting up. Press Enter to open the shell now...
100% [=====]
Karaf started in 59s. Bundle stats: 430 active, 431 total
```

```

      _____
     /         \
    /           \
   /             \
  /               \
 /                 \
/                   \
\                   /
 \                 /
  \               /
   \             /
    \           /
     \         /
      \       /
       \     /
        \   /
         \ /
          V

```

```

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

```

Figure 3. 12: Démarrage de ODL



Par rapport à nos besoins et pour un fonctionnement optimal de notre contrôleur, nous installons quelques fonctionnalités :

```
transaction-manager-narayana (Narayana Tra
opendaylight-user@root>feature:install odl-dlux-all
```

Figure 3. 13: Installation des fonctionnalités GUI

Dans la figure ci-dessus nous avons installé les fonctionnalités d'interface graphique.

Notre contrôleur est fin prêt à exploiter et nous accédons à l'interface d'administration en tapant, dans le navigateur, l'adresse IP du serveur sur laquelle écoute ODL et le port utilisé par l'application d'interface graphique :

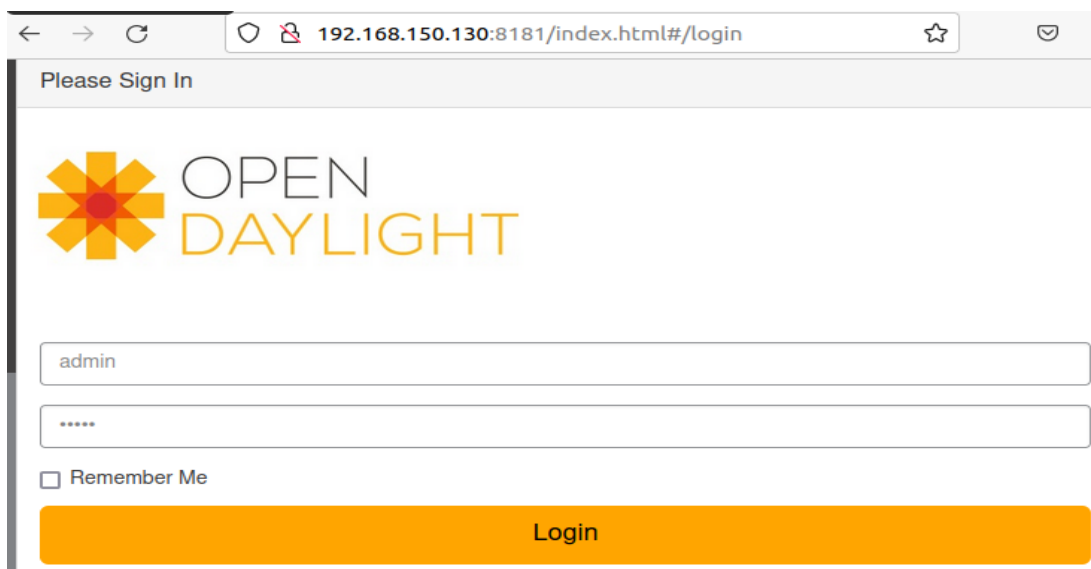


Figure 3. 14: Connexion GUI

Le login et le mot de passe de connexion sont par défaut : **admin**

b. Création de la topologie sous Mininet

Nous avons personnalisé la partie infrastructure de notre réseau via l'API Mininet en créant et éditant le fichier **matopologie.py**, puis nous l'avons généré. Le fichier se trouve dans le sous-dossier **custom** du dossier **mininet** :




```

GNU nano 2.5.3                                     File: mininet/custom/matopologie.py

#!/usr/bin/python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Node, Controller, RemoteController, OVSSwitch
from mininet.log import setLogLevel, info
from mininet.cli import CLI
from mininet.util import irange
from mininet.link import TCLink

class MaTopo( Topo ):

    def build( self ):

        h1 = self.addHost( 'h1', ip='192.168.150.100/24' )
        h2 = self.addHost( 'h2', ip='192.168.150.101/24' )
        h3 = self.addHost( 'h3', ip='192.168.150.102/24' )
        h4 = self.addHost( 'h4', ip='192.168.150.103/24' )
        h5 = self.addHost( 'h5', ip='192.168.150.104/24' )
        h6 = self.addHost( 'h6', ip='192.168.150.105/24' )
        h7 = self.addHost( 'h7', ip='192.168.150.106/24' )
        h8 = self.addHost( 'h8', ip='192.168.150.107/24' )

        s1 = self.addSwitch( 's1', dpid='0000000000000001', protocols='OpenFlow13' )
        s2 = self.addSwitch( 's2', dpid='0000000000000002', protocols='OpenFlow13' )
        s3 = self.addSwitch( 's3', dpid='0000000000000003', protocols='OpenFlow13' )
        s4 = self.addSwitch( 's4', dpid='0000000000000004', protocols='OpenFlow13' )
        s5 = self.addSwitch( 's5', dpid='0000000000000005', protocols='OpenFlow13' )
        s6 = self.addSwitch( 's6', dpid='0000000000000006', protocols='OpenFlow13' )

        #Coeur
        self.addLink ( s3, s6 )

        #distribution
        self.addLink ( s1, s2 )
        self.addLink ( s2, s3 )
        self.addLink ( s3, s4 )
        self.addLink ( s4, s5 )

        #access
        self.addLink( s1, h1 )
        self.addLink( s2, h2 )
        self.addLink( s2, h3 )
        self.addLink( s3, h4 )
        self.addLink( s3, h5 )
        self.addLink( s4, h6 )
        self.addLink( s4, h7 )
        self.addLink( s5, h8 )

    def run():
        topo = MaTopo()

        net = Mininet( topo=topo )
        net.start()
        CLI( net )
        net.stop()
        if __name__ == '__main__':
            setLogLevel( 'info' )

            run()

topos = { 'mytopo': MaTopo }

```

Figure 3. 15: Fichier de création de la topologie

Tout en s'assurant que nous notre contrôleur OpenDaylight est démarré, nous allons générer notre topologie sur Mininet en tapant la commande :

sudo mn --custom mininet/custom/ matopologie.py --topo mytopo --mac --controller=remote, ip=192.168.150.130,port=6633 --switch=ovsk,protocols=OpenFlow13



```
mininet@mininet-vm:~$ sudo mn --custom mininet/custom/topoperso.py --topo mytopo --mac --controller=
remote,ip=192.168.150.130,port=6633 --switch=ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s2) (h3, s2) (h4, s3) (h5, s3) (h6, s4) (h7, s4) (h8, s5) (s1, s2) (s2, s3) (s3, s4) (
s3, s6) (s4, s5)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> _
```

Figure 3. 16: Topologie générée

Maintenant que le déploiement sur Mininet est fait, nous pouvons voir notre architecture via l'interface graphique du contrôleur :

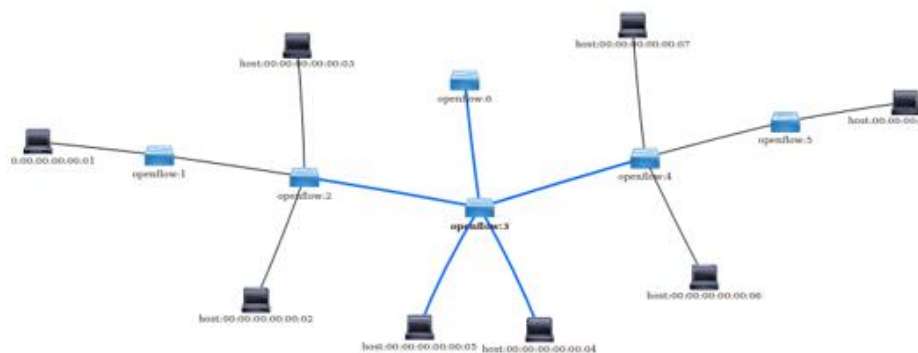


Figure 3. 17: Topologie créée

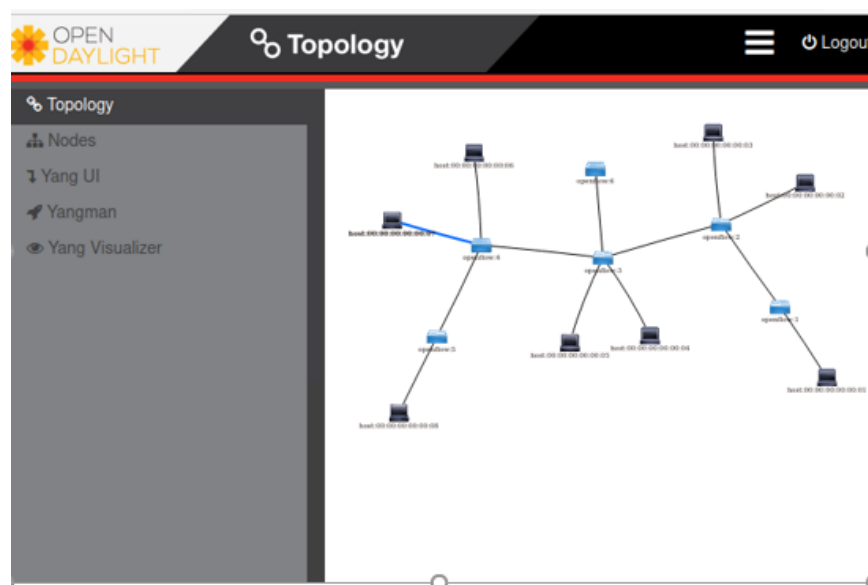


Figure 3. 18: Vue de la topologie sous ODL

Sur notre architecture le contrôleur est connecté au réseau sur le switch S6.



Nous avons dressé un tableau récapitulatif des hôtes générés avec notre topologie :

Tableau 3. 3: Tableau récapitulatif des hôtes créés

Hôtes	Adresses MAC	Adresses IP
h1	00 :00 :00 :00 :00 :01	192.168.150.100/24
h2	00 :00 :00 :00 :00 :02	192.168.150.101/24
h3	00 :00 :00 :00 :00 :03	192.168.150.102/24
h4	00 :00 :00 :00 :00 :04	192.168.150.103/24
h5	00 :00 :00 :00 :00 :05	192.168.150.104/24
h6	00 :00 :00 :00 :00 :06	192.168.150.105/24
h7	00 :00 :00 :00 :00 :07	192.168.150.106/24
h8	00 :00 :00 :00 :00 :08	192.168.150.107/24

3.2.1.2. Planification du réseau

a. Traffic Shaping

Le Traffic Shaping ou régulation de flux est le contrôle du volume des échanges sur un réseau informatique dans le but d'optimiser ou de garantir de meilleures performances.

De nouveaux algorithmes de mise en forme du trafic (TS) sont proposés pour la mise en œuvre d'une technique de gestion de la bande passante avec la qualité de service (QoS) afin d'optimiser les performances et de résoudre les problèmes de congestion du réseau. Plus précisément, deux algorithmes, à savoir « Étiquetage de paquets, mise en file d'attente et transfert vers des files d'attente » et « Allocation de bande passante », sont proposés pour la mise en œuvre d'une technique de mise en file d'attente équitable pondérée (WFQ), en tant que nouvelle méthodologie dans un banc d'essai en tranches SDN pour réduire la congestion et faciliter une circulation fluide. Cette méthodologie visait à améliorer la QoS qui fait deux choses simultanément, premièrement, rendre le trafic conforme à un débit individuel en utilisant WFQ pour créer la file d'attente appropriée pour chaque paquet. Deuxièmement, la méthodologie est combinée avec la gestion de tampon, décidant de mettre ou non le paquet dans la file d'attente selon l'algorithme défini à cet effet. De cette façon, la latence et la congestion restent maîtrisées, répondant ainsi aux exigences des services en temps réel. [3]



Nous allons utiliser le programme **iperf** pour contrôler le flux sur notre réseau virtuel SD-LAN.

- ✚ Logiciel iperf : perf est un logiciel informatique client-serveur développé par le National Laboratory for Applied Network Research permettant la mesure de différentes variables d'une connexion réseau IP. Iperf est disponible sur différents systèmes d'exploitation.
- ✚ Installation de iperf : nous allons installer iperf sur notre contrôleur ODL qui fera office de serveur iperf et sur Mininet sera le client iperf :

[21]

```
levymab@ubuntu:~$ sudo apt install iperf3
[sudo] password for levymab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Figure 3. 19: Installation iperf3 serveur

```
mininet@mininet-vm:~$ sudo apt install iperf3
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Figure 3. 20: Installation iperf3 client

Démarrons maintenant le service iperf sur le serveur et sur le client :

```
levymab@ubuntu:~$ iperf3 -s
-----
Server listening on 5201
-----
Accepted connection from 192.168.150.129, port 51028
[ 5] local 192.168.150.130 port 5201 connected to 192.168.150.129 port 51030
[ ID] Interval           Transfer     Bitrate
[ 5] 0.00-1.00      sec    121 MBytes   1.02 Gbits/sec
[ 5] 1.00-2.00      sec    119 MBytes   1.00 Gbits/sec
[ 5] 2.00-3.00      sec    124 MBytes   1.04 Gbits/sec
[ 5] 3.00-4.00      sec    92.8 MBytes   778 Mb/s
[ 5] 4.00-5.00      sec    93.7 MBytes   787 Mb/s
[ 5] 5.00-6.00      sec    112 MBytes   943 Mb/s
[ 5] 6.00-7.00      sec    114 MBytes   952 Mb/s
[ 5] 7.00-8.00      sec    99.6 MBytes   835 Mb/s
[ 5] 8.00-9.00      sec    108 MBytes   910 Mb/s
[ 5] 9.00-10.00     sec    104 MBytes   871 Mb/s
[ 5] 10.00-10.06    sec     5.12 MBytes  771 Mb/s
-----
[ ID] Interval           Transfer     Bitrate
[ 5] 0.00-10.06    sec    1.07 GBytes   912 Mb/s
-----
receiver
```

Figure 3. 21: Démarrage iperf3 serveur

Nous pouvons voir les mesures au niveau de iperf3 côté serveur. Pour avoir ces mesures il faut commencer par démarrer d'abord iperf3 serveur avec la



commande : **iperf3 -s** puis on démarre iperf3 client en indiquant l'adresse IP du serveur : **Iperf3 -c 192.168.130**

Ainsi, la connexion entre le serveur iperf3 et son client est établie.

```
mininet@mininet-um:~$ iperf3 -c 192.168.150.130
Connecting to host 192.168.150.130, port 5201
[ 41] local 192.168.150.129 port 51030 connected to 192.168.150.130 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr   Cwnd
[ 41] 0.00-1.01   sec    130 MBytes  1.08 Gbits/sec    73   1.48 MBytes
[ 41] 1.01-2.00   sec    119 MBytes  1.00 Gbits/sec     0   1.61 MBytes
[ 41] 2.00-3.03   sec    124 MBytes  1.01 Gbits/sec     0   1.72 MBytes
[ 41] 3.03-4.00   sec    91.2 MBytes  790 Mbits/sec    16   1.26 MBytes
[ 41] 4.00-5.01   sec    93.8 MBytes  782 Mbits/sec     0   1.35 MBytes
[ 41] 5.01-6.00   sec    114 MBytes  960 Mbits/sec     0   1.42 MBytes
[ 41] 6.00-7.01   sec    112 MBytes  935 Mbits/sec     0   1.46 MBytes
[ 41] 7.01-8.01   sec    100 MBytes  840 Mbits/sec     0   1.49 MBytes
[ 41] 8.01-9.00   sec    108 MBytes  910 Mbits/sec     0   1.52 MBytes
[ 41] 9.00-10.00  sec    104 MBytes  870 Mbits/sec     0   1.57 MBytes
-----
[ ID] Interval           Transfer     Bandwidth       Retr
[ 41] 0.00-10.00  sec    1.07 GBytes  918 Mbits/sec    89
[ 41] 0.00-10.00  sec    1.07 GBytes  917 Mbits/sec
iperf Done.
```

Figure 3. 22: Démarrage iperf3 serveur

Les mesures (Bande passante, la pause entre rapports de bande passante, et le volume de transfert) que nous avons sur le serveur iperf sont quasiment identiques à celles sur le client.

✚ **Expérience :** Cette expérience consiste à implémenter une politique QoS avec différentes bandes passantes et deux files d'attente. Puis nous ferons correspondre les QoS à la file d'attente pour l'appliquer à un port du switch que nous aurons choisi. Et enfin nous utiliserons l'API Northbound REST pour envoyer du code source au contrôleur ODL.

- Configuration de la QoS et files d'attente

Nous allons appliquer nos QoS et files d'attente au switch **S1**, port **eth1**. On commence par entrer les paramètres de QoS et file d'attente comme suit :

```
mininet> sh sudo ovs-vsctl set port s1-eth1 qos=@newqos -- --id=@newqos create qos type=linux-htb ot
her-config:max-rate=50000000 queues=1=@q1 -- --id=@q1 create queue other-config:max-rate=50000000 ot
her-config:min-rate=15000000
07d6b593-7eff-4a7e-be18-c0235d0cc3c9
f0758377-ee13-447b-b45d-dff88701acab
mininet> _
```

Figure 3. 23: Configuration QoS1 et File d'attente 1

Ensuite nous faisons le mappage du flux avec notre QoS en se servant du logiciel Postman qui nous permet d'envoyer notre code en format Xml au contrôleur via l'API Northbound :



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <strict>false</strict>
  <hard-timeout>1800</hard-timeout>
  <idle-timeout>3600</idle-timeout>
  <cookie>103</cookie>
  <cookie_mask>255</cookie_mask>
  <installHw>false</installHw>
  <priority>1000</priority>
  <flow-name>flow1</flow-name>
  <match>
```

Figure 3. 24: Extrait code mappage du flux

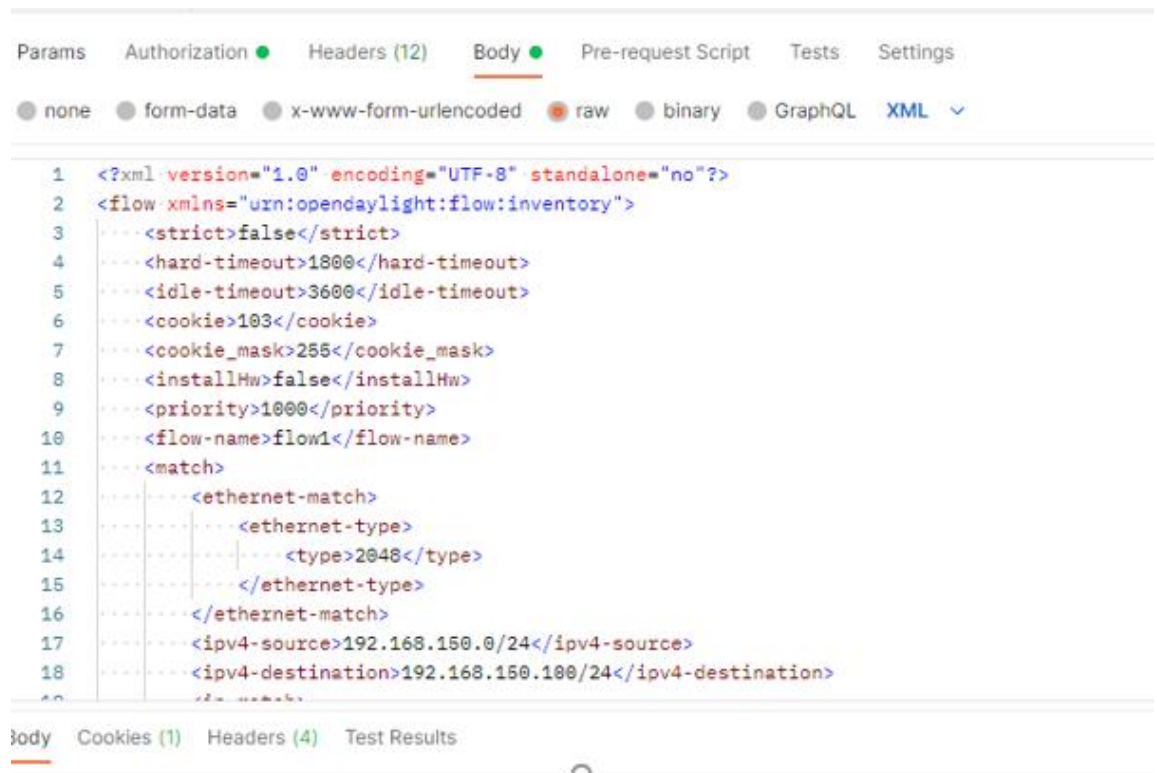


Figure 3. 25: Envoie du code avec Postman

Nous envoyons notre code à l'adresse web suivante :

<http://192.168.1.12:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1>

La méthode d'envoi de la requête vers le contrôleur est **PUT** et le type du contenu est **application/xml**. Nous avons choisi l'autorisation **Basic Auth** (nom d'utilisateur/mot de passe : **admin/admin**)

La QoS et la file d'attente ci-dessus sont celles du trafic allant à l'hôte h1. Nous l'avons aussi fait pour l'hôte h3.

La politique QoS que nous avons appliquée, de priorité 1000, est de type **linux-htb** avec une bande passante maximum de 50 Mb/s, deux files d'attente q1 et q2 sont



créées. Les taux de transfert minimum sont 15 Mb/s pour la file d'attente q1 et 5 Mb/s pour la file d'attente q2. Ces taux de transfert ont été fixé à un maximum de 50 Mb/s en ce qui concerne la q1 et 20 Mb/s pour la q2.

Une fois notre implémentation, pour les deux trafics, nous passons aux tests :

```
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['7.76 Gbits/sec', '7.76 Gbits/sec']
mininet> iperf h3 h1
*** Iperf: testing TCP bandwidth between h3 and h1
.*** Results: ['7.10 Gbits/sec', '7.12 Gbits/sec']
mininet> _
```

Figure 3. 26: Test iperf avant QoS

La figure 3.26 montre un test iperf entre es hôtes h1 et h3 avant l'implémentation de notre politique QoS et files d'attente. On peut constater que les taux de transfert et la bande passante sont à plus de 7 Gb/s.

Maintenant observons les résultats du test fait après l'application de notre politique QoS :

```
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['18.4 Mbits/sec', '21.5 Mbits/sec']
mininet> iperf h3 h1
*** Iperf: testing TCP bandwidth between h3 and h1
.*** Results: ['18.4 Mbits/sec', '22.1 Mbits/sec']
mininet>
```

Figure 3. 27: Test iperf après application QoS

Sur la figure 3.27 le test iperf entre les hôtes h1 et h3 après l'implémentation de notre politique QoS et files d'attente montre que les taux de transfert et la bande passante correspondent bien à ce que nous avons défini comme paramètres dans notre configuration. Nous avons un taux de transfert de 18 Mb/s et une bande passante qui ne dépasse pas 50 Mb/s.

b. DiffServ

Les Services Différenciés (DiffServ) sont une technique qui permet de classer et de contrôler le trafic tout en fournissant de la QoS en différenciant les services des données. L'utilisation de DiffServ dans un réseau permet de configurer directement les paramètres pertinents sur les commutateurs et les routeurs plutôt que d'utiliser un protocole de réservation de ressources.



Les valeurs DSCP (Differentiated Services Code Point) et le marquage DSCP avec des compteurs peuvent être utilisés pour implémenter des exigences QoS élevées de sorte que la garantie de bande passante soit fournie à un ensemble sélectionné de flux de trafic pour atteindre une gestion de bande passante évolutive afin de fournir une qualité de service élevée avec SDN.

Implémentation du DSCP : Tout d'abord il faut s'assurer que notre contrôleur est capable d'interagir avec les fonctionnalités de QoS. Dès que cela est vérifié, il faut installer les fonctionnalités de Network Intent Composition (NIC) qui nous permettront de créer notre QoS dans le contrôleur qui sera utilisé pour la créer une entrée dans la table de flux des switch Mininet. Les switches émulés sous Mininet seront capables de reconnaître un flux prioritaire en lui accordant sa priorité souhaitée.

Approches d'injection de la QoS : On distingue deux méthodes pour injecter de la QoS aux switchs depuis ODL.

- Méthode d'injection de la QoS par la CLI d'ODL

Via la console ODL on crée nos règles QoS avec les commandes suivantes :

intent:qosConfig -p Highest_Quality -d 46 et **intent:qosConfig -p Lowest_Quality -d 4**

Nous avons configuré deux règles, la première nommée **Highest_Quality** la plus prioritaire avec une valeur DSCP de 46 et la deuxième nommée **Lowest_Quality** de priorité inférieure par rapport à la première avec une valeur DSCP de 4.

Après cela, il faut l'appliquer aux flux dans les deux sens entre la (es) machine (s) qu'on veut prioriser par rapport aux autres.

- Méthode par injection de la QoS en utilisant l'API Rest

C'est par cette méthode que nous avons procédé pour la créer de nos règles de QoS. En servant de Postman nous envoyons notre code au contrôleur :




```

15  ... <table_id>0</table_id>
16  ... <id>4</id>
17  ... <cookie_mask>255</cookie_mask>
18  ... <match>
19  ...     <ethernet-match>
20  ...         <ethernet-type>
21  ...             <type>2048</type>
22  ...         </ethernet-type>
23  ...         <ethernet-destination>
24  ...             <address>00:00:00:00:00:02</address>
25  ...         </ethernet-destination>
26  ...         <ethernet-source>
27  ...             <address>00:00:00:00:00:01</address>
28  ...         </ethernet-source>
29  ...     </ethernet-match>
30  ...     <tcp-destination-port>5001</tcp-destination-port>
31  ...     <ip-match>
32  ...         <ip-protocol>4</ip-protocol>
33  ...         <ip-dscp>46</ip-dscp>
34  ...         <ip-ecn>0</ip-ecn>
35  ...     </ip-match>
36  ... </match>
37  ... <hard-timeout>1200</hard-timeout>
38  ... <cookie>255</cookie>

```

Figure 3. 28: Envoie de la QoS via l'API

Nous avons, sur la figure 3.28, la QoS dans un sens c'est-à-dire de h1 à h2. Il faut l'appliquer dans l'autre sens aussi et l'envoyer à la bonne adresse comme nous l'avons fait pour la Traffic Shaping.

Comme on peut le remarquer, nous avons appliqué notre QoS entre les hôtes h1 et h2 dont les adresses MAC sont respectivement **MAC 00 :00 :00 :00 :00 :01** et **00 :00 :00 :00 :00 :02**. Les autres valeurs de DSCP de notre code sont décrites en annexes.

Maintenant que nos valeurs DSCP ont été appliquées aux flux souhaités, on tape la commande **sudo ovs-ofctl dump-flows S1** pour obtenir le résultat suivant :

```

mininet> sh sudo ovs-ofctl -O OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b00000000000001, duration=1728.123s, table=0, n_packets=347, n_bytes=29495, priority=10
 0,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2a0000000000000382, duration=12.711s, table=0, n_packets=12, n_bytes=840, idle_timeout=600
 , hard_timeout=300, priority=10,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:1
 cookie=0x2a0000000000000383, duration=12.711s, table=0, n_packets=12, n_bytes=840, idle_timeout=600
 , hard_timeout=300, priority=10,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:2

```

Figure 3. 29: Résultats des valeurs DSCP appliquées



La figure 3.28 nous montre bien les valeurs DSCP que nous avons appliquées dans notre QoS.

3.2.1.3. Installation de flux avec OpenFlow Manager

a. OpenFlow Manager

OpenFlow Manager (OFM) est une application développée pour s'exécuter sur ODL afin de visualiser les topologies OpenFlow (OF), de programmer les chemins OF et de collecter les statistiques OF. OFM utilise l'API RESTCONF sur l'interface nord pour la communication avec OpenDaylight.

L'architecture OFM se présente comme suit :

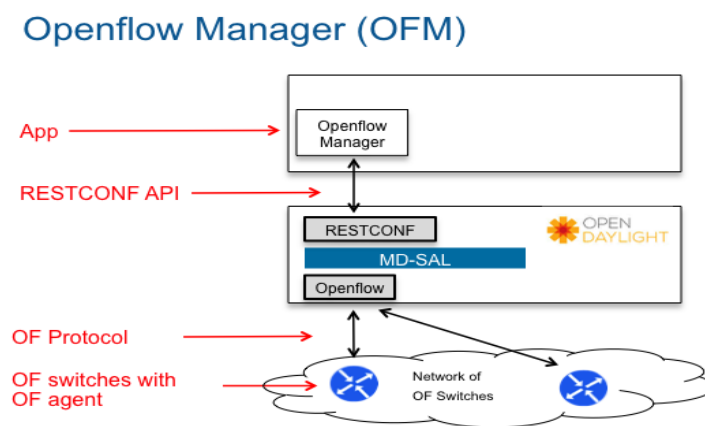


Figure 3. 30: Architecture OpenFlow Manager

De haut en bas, nous avons un réseau de commutateurs OpenFlow utilisant un paradigme de transfert "Match/Action" pour exécuter des opérations de commutation de flux à travers le réseau. Les commutateurs OpenFlow prennent en charge un agent OpenFlow et utilisent MPLS pour les paquets de commutation d'étiquettes sur le réseau, et OSPF ou ISIS pour maintenir et distribuer la base de données de topologie (état des liens) entre tous les routeurs du réseau. L'un des routeurs est un BGP-LS et il transporte une copie de la base de données de topologie vers le contrôleur ODL. Les routeurs exécuteront également PCEP (signifie le protocole d'élément de calcul de chemin) utilisé par le contrôleur ODL pour demander à un routeur source de configurer un chemin conçu pour le trafic MPLS vers un routeur de destination. Ce sont des détails très spécifiques au réseau et au protocole, que franchement l'utilisateur final peut ne pas connaître ou dont il ne se soucie pas.



À l'intérieur d'ODL, il existe des modèles YANG de la topologie du réseau et de la manière de programmer les flux sur le réseau. La couche d'adaptation de service pilotée par modèle (MD-SAL) prend ces modèles et génère automatiquement un ensemble d'API REST (appelées RESTCONF) que les applications peuvent appeler. OFM est l'application qui appelle les API RESTCONF pour récupérer l'inventaire des commutateurs OF ainsi que pour programmer les flux et collecter les statistiques. L'autre élément clé ici est l'interface utilisateur basée sur HTML5/CSS/Javascript utilisant divers Frameworks open source, notamment AngularJS et NeXt. OFM est accessible via un navigateur Web tel que Chrome. [27]

b. Installation de OFM

Il existe deux méthodes pour installer et faire fonctionner OFM avec une instance d'ODL :

- Méthode Vagrant en allant sur la page Vagrant.
- Installation pas à pas d'OFM, ODL et mininet (réseau virtuel OpenFlow).

Nous choisissons la deuxième méthode, l'installation pas à pas étant donné que notre projet est réalisé dans un environnement réseau virtuel OpenFlow.

Après avoir terminé l'installation d'OFM, nous démarrons le serveur Web puis nous accédons à l'interface GUI. Sur l'interface GUI nous pouvons voir notre topologie réseau sur l'onglet Basic view :

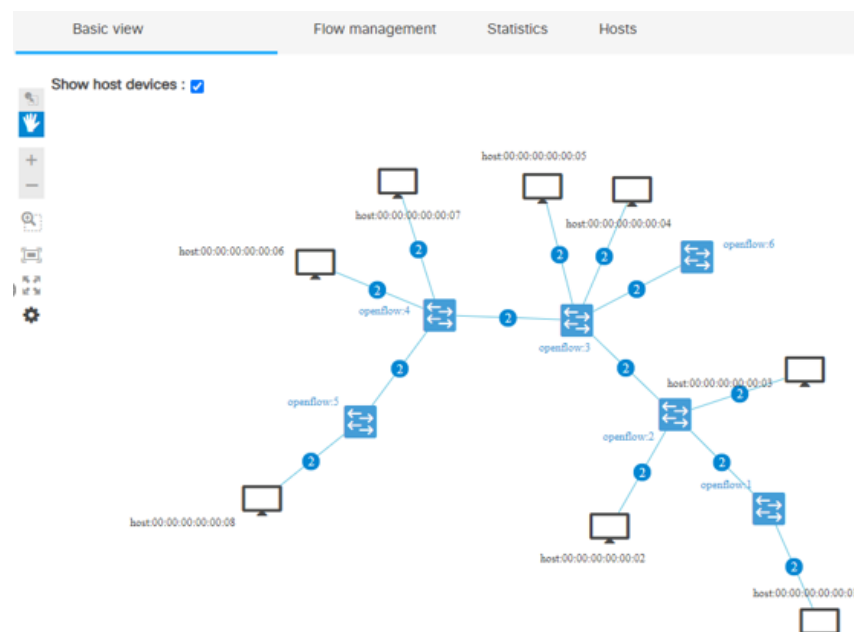
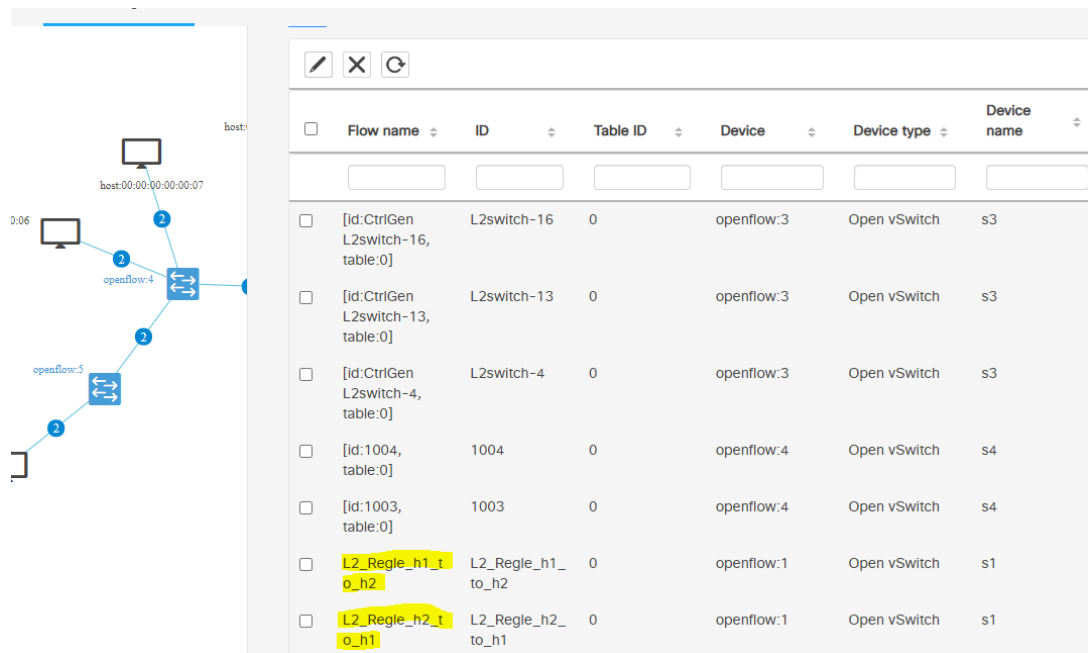


Figure 3. 31: Vue topologie réseau via GUI OFM

c. Configuration des flux

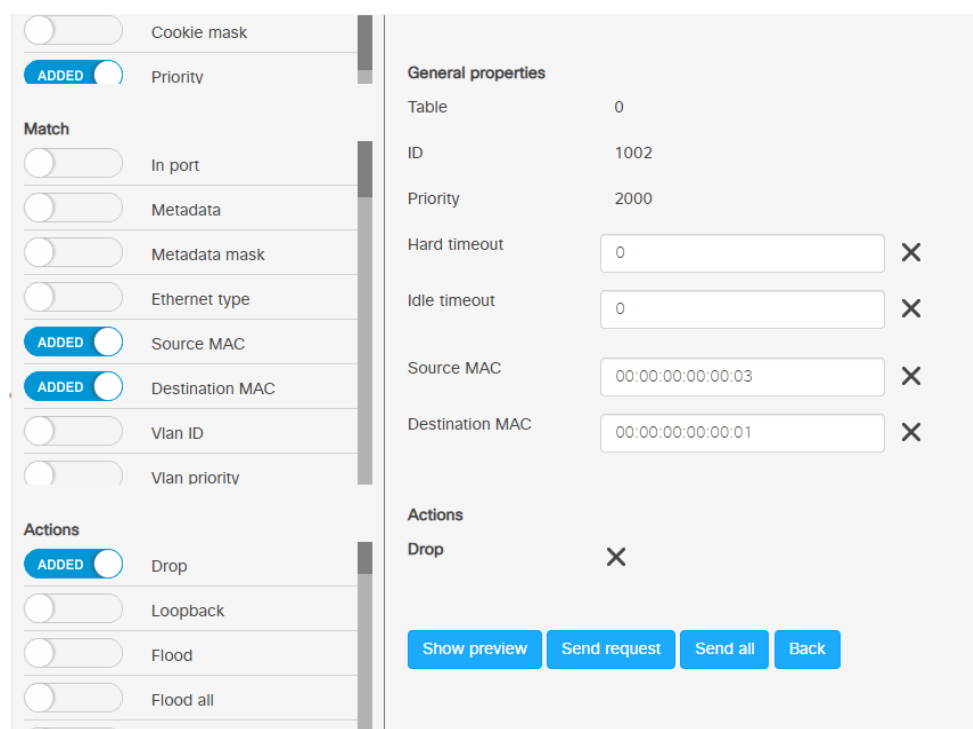
En ce qui concerne la programmation de flux, OFM supporte toutes les actions et identifications de paquets disponibles sur OpenFlow 1.3. Pour créer un nouveau flux il suffit d'aller sur l'onglet Flow Management. Sur ce même onglet se trouvent tous les flux que nous avons envoyé au contrôleur avec Postman :



<input type="checkbox"/>	Flow name	ID	Table ID	Device	Device type	Device name
<input type="checkbox"/>	[id:CtrlGen L2switch-16, table:0]	L2switch-16	0	openflow:3	Open vSwitch	s3
<input type="checkbox"/>	[id:CtrlGen L2switch-13, table:0]	L2switch-13	0	openflow:3	Open vSwitch	s3
<input type="checkbox"/>	[id:CtrlGen L2switch-4, table:0]	L2switch-4	0	openflow:3	Open vSwitch	s3
<input type="checkbox"/>	[id:1004, table:0]	1004	0	openflow:4	Open vSwitch	s4
<input type="checkbox"/>	[id:1003, table:0]	1003	0	openflow:4	Open vSwitch	s4
<input type="checkbox"/>	L2_Regle_h1_to_h2	L2_Regle_h1_to_h2	0	openflow:1	Open vSwitch	s1
<input type="checkbox"/>	L2_Regle_h2_to_h1	L2_Regle_h2_to_h1	0	openflow:1	Open vSwitch	s1

Figure 3. 32: Liste des flux sur OFM

Passons maintenant à la programmation de flux :



☐ Cookie mask
☒ **ADDED** Priority
Match
☐ In port
☐ Metadata
☐ Metadata mask
☐ Ethernet type
☒ **ADDED** Source MAC
☒ **ADDED** Destination MAC
☐ Vlan ID
☐ Vlan priority
Actions
☒ **ADDED** Drop
☐ Loopback
☐ Flood
☐ Flood all

General properties
 Table 0
 ID 1002
 Priority 2000
 Hard timeout 0 ✕
 Idle timeout 0 ✕
 Source MAC 00:00:00:00:00:03 ✕
 Destination MAC 00:00:00:00:00:01 ✕
Actions
 Drop ✕





Figure 3. 33: Programmation du flux de h1 à h3

Le flux que nous venons de configurer bloque la communication entre les hôtes h1 et h3, mais aussi entre h6, h7, h8 et le reste du réseau. Nous faisons le test pour voir :

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X h4 h5 X X X
h2 -> h1 h3 h4 h5 X X X
h3 -> X h2 h4 h5 X X X
h4 -> h1 h2 h3 h5 X X X
h5 -> h1 h2 h3 h4 X X X
h6 -> X X X X X h7 X
h7 -> X X X X X h6 X
h8 -> X X X X X X X
*** Results: 64% dropped (20/56 received)
mininet>
```

Figure 3. 34: Ping de vérification des flux entre configurés

On peut constater sur la figure ci-dessus que le réseau se comporte comme prévu en termes d'accessibilité, ce qui confirme la fiabilité des restrictions appliquées. La suite sur la programmation de flux est disponible en annexe.

3.2.2. Déploiement d'un réseau de superposition de type VXLAN

3.2.2.1. Définition VXLAN

Le VXLAN est une technologie de virtualisation réseau qui vise à résoudre des problèmes d'évolutivité associés au déploiement du Cloud Computing. Il utilise une technique d'encapsulation proche du VLAN et permet d'encapsuler des trames Ethernet de couche 2 sur un réseau de couche 3.

3.2.2.2. Principe de fonctionnement

VXLAN est essentiellement une technologie de tunnellation. Il établit un



tunnel logique sur le réseau IP entre les périphériques réseau source et de destination pour encapsuler les paquets côté utilisateur et les transmettre via le tunnel.

Sur le réseau VXLAN, certains nouveaux éléments sont présents tels que VTEP et VNI. Ils ne sont pas inclus dans les réseaux de centres de données traditionnels. Un VTEP est un périphérique sur un réseau VXLAN. Il s'agit du point de départ ou d'arrivée d'un tunnel VXLAN, qui encapsule et décapsule respectivement les trames de données utilisateur d'origine.

Un VTEP peut être un périphérique réseau indépendant ou un commutateur virtuel déployé sur un serveur. Le VTEP source encapsule les trames de données d'origine envoyées par le serveur source dans des paquets VXLAN et transmet les paquets VXLAN au VTEP de destination sur le réseau IP. Le VTEP de destination désencapsule ensuite les paquets VXLAN dans les trames de données d'origine et transmet les trames au serveur de destination.

Chaque sous-réseau de couche 2 a un identifiant de réseau VXLAN (VNI) unique qui segmente le trafic. Comme le champ ID VLAN dans une trame Ethernet n'a que 12 bits, le VLAN ne peut pas répondre aux exigences d'isolation sur les réseaux de centres de données. L'émergence de VNI est précisément de résoudre ce problème. Un VNI est un identifiant d'utilisateur similaire à un ID de VLAN. Les machines virtuelles avec différents VNI ne peuvent pas communiquer au niveau de la couche 2. Le VNI étant un identifiant locataire, lors de l'encapsulation de paquets VXLAN, un VNI 24 bits est ajouté au paquet VXLAN, ce qui permet au VXLAN d'isoler un grand nombre de locataires. [20]

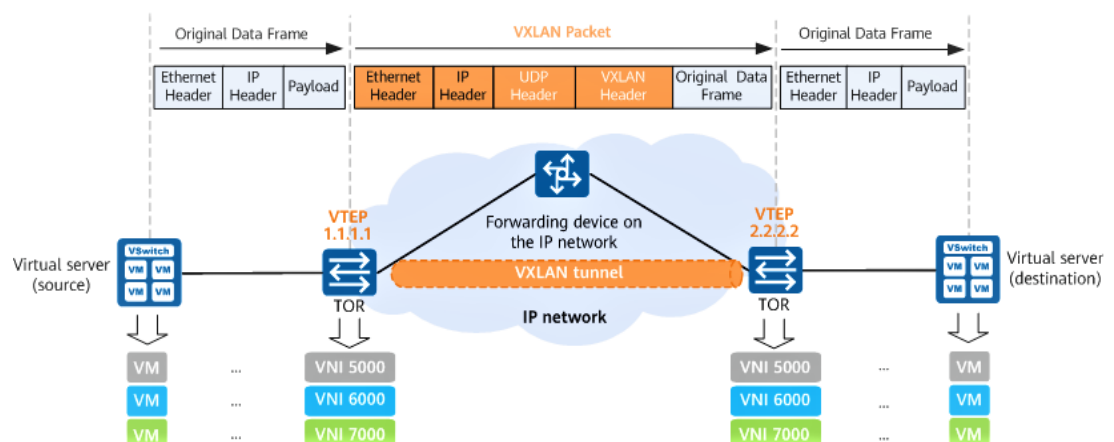


Figure 3. 35: Exemple topologie VXLAN



Un réseau VXLAN peut être déployé soit :

- ✓ En mode Unicast : il se fait en automatisant la découverte des VTEP et en s'appuyant sur la capacité d'apprentissage des VTEP pour retenir la localisation des adresses MAC source ;
- ✓ En mode Multicast : dans ce mode, la découverte des VTEP est automatique, les adresses MAC sources et leur localisation sont retenues par le VTEP mais cette méthode est envisageable sur internet ;
- ✓ Par un contrôleur SDN : le contrôleur SDN se charge de piloter les VTEP ;

Le mode Unicast n'est pas appropriée quand le nombre de VTEP augmente dans le cadre d'une entreprise à plusieurs sites par exemple.

Dans notre cas nous nous intéresserons au déploiement par un contrôleur. En passant par le contrôleur nous pourrions configurer notre réseau peu importe le nombre d'équipements présents dans le réseau et l'étendue de ce réseau.

3.2.2.3. Proposition de la topologie VXLAN

La topologie qui est proposée sera composée d'un contrôleur SDN OpenFlow et de trois (3) serveurs virtuels sous VMware sur lesquels seront installés des OpenvSwitch. Ces serveurs feront partie d'un même réseau LAN de couche trois (3). Ensuite nous une superposition des trois (3) OpenvSwitch, connectés au contrôleur, sera faite.

Le contrôleur assurera la gestion des VTEP du réseau VXLAN. Des tunnels VXLAN seront créés entre :

- OpenvSwitch1---OpenvSwitch2
- OpenvSwitch1---OpenvSwitch3
- OpenvSwitch2---OpenvSwitch3



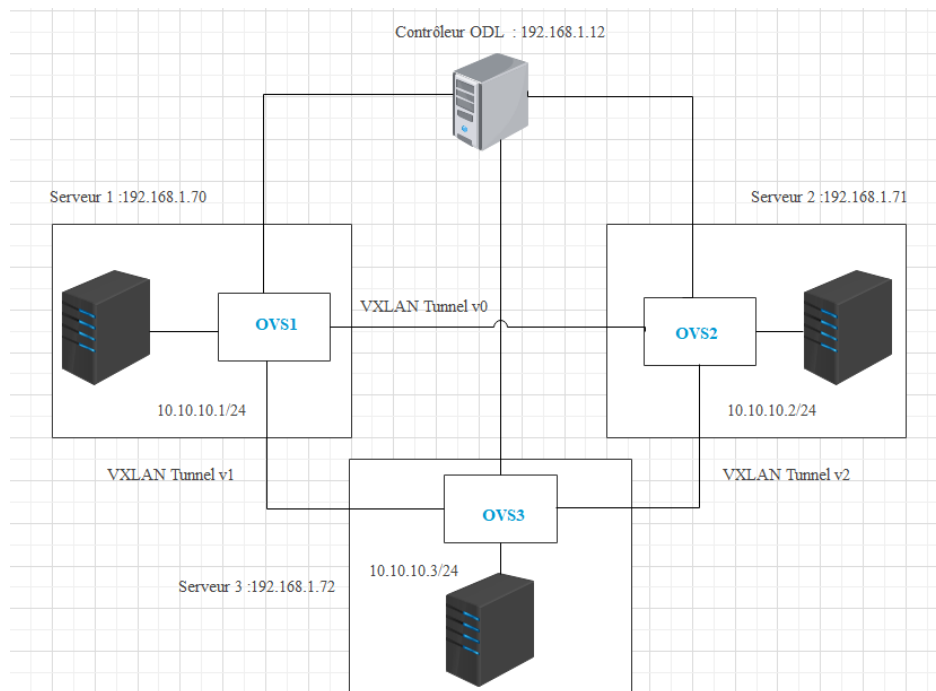



Figure 3. 36: Topologie VXLAN

a. Déploiement du contrôleur Floodlight

Tout d'abord, nous téléchargeons la machine virtuelle Floodlight V1.1 préconfigurée sur le site officiel au lien : <http://opennetlinux.org/binaries/floodlight-vm.zip>

Une fois le déploiement effectif, nous accédons à l'interface GUI avec un navigateur en tapant l'adresse IP et le port d'écoute de notre contrôleur : <http://192.168.1.12:8080/ui/index.html>


Dashboard Topology Switches Hosts
Live updates

Controller Status

Hostname:	localhost:6633
Healthy:	true
Uptime:	1325 s
JVM memory bloat:	12566672 free out of 167321600
Modules loaded:	n.f.debugcounter.DebugCounterServiceImpl, n.f.accesscontrol.ACL, n.f.testmodule.TestModule, n.f.ui.web.StaticWebRouteable, n.f.virtualnetwork.VirtualNetworkFilter, n.f.devicemanager.internal.DeviceManagerImpl, n.f.core.internal.OFSwitchManager, n.f.linkdiscovery.internal.LinkDiscoveryManager, n.f.loadbalancer.LoadBalancer, n.f.topology.TopologyManager, n.f.dhcpserver.DHCPService, n.f.forwarding.Forwarding, n.f.flowcache.FlowReconcileManager, n.f.devicemanager.internal.DefaultEntityClassifier, n.f.storage.memory.MemoryStorageSource, n.f.jython.JythonDebugInterface, n.f.restserver.RestApiServer, org.sdnplatform.sync.internal.SyncManager, n.f.learningswitch.LearningSwitch, n.f.hub.Hub, n.f.firewall.Firewall, n.f.perfmon.PktInProcessingTime, n.f.core.internal.ShutdownServiceImpl, org.sdnplatform.sync.internal.SyncTorture, n.f.staticflowentry.StaticFlowEntryPusher, n.f.threadpool.ThreadPool, n.f.core.internal.FloodlightProvider, n.f.debugevent.DebugEventService,

Switches (3)

Figure 3. 37: GUI Floodlight



L'interface GUI présente un tableau de bord, des onglets pour la topologie du réseau auquel le contrôleur est connecté, les switchs du réseau et les hôtes présents sur le réseau.

Pour une meilleure exploitation du contrôleur, nous avons fait une mise à jour et nous sommes passés à la version V1.2. Cette version offre une interface GUI plus sympa appelée Web UI comprenant :

- ✓ Les liens de connexion ;
- ✓ La topologie ;
- ✓ Le pare-feu ;
- ✓ Les ACL (Liste de Contrôle d'Accès) ;
- ✓ Les Statistiques ;
- ✓ Les détails sur l'utilisation de la mémoire ;
- ✓ Une vue sur tous les modules et ceux qui sont utilisés sur le contrôleur.

b. Configuration réseau VXLAN

Nous commençons par installer OpenvSwitch sur chacun de nos serveurs :

```
levymab@ubuntu3:~$ sudo apt install openvswitch-switch
[sudo] password for levymab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Figure 3. 38: Installation d'OpenvSwitch

Après l'installation d'OpenvSwitch, nous créons des ponts et rajoutons notre contrôleur à ces ponts :

```
levymab@ubuntu3:~$ sudo ovs-vsctl add-br switch3
```

Figure 3. 39: Création d'un pont et connexion au contrôleur SDN

Passons maintenant aux tunnels VXLAN entre nos trois (3) ponts :

```
levymab@ubuntu3:~$ sudo ovs-vsctl add-port switch3 v1 -- set interface v1 type=
vxlan option:remote_ip=192.168.1.70 ofport_request=10
```

Figure 3. 40: Création des tunnels

Des adresses doivent être attribuées à nos différents OVS pour vérifier la communication entre les tunnels créés :



```
levymab@ubuntu3:~$ sudo ip addr add 10.10.10.3/24 dev switch3
levymab@ubuntu3:~$ sudo ifconfig switch3 up
levymab@ubuntu3:~$
```

Figure 3. 41: Configuration adresse IP OVS

```
levymab@ubuntu3:~$ ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=0.574 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=1.09 ms
^C
--- 10.10.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms

levymab@ubuntu2:~$ ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_seq=1 ttl=64 time=4.65 ms
64 bytes from 10.10.10.3: icmp_seq=2 ttl=64 time=1.25 ms
^C
--- 10.10.10.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.246/2.949/4.653/1.703 ms
levymab@ubuntu2:~$
```

Figure 3. 42: Test communication entre OVS3 et OVS2

Nous allons voir un récapitulatif de la configuration qui a été faite sur l'OVS3. Nous obtenons ces informations en tapons la commande : ***sudo ovs-vsctl show***. [28]

c. Gestion du réseau VXLAN

La gestion du réseau de superposition mis en place, peut être assurée depuis l'interface Webui :

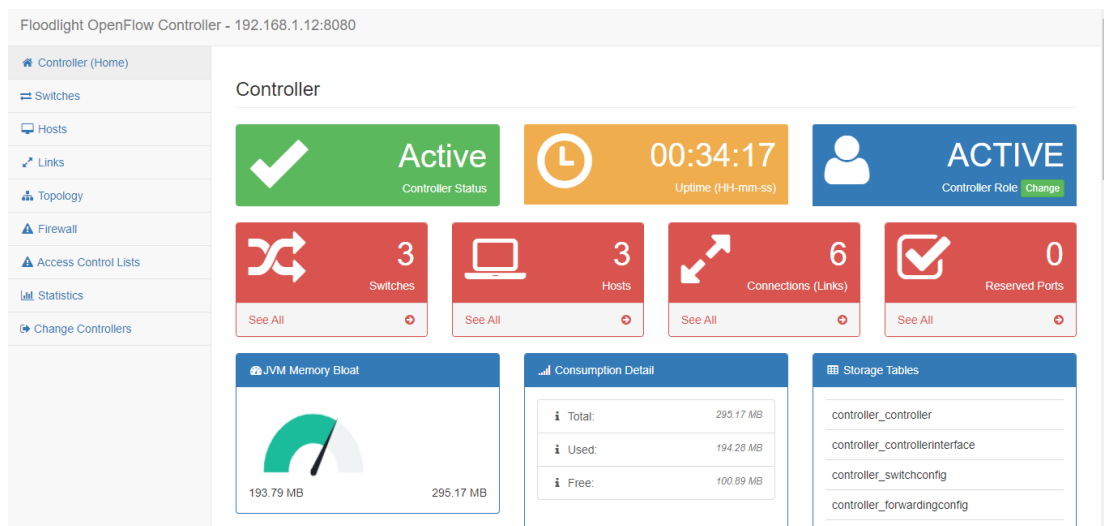


Figure 3. 43 : Interface WebUI Floodlight

Nous pouvons regarder la topologie du réseau :



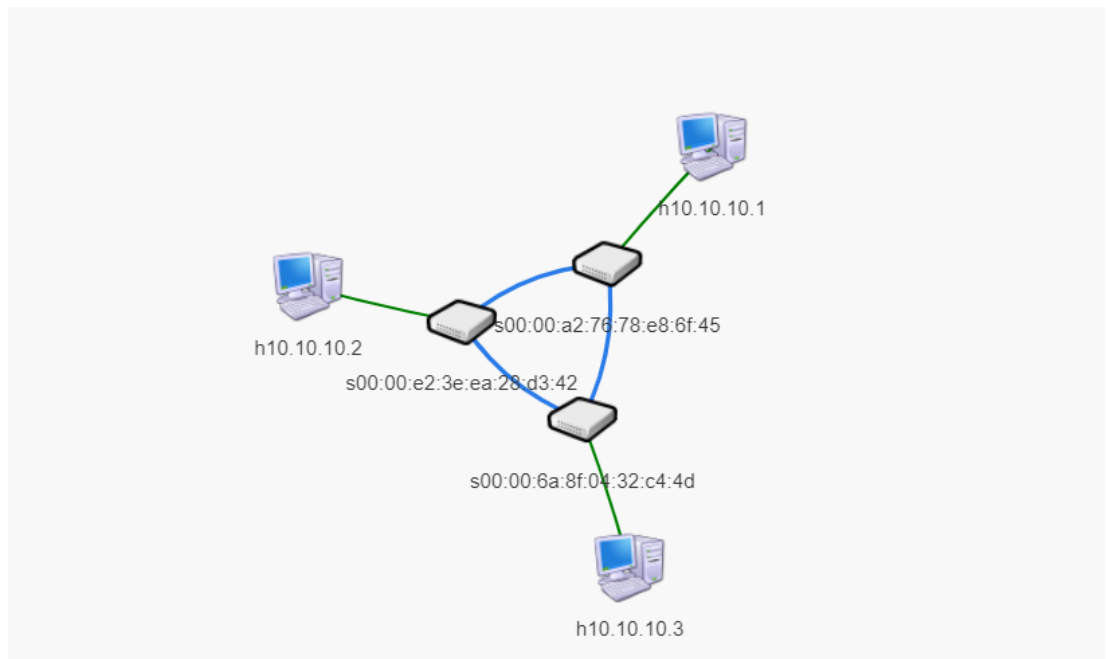


Figure 3. 44: Topologie réseau VXLAN via Web UI

Nous pouvons configurer les ACL pour autoriser ou bloquer certains flux. On commence par créer une règle interdisant la communication entre le PC connecté au Switch 2 et le PC connecté au Switch 3 :

Figure 3. 45: Création règle ACL

On clique sur créer. On peut ensuite voir la règle que nous venons de créer :



Control List										
	Source	Dest	Source IP	Mask	Dest IP	Mask	Prot.	Dest TP	Act.	Delete
	10.10.10.2/24	10.10.10.3/24	168430082	24	168430083	24	1	0	DENY	<button>Delete</button>

Showing 1 to 1 of 1 entries

Figure 3. 46: Vue ACL créée

Maintenant nous pouvons tester :

```

3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group de
   link/ether 36:d4:bd:75:60:04 brd ff:ff:ff:ff:ff:ff
5: switch2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state U
   link/ether e2:3e:ea:28:d3:42 brd ff:ff:ff:ff:ff:ff
   inet 10.10.10.2/24 scope global switch2
       valid_lft forever preferred_lft forever
   inet6 fe80::e03e:eaff:fe28:d342/64 scope link
       valid_lft forever preferred_lft forever
8: vxlan_sys_4789: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65535 qdisc noqueue
   link/ether fe:a5:86:54:82:52 brd ff:ff:ff:ff:ff:ff
levy@ubuntu:~$ ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
From 10.10.10.2 icmp_seq=10 Destination Host Unreachable
From 10.10.10.2 icmp_seq=11 Destination Host Unreachable
From 10.10.10.2 icmp_seq=12 Destination Host Unreachable
From 10.10.10.2 icmp_seq=13 Destination Host Unreachable
From 10.10.10.2 icmp_seq=14 Destination Host Unreachable
From 10.10.10.2 icmp_seq=15 Destination Host Unreachable
From 10.10.10.2 icmp_seq=16 Destination Host Unreachable

```

Figure 3. 47: Test communication bloqué

On voit bien sur la figure 3.47 que la communication entre 10.10.10.2/24 et 10.10.10.3/24 est bloquée.

Conclusion

Le chapitre portant sur l'implémentation de la solution de réseau à définition logicielle nous a permis d'utiliser les ressources et outils pratiques nécessaires au déploiement d'un contrôleur SDN. De cette implémentation, nous avons pu toucher l'ensemble des aspects techniques (virtualisation, communication avec APIs, QoS, files d'attente, bridges OVS, ...) liés à la mise en place et l'exploitation d'un environnement réseau SDN.



CONCLUSION GÉNÉRALE

Le SDN a évolué très rapidement depuis l'observation initiale que les ressources de calcul et de réseau ont changé d'une manière ayant permis et même recommandé la réorganisation du contrôle du réseau. Plus précisément, la croissance des débits de données et la réduction du temps de décision de transfert qui en résulte ont imposé des exigences très différentes à la fonction de transfert du plan de données par rapport aux exigences imposées par le traitement d'ensembles de données de plus en plus complexe. Le SDN a montré un moyen de tirer parti des modifications de capacités de réseau et de calcul pour fournir un nouveau moyen de gérer les réseaux modernes.

Ainsi, les recherches et analyses faisant l'objet de l'étude menée dans le cadre de ce projet nous ont permis de procéder étape par étape au déploiement d'une solution SDN pour une administration plus efficace et plus flexible du réseau du système hospitalier de Mouyondzi. Il est manifeste que l'étude et la mise en œuvre d'une solution SDN nécessite, une bonne connaissance des principes et règles de fonctionnement des couches du SDN mais aussi des interfaces de communication entre les différentes couches.

Toutes les installations et configurations effectuées dans ce projet, font partie des tâches et prérequis à suivre pour déployer notre contrôleur SDN pour la rénovation de l'infrastructure réseaux des hôpitaux de la ville de Mouyondzi. La solution de réseau à définition logicielle viendra répondre au besoin de gestion centralisée des réseaux, d'interconnexion et de partage de données entre les hôpitaux de ville de Mouyondzi.

Ce qui représentait le principal objectif à atteindre dans le cadre de cette étude.

Conformément à la mise en œuvre de notre solution SDN, une évaluation de ressources (open source en majorité) a été effectuée. Elle a permis de déterminer approximativement le coût d'investissement du projet et de montrer l'avantage de pouvoir la réaliser.

Du point d'espace et environnement, notre étude montre que le déploiement de la solution SDN ne requiert pas nécessairement beaucoup d'éléments physiques. Cela



permet d'éviter un encombrement dans les salles serveurs et l'utilisation de plus d'énergie électrique.

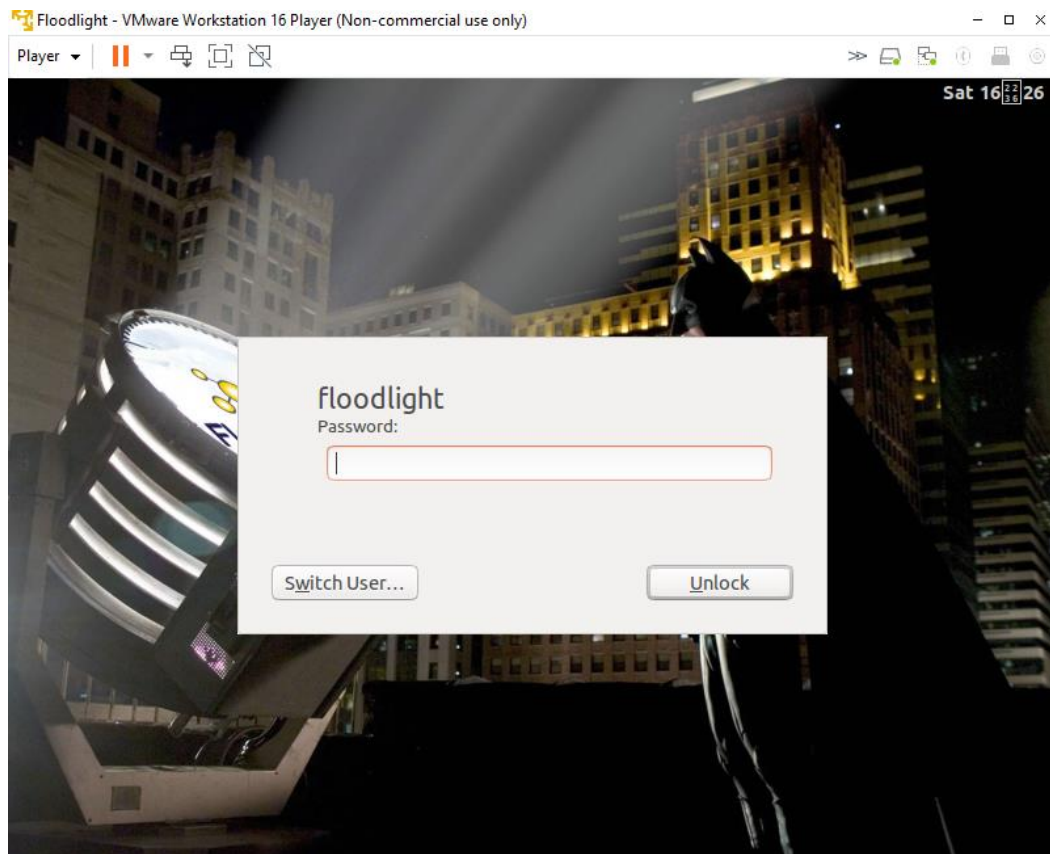
Aussi dans le sillage du développement du Big Data et les avancées de nouvelles technologies au Congo, on pourrait envisager la mise en place d'une plateforme de centralisation de données médicales.



ANNEXE A : Déploiement du contrôleur Floodlight

Le déploiement de notre contrôleur Floodlight s'est fait assez simplement. Nous avons téléchargé la VM sur le site officiel : <http://opennetlinux.org/binaries/floodlight-vm.zip>

Ensuite nous importons la machine sous Vmware, puis la démarrons :



Le nom d'utilisateur et le mot de passe sont : **floodlight**

Maintenant que la machine est lancée, nous compilons les fichiers du Floodlight :

```
floodlight@floodlight:~$ cd floodlight/
floodlight@floodlight:~/floodlight$ ant
Buildfile: /home/floodlight/floodlight/build.xml

init:

compile:
[javac] Compiling 1 source file to /home/floodlight/floodlight/target/bin

compile-test:

dist:
[jar] Building jar: /home/floodlight/floodlight/target/floodlight.jar
[jar] Building jar: /home/floodlight/floodlight/target/floodlight-test.jar


BUILD SUCCESSFUL
Total time: 19 seconds
floodlight@floodlight:~/floodlight$
```

Après la compilation, nous démarrons Floodlight :




```
floodlight@floodlight:~/floodlight$ java -jar target/floodlight.jar
16:51:42.954 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from src
/main/resources/floodlightdefault.properties
16:51:43.105 WARN [n.f.r.RestApiServer:main] HTTPS disabled; HTTPS will not be u
sed to connect to the REST API.
16:51:43.105 WARN [n.f.r.RestApiServer:main] HTTP enabled; Allowing unsecure acc
ess to REST API on port 8080.
16:51:44.184 WARN [n.f.c.i.OFSwitchManager:main] SSL disabled. Using unsecure co
nnections between Floodlight and switches.
16:51:44.184 INFO [n.f.c.i.OFSwitchManager:main] Clear switch flow tables on ini
tial handshake as master: TRUE
```

Nous avons terminé le déploiement et on peut accéder à l'interface GUI en tapons dans un navigateur : <http://192.168.1.11:8080/ui/index.html>



Dashboard Topology Switches Hosts
Live updates

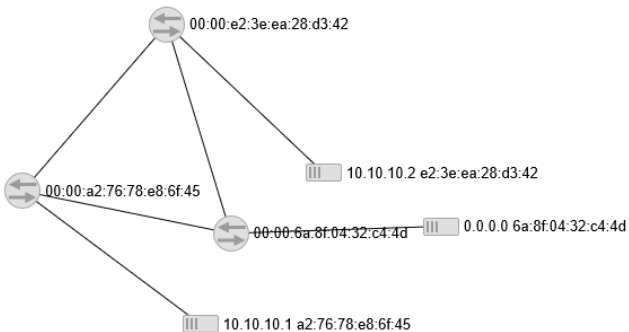
Controller Status

Hostname:	localhost:6633
Healthy:	true
Uptime:	1325 s
JVM memory bloat:	12566672 free out of 167321600
Modules loaded:	n.f.debugcounter.DebugCounterServiceImpl, n.f.accesscontrol.ACL, n.f.testmodule.TestModule, n.f.ui.web.StaticWebRouteable, n.f.virtualnetwork.VirtualNetworkFilter, n.f.devicemanager.internal.DeviceManagerImpl, n.f.core.internal.OFSwitchManager, n.f.linkdiscovery.internal.LinkDiscoveryManager, n.f.loadbalancer.LoadBalancer, n.f.topology.TopologyManager, n.f.dhcpserver.DHCPManager, n.f.forwarding.Forwarding, n.f.flowcache.FlowReconcilerManager, n.f.devicemanager.internal.DefaultEntityClassifier, n.f.storage.memory.MemoryStorageSource, n.f.jython.JythonDebugInterface, n.f.restserver.RestApiServer, org.sdnplatform.sync.internal.SyncManager, n.f.learningswitch.LearningSwitch, n.f.hub.Hub, n.f.firewall.Firewall, n.f.perfmon.PktInProcessingTime, n.f.core.internal.ShutdownServiceImpl, org.sdnplatform.sync.internal.SyncTorture, n.f.staticflowentry.StaticFlowEntryPusher, n.f.threadpool.ThreadPool, n.f.core.internal.FloodlightProvider, n.f.debugevent.DebugEventService,

Switches (3)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:a2:76:78:e8:6f:45	/192.168.1.10:57820	Nicira, Inc.	168	11819	6	29/09/2022 21:51:16
00:00:6a:8f:04:32:c4:4d	/192.168.1.22:54542	Nicira, Inc.	20949	885681	6	29/09/2022 21:42:19
00:00:e2:3e:ea:28:d3:42	/192.168.1.13:50652	Nicira, Inc.	31626	1335480	6	29/09/2022 21:42:12


Dashboard Topology Switches Hosts



Notre contrôleur est désormais opérationnel.

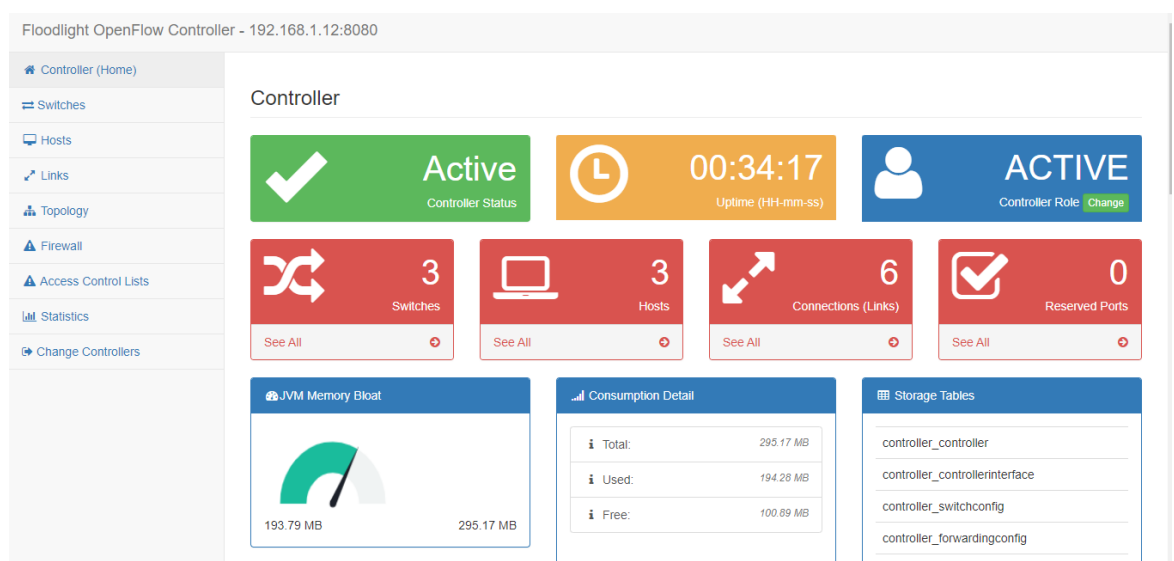


- Mise à jour de Floodlight :

Nous avons mis à jour le contrôleur en suivant la procédure suivante :

```
~/floodlight$ sudo git pull origin master
~/floodlight$ sudo git submodule update
~/floodlight$ sudo apt-get install software-properties-common
~/floodlight$ sudo add-apt-repository ppa:linuxuprising/java
```

Une fois la mise à jour terminée, nous obtenons une interface GUI plus intéressante :



Nous avons une vue sur les switches, les PCs, le pare-feu, etc ...

Switches

Switches Connected		
Switch ID	IPv4 Address	Connected Since
00:00:6a:8f:04:32:c4:4d	/192.168.1.22:54234	Sat Oct 15 2022 09:56:31 GMT+0000 (heure moyenne de Greenwich)
00:00:a2:76:78:e8:6f:45	/192.168.1.10:50246	Sat Oct 15 2022 09:51:08 GMT+0000 (heure moyenne de Greenwich)
00:00:e2:3e:ea:28:d3:42	/192.168.1.13:47186	Sat Oct 15 2022 09:48:16 GMT+0000 (heure moyenne de Greenwich)
Showing 1 to 3 of 3 entries		

Switch Roles	
Switch MAC	Role
00:00:e2:3e:ea:28:d3:42	MASTER
00:00:a2:76:78:e8:6f:45	MASTER
00:00:6a:8f:04:32:c4:4d	MASTER

Sur la capture ci-dessus, on peut voir les adresses Mac et IP des switches et la durée depuis laquelle les switches sont connectés au contrôleur.

Pour afficher les ordinateurs présents sur le réseau VXLAN on clique sur **Hosts** :

Hosts

Hosts Connected					
MAC	IPv4 Address	IPv6 Address	Switch	Port	Last Seen
6a:8f:04:32:c4:4d	10.10.10.3	fe80::688f:4ff:fe32:c44d	00:00:6a:8f:04:32:c4:4d	local	1665828075208
a2:76:78:e8:6f:45	10.10.10.1	fe80::a076:78ff:fee8:6f45	00:00:6a:8f:04:32:c4:4d000:00:a2:76:78:e8:6f:45	10local	1665828125188
e2:3e:ea:28:d3:42	10.10.10.2	fe80::e03e:eaff:fe28:d342	00:00:e2:3e:ea:28:d3:42000:00:6a:8f:04:32:c4:4d	local011	1665828104902

Showing 1 to 3 of 3 entries



ANNEXE B : Configuration du Traffic Shaping (QoS et file d'attente)

Cette annexe présente la procédure complète de configuration de la QoS et de la file d'attente avec les différentes bandes passantes.

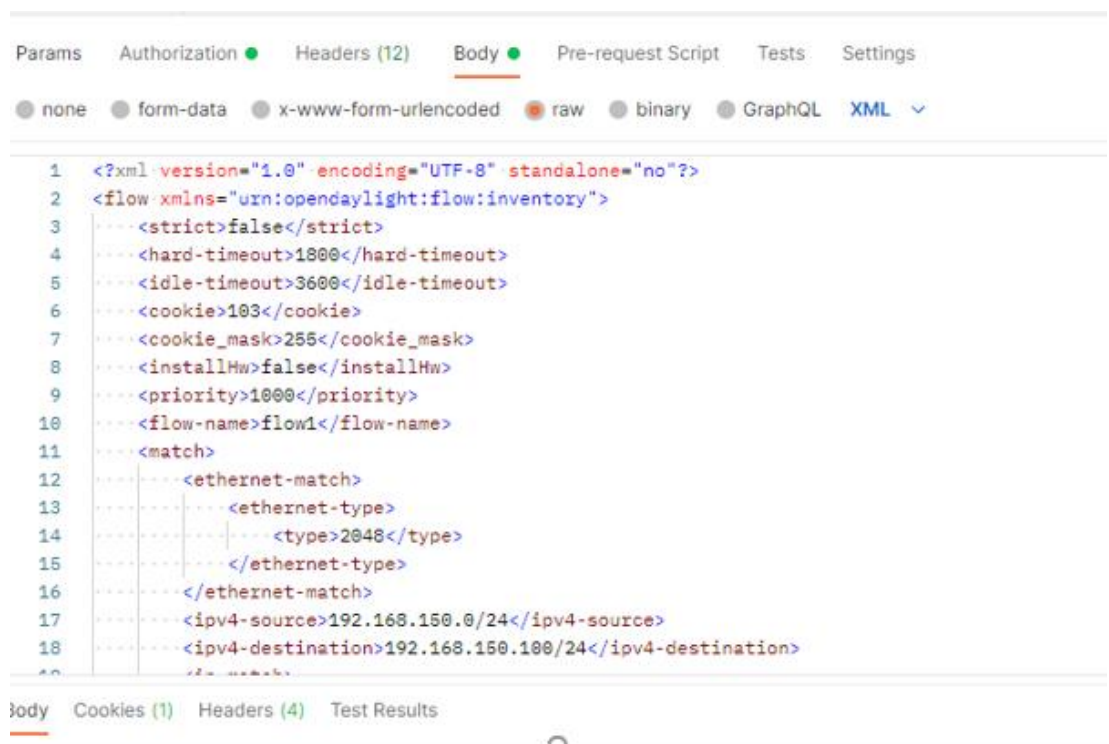
- QoS appliquée au port eth1 du switch 1 :

```
mininet> sh sudo ovs-vsctl set port s1-eth1 qos=@newqos -- --id=@newqos create qos type=linux-htb other-config:max-rate=50000000 queues=1=@q1 -- --id=@q1 create queue other-config:max-rate=50000000 other-config:min-rate=15000000
07d6b593-7eff-4a7e-be18-c0235d0cc3c9
f0758377-ee13-447b-b45d-dff88701acab
mininet> _
```

- Mappage du flux avec la QoS en utilisant Postman :

Ce mappage consiste à envoyer du code au contrôleur via l'API Rest Northbound à l'adresse :

<http://192.168.1.12:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1>



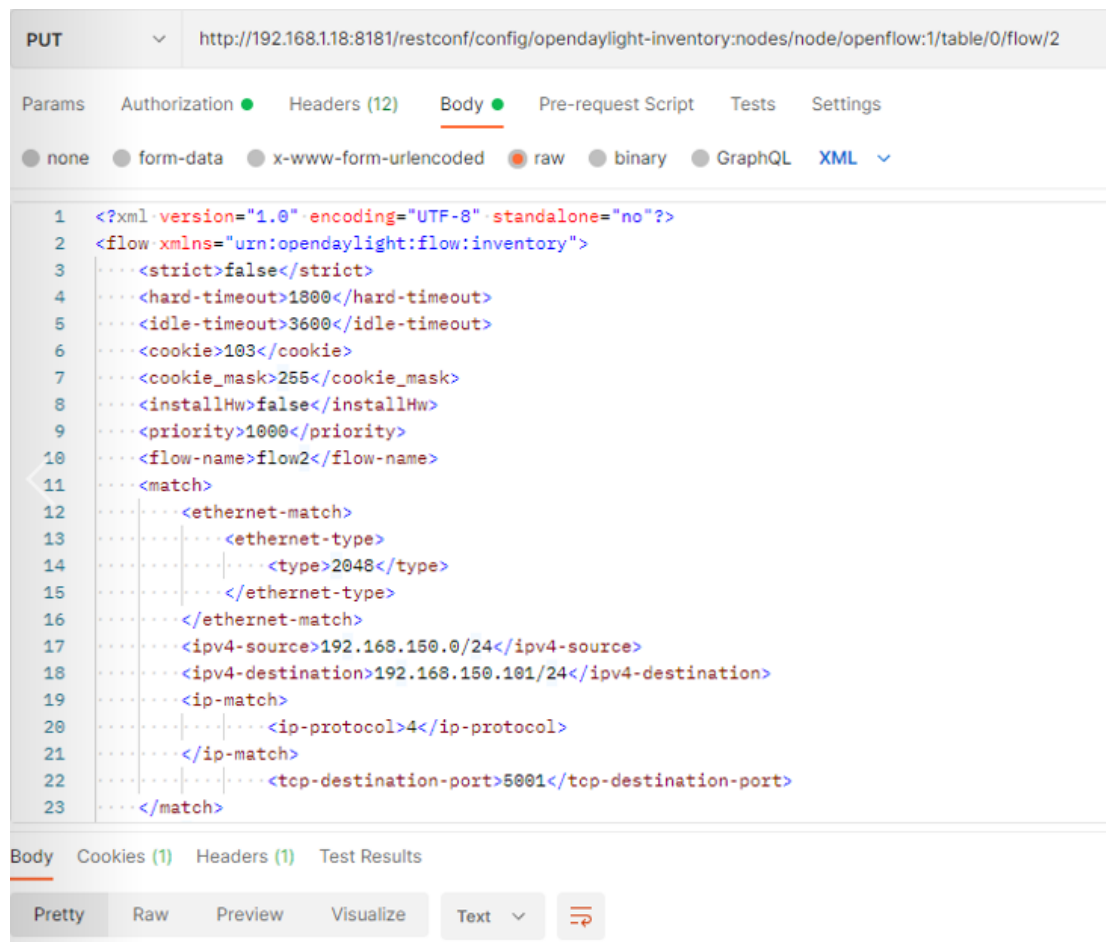
- QoS appliquée au port eth2 du switch1 :

```
mininet> sh sudo ovs-vsctl set port s1-eth2 qos=@newqos2 -- --id=@newqos2 create qos type=linux-htb other-config:max-rate=50000000 queues=2=@q2 -- --id=@q2 create queue other-config:max-rate=50000000 other-config:min-rate=15000000
ecde1cb0-785a-4382-befb-0e44fb16749b
803ca514-01d1-4d56-b3af-be680bad4032
mininet> _
```

- Mappage du flux à la QoS en se servant de Postman :



Nous allons envoyer du code XML du flux 2 à la même adresse web :
<http://192.168.1.12:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:1/table/0/flow/2>



Après avoir appliqué la QoS avec succès, on peut le vérifier sur Mininet avec la commande : `ovs-vsctl list qos`

```

mininet> sh ovs-vsctl list qos
    _uuid                : 7e6f95d2-57c6-4a9b-a571-228558458142
    external_ids          : {}
    other_config          : {max-rate="50000000"}
    queues                : {1=5d79d9e0-0c1c-40fc-9988-3b0e64a1e1f8}
    type                  : linux-htb

    _uuid                : 7f1514aa-99cd-49cd-9a8c-6a5a49c3926e
    external_ids          : {}
    other_config          : {max-rate="50000000"}
    queues                : {2=745a73b5-8e44-48b8-8ba3-8bc30d0a7a2d}
    type                  : linux-htb
mininet>

```

Ci-dessus, nous avons le résultat après la création de la QoS sur les ports s1-eth1 et s1-eth2 du switch 1. Comme on peut le constater, les deux QoS sont appliquées



avec une bande passante maximale de 50 mb/s. Il y'a également une file d'attente q1 attachée à la première QoS et une file d'attente q2 attachée à la seconde QoS.

Nous pouvons vérifier la configuration de nos files d'attente : **ovs-vsctl list queue**

```
mininet> sh ovs-vsctl list queue
_uuid          : 5d79d9e0-0c1c-40fc-9988-3b0e64a1e1f8
dscp           : []
external_ids   : {}
other_config   : {max-rate="50000000", min-rate="15000000"}

_uuid          : 745a73b5-8e44-48b8-8ba3-8bc30d0a7a2d
dscp           : []
external_ids   : {}
other_config   : {max-rate="50000000", min-rate="15000000"}
mininet> _
```

Nous avons appliqué à nos files d'attente un taux de transfert maximum de 50 mb/s et un taux de transfert de 15 mb/s minimum.

Maintenant nous observons les résultats des tests :

```
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['38.8 Mbits/sec', '46.7 Mbits/sec']
mininet> iperf h3 h1
*** Iperf: testing TCP bandwidth between h3 and h1
.*** Results: ['39.7 Mbits/sec', '48.0 Mbits/sec']
mininet> iperf h2 h4
*** Iperf: testing TCP bandwidth between h2 and h4
*** Results: ['9.37 Gbits/sec', '9.34 Gbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
.*** Results: ['41.0 Mbits/sec', '49.9 Mbits/sec']
```

On remarque que le test de performance montre que la communication entre h1 et h3 a soumise à une bande passante maximum d'environ 50 mb/s. L'hôte h1 étant connecté au switch s1 au port eth1, switch s1 au switch s2 au port eth2 et l'hôte h3 connecté au switch s2 alors la QoS configurée s'applique bien.

La communication entre les hôtes h2 et h4 indique une bande passante de près de 10 gb/s. Cela confirme donc que le trafic shaping que nous avons configuré fonctionne bien



ANNEXE C : Implémentation du DSCP

Nous avons procédé par injection de la QoS en utilisant l'API Rest plutôt que d'injecter la QoS directement sur la CLI du contrôleur ODL.

- Envoie de la règle de l'hôte h2 à l'hôte h1 avec Postman :

On envoie du code XML à l'adresse :

<http://192.168.1.12:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:1/table/0/flow/3>

The screenshot shows the Postman interface with the 'Body' tab selected. The content type is set to 'XML'. The XML payload is as follows:

```

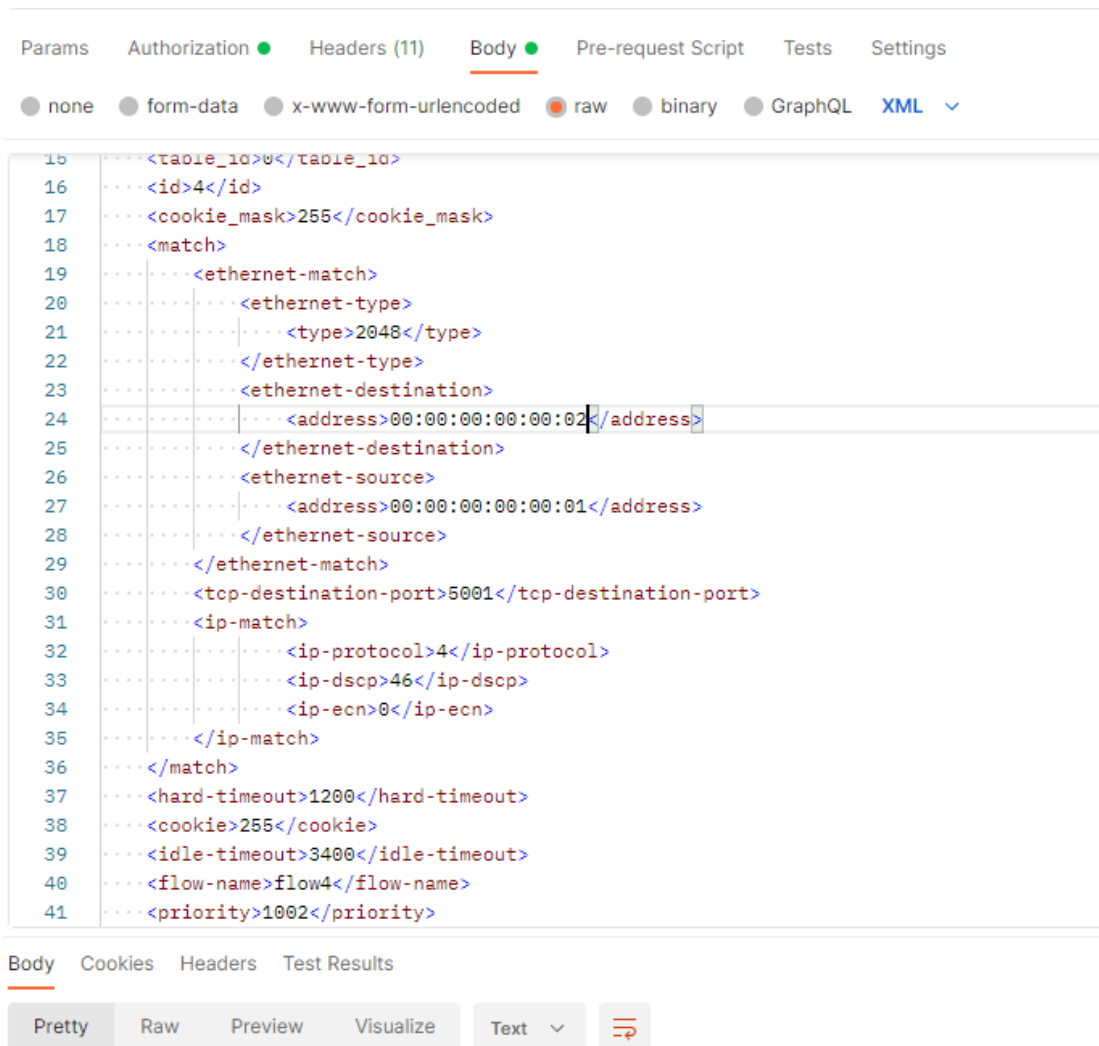
15 <table_id>0</table_id>
16 <id>3</id>
17 <cookie_mask>255</cookie_mask>
18 <match>
19 <ethernet-match>
20 <ethernet-type>
21 <type>2048</type>
22 </ethernet-type>
23 <ethernet-destination>
24 <address>00:00:00:00:00:01</address>
25 </ethernet-destination>
26 <ethernet-source>
27 <address>00:00:00:00:00:02</address>
28 </ethernet-source>
29 </ethernet-match>
30 <tcp-destination-port>5001</tcp-destination-port>
31 <ip-match>
32 <ip-protocol>4</ip-protocol>
33 <ip-dscp>46</ip-dscp>
34 <ip-ecn>0</ip-ecn>
35 </ip-match>
36 </match>
37 <hard-timeout>1200</hard-timeout>
38 <cookie>255</cookie>
39 <idle-timeout>3400</idle-timeout>
40 <flow-name>flow3</flow-name>
41 </flow>
  
```

- Envoie de la règle de l'hôte h1 à l'hôte h2 avec Postman :

Nous envoyons le code à l'adresse :

<http://192.168.1.12:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:1/table/0/flow/4>





Une fois la configuration DSCP terminée, nous pouvons afficher les valeurs de flux :

```

mininet> sh sudo ovs-ofctl -O OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b00000000000001, duration=1728.123s, table=0, n_packets=347, n_bytes=29495, priority=10
 0,d_l_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2a0000000000000382, duration=12.711s, table=0, n_packets=12, n_bytes=840, idle_timeout=600
 , hard_timeout=300, priority=10,d_l_src=00:00:00:00:00:01,d_l_dst=00:00:00:00:00:02 actions=output:1
 cookie=0x2a0000000000000383, duration=12.711s, table=0, n_packets=12, n_bytes=840, idle_timeout=600
 , hard_timeout=300, priority=10,d_l_src=00:00:00:00:00:02,d_l_dst=00:00:00:00:00:01 actions=output:2
    
```

On voit bien, que notre règle a été appliquée entre les hôtes h1 et h2 avec leurs adresses MAC respectives : 00 :00 :00 :00 :00 :01 et 00 :00 :00 :00 :00 :04. Nous avons appliqué une valeur critique de DSCP sur le code <ip-dscp>46</ip-dscp> et une valeur ECN <ip-ecn>0</ip-ecn> ce qui fera en sorte qu'en cas de congestion le flux passant entre les PC1 et PC4 ne sera pas rejeté.



ANNEXE D : Installation d'OpenFlow Manager

Pour installer OFM il faut au préalable installer Nodejs (environnement d'exécution JavaScript côté serveur). Nodejs nous permettra de faire fonctionner grunt :

```
root@ubuntu:~# apt install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Puis on télécharge OFM en se servant du clonage du répertoire github de l'application :

```
root@ubuntu:~# git clone https://github.com/CiscoDevNet/opendaylight-openflow-App.git
```

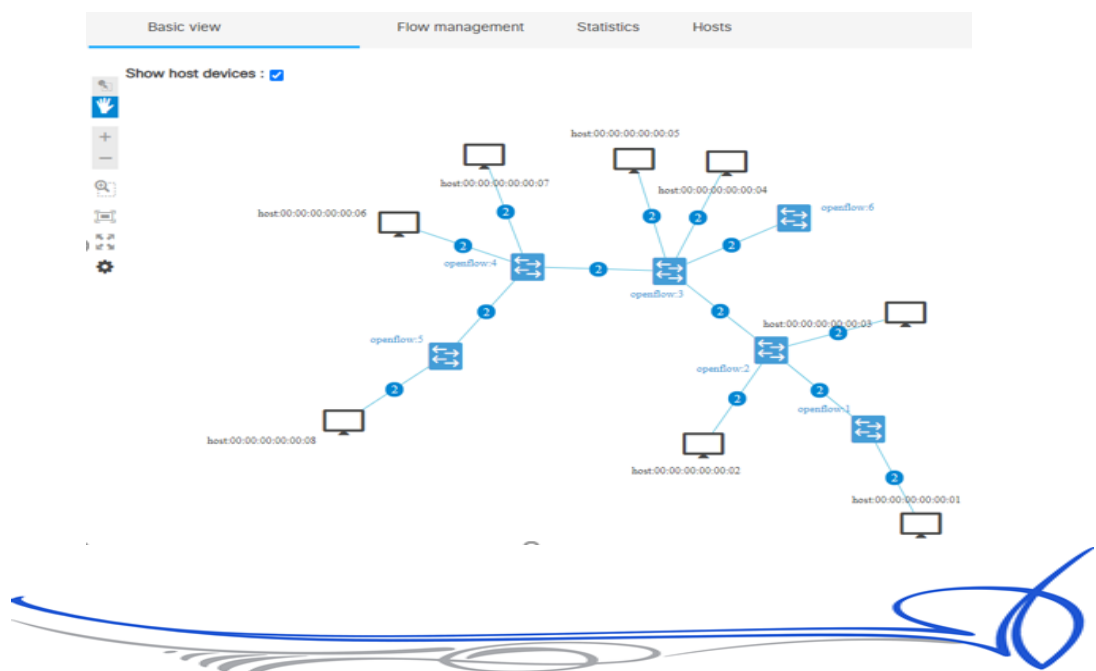
Après avoir téléchargé OFM, on installe **grunt** :

```
root@ubuntu:~# npm install -g grunt-cli
/usr/local/bin/grunt -> /usr/local/lib/node_modules/grunt-cli/bin/grunt
+ grunt-cli@1.4.3
updated 1 package in 26.831s
```

Une fois l'installation terminée, on utilise la commande grunt pour démarrer une instance du serveur web OFM :

```
root@ubuntu:~# cd opendaylight-openflow-App/
root@ubuntu:~/opendaylight-openflow-App# grunt
Running "connect:dev" (connect) task
Waiting forever...
Started connect web server on http://localhost:9000
```

Maintenant que OFM est installé, nous accédons à l'interface d'administration en mettons l'adresse IP de la machine sur laquelle est installé OFM et le numéro de port **9000**. Lorsque nous accédons à l'application nous pouvons voir la topologie de notre réseau :



ANNEXE E : Installation de flux avec OFM

Pour créer un nouveau flux il suffit d'aller sur l'onglet Flow Management. On trouve sur ce même onglet tous les flux que nous avons envoyé au contrôleur avec Postman.

- Flux bloquant la communication entre h1 et h3 :

The screenshot shows the OFM Flow Management interface. On the left, there is a sidebar with various configuration options. The 'Match' section includes 'In port', 'Metadata', 'Metadata mask', 'Ethernet type', 'Source MAC' (marked 'ADDED'), 'Destination MAC' (marked 'ADDED'), 'Vlan ID', and 'Vlan priority'. The 'Actions' section includes 'Drop' (marked 'ADDED'), 'Loopback', 'Flood', and 'Flood all'. The main panel displays the 'General properties' for the flow: Table 0, ID 1002, Priority 2000, Hard timeout 0, Idle timeout 0, Source MAC 00:00:00:00:00:03, and Destination MAC 00:00:00:00:00:01. The 'Actions' section shows 'Drop' with a close button. At the bottom, there are buttons for 'Show preview', 'Send request', 'Send all', and 'Back'.

- Flux autorisant la communication entre h6 et h7 :

The screenshot shows the OFM Flow Management interface for a different flow. The 'Match' section includes 'In port', 'Metadata', 'Metadata mask', 'Ethernet type', 'Source MAC' (marked 'ADDED'), 'Destination MAC' (marked 'ADDED'), 'Vlan ID', and 'Vlan priority'. The 'Actions' section includes 'Drop', 'Loopback', 'Flood', and 'Flood all'. The main panel displays the 'General properties' for the flow: Table 0, ID 1007, Priority 2000, Hard timeout 0, Idle timeout 0, Source MAC 00:00:00:00:00:07, and Destination MAC 00:00:00:00:00:06. The 'Actions' section shows 'Normal' with a close button and a 'Maximum length' of 6566. At the bottom, there are buttons for 'Show preview', 'Send request', 'Send all', and 'Back'.



- Flux bloquant la communication entre les hôtes h6 et h7 avec le reste du réseau :

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X h4 h5 X X X
h2 -> h1 h3 h4 h5 X X X
h3 -> X h2 h4 h5 X X X
h4 -> h1 h2 h3 h5 X X X
h5 -> h1 h2 h3 h4 X X X
h6 -> X X X X X h7 X
h7 -> X X X X X h6 X
h8 -> X X X X X X X
*** Results: 64% dropped (20/56 received)
mininet>
```

L'installation du flux qui bloque la communication entre l'hôte h7 et le du réseau bloque également la communication entre h8 et le reste du réseau.



ANNEXE F : Configuration du réseau de superposition VXLAN

Dans cette annexe nous poursuivons la configuration sur les serveurs Ubuntu 1 et Ubuntu2.

- Configuration sur Ubuntu 1 dont l'adresse IP est 192.168.1.70

```
levymab@ubuntu1:~$ sudo apt install openvswitch-switch
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Nous créons le pont switch 1 :

```
levymab@ubuntu1:~$ sudo ovs-vsctl add-br bridge switch1
```

Nous connectons maintenant le contrôleur au pont :

```
levymab@ubuntu1:~$ sudo ovs-vsctl set-controller switch1 tcp:192.168.1.12:6653
```

Ensuite nous configurons les tunnels vers le switch 2 et le switch 3 :

```
levymab@ubuntu1:~$ sudo ovs-vsctl add-port switch1 v0 -- set interface v0 type=vxlan option:remote_ip=192.168.1.71 ofport_request=11
levymab@ubuntu1:~$ sudo ovs-vsctl add-port switch1 v1 -- set interface v1 type=vxlan option:remote_ip=192.168.1.72 ofport_request=10
```

Il ne nous reste plus qu'à assigner une adresse IP au bridge et tester la communication :

```
levymab@ubuntu1:~$ sudo ip addr add 10.10.10.2/24 dev switch1
levymab@ubuntu1:~$ sudo ifconfig switch1 up
levymab@ubuntu1:~$
```

Testons la communication avec le bridge switch 3 par exemple :

```
levymab@ubuntu1:~$ ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_seq=1 ttl=64 time=1.27 ms
64 bytes from 10.10.10.3: icmp_seq=2 ttl=64 time=1.05 ms
^C
--- 10.10.10.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
```

Nous pouvons afficher le récapitulatif de la configuration qui a été faite sur l'OVS1 avec la commande : *sudo ovs-vsctl show*



```
levymab@ubuntu1:~$ sudo ovs-vsctl show
[sudo] password for levymab:
b5e6fb01-24e3-48d9-9d3e-d9a3649eb831
    Bridge switch1
        Controller "tcp:192.168.1.12:6653"
        Port v0
            Interface v0
                type: vxlan
                options: {remote_ip="192.168.1.71"}
        Port switch1
            Interface switch1
                type: internal
        Port v1
            Interface v1
                type: vxlan
                options: {remote_ip="192.168.1.72"}
    ovs_version: "2.13.5"
levymab@ubuntu1:~$
```

- Configuration sur Ubuntu 2 dont l'adresse IP est 192.168.1.71

```
levymab@ubuntu2:~$ sudo apt install openvswitch-switch
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

On crée notre bridge switch2 :

```
levymab@ubuntu2:~$
levymab@ubuntu2:~$ sudo ovs-vsctl add-br switch2
```

Après l'installation d'OpenvSwitch et la création du pont, nous rajoutons notre contrôleur au pont :

```
levymab@ubuntu2:~$ sudo ovs-vsctl set-controller switch2 tcp:192.168.1.12:6653
```

Passons maintenant aux tunnels VXLAN, un tunnel avec le bridge switch1 :

```
levymab@ubuntu2:~$ sudo ovs-vsctl add-port switch2 v0 -- set interface v0 type=vxlan option:remote_ip=192.168.1.70 ofport_request=11
```

Un autre tunnel avec le bridge switch3 :

```
levymab@ubuntu2:~$ sudo ovs-vsctl add-port switch2 v2 -- set interface v2 type=vxlan option:remote_ip=192.168.1.72 ofport_request=11
```

Des adresses doivent être attribuées à notre OVS pour la communication entre les tunnels créés :

```
levymab@ubuntu2:~$ sudo ip addr add 10.10.10.1/24 dev switch2
levymab@ubuntu2:~$ sudo ifconfig switch2 up
levymab@ubuntu2:~$
```



```
levymab@ubuntu2:~$ ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_seq=1 ttl=64 time=4.65 ms
64 bytes from 10.10.10.3: icmp_seq=2 ttl=64 time=1.25 ms
^C
--- 10.10.10.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.246/2.949/4.653/1.703 ms
levymab@ubuntu2:~$
```

On peut voir, après l'assignation d'une adresse ip, que la communication avec le bridge switch 3 est bien établie.

Nous allons voir un récapitulatif de la configuration qui a été faite sur l'OVS2.

Nous obtenons ces informations en tapons la commande : ***sudo ovs-vsctl show***

```
levymab@ubuntu2:~$ sudo ovs-vsctl show
35a4e6ea-3d89-41ac-adff-a9fe6738d3d8
    Bridge switch2
        Controller "tcp:192.168.1.12:6653"
        Port v0
            Interface v0
                type: vxlan
                options: {remote_ip="192.168.1.70"}
        Port v2
            Interface v2
                type: vxlan
                options: {remote_ip="192.168.1.72"}
        Port switch2
            Interface switch2
                type: internal
    ovs_version: "2.13.5"
levymab@ubuntu2:~$
```



BIBLIOGRAPHIE

[1] M. Junior ABOTSI « ETUDE ET MISE EN PLACE D'UNE SOLUTION SDN AVEC APPLICATION AU VXLAN » Mémoire de fin d'études de master Télécommunications et Réseaux. ESTM Février 2021.

[2] Innonce DISU HAFID et Adama THIAM « ETUDE DES CONCEPTS ET APPLICATIONS DES RESEAUX DEFINIS PAR LES LOGIGIELS (SDN) » Mémoire de fin d'études de Licence Télécommunications et Réseaux. EC2LT 2020.

[3] M. Chikhi MEHDI « ÉTUDE ET IMPLEMENTATION DE L'APPROCHE SOFTWARE-DEFINED NETWORK DANS UN RESEAU LOCAL » Mémoire de fin d'études pour l'obtention du Master Systèmes Informatiques et Réseaux 2019.

[4] M. Anis AICHAOUI et Yanis AITBELKACEM « ETUDE ET IMPLEMENTATION D'UNE ARCHITECTURE SDN LAN » Mémoire de fin d'études de master académique Réseaux et Télécommunications 2018.

[5] Joé HABRAKEN et Matt HAYDEN « Les Réseaux » 3iem Edition 2007.

[6] <https://www.adiac-congo.com/content/sante-une-application-pour-une-meilleure-gestion-hospitaliere-128203> 08 Octobre 2021

[7] <https://www.nexcom.fr/formation/sdn-nfv-virtualisation-reseaux-et-cloud/> 16 Octobre 2021.

[8] https://fr.getamap.net/cartes/republic_of_the_congo/kouilou/_mouyondzi/ 29 Octobre 2021.

[9] https://fr.wikipedia.org/wiki/Software-defined_networking 19 Novembre 2021.

[10] <https://www.silicon.fr/hub/colt-hub/5g-et-sdn-vers-le-on-demand-sur-les-mobiles> 25 Novembre 2021.

[11] <https://www.silicon.fr/hub/colt-hub/5g-et-sdn-vers-le-on-demand-sur-les-mobiles> 28 Novembre 2021.

[12] <https://www.vmware.com/fr/topics/glossary/content/software-defined-networking.html> 29 Novembre 2021.



- [13] <https://www.juniper.net/fr/fr/products/sdn-and-orchestration/contrail.html>
29 Novembre 2021
- [14] <https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-1-introduction/> 02 Décembre 2021.
- [15] <https://www.cisco.com/site/us/en/products/networking/dna-center-platform/index.html> 11 Décembre 2021
- [16] https://techhub.hpe.com/eginfolib/networking/docs/sdn/sdnc2_7/5200-0910prog/content/s_sdnc-embedded-apps.html 19 Décembre 2021
- [17] <https://www.usenix.org/conference/nsdi21/presentation/ferguson> 04
Janvier 2022
- [18] <https://ryu-sdn.org/> 07 Janvier 2022
- [19] <https://docs.openvswitch.org/en/latest/intro/why-ovs/> 10 Janvier 2022.
- [20] <https://docs.citrix.com/fr-fr/citrix-adc/current-release/networking/vxlans.html> 16 Janvier 2022.
- [21] <https://securitynetworkinglinux.wordpress.com/2021/01/13/how-to-perform-throughput-testing-using-iperf3-on-ubuntu-20-04-cli/> 23 Janvier 2022
- [22] <http://mininet.org/overview/> 16 Janvier 2022.
- [23] <https://mininet-wifi.github.io/> 18 Février 2022
- [24] <https://www.opendaylight.org/about/platform-overview> 21 Février 2022.
- [25] <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343549/> 18 Mars 2022.
- [26] <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/>
28 Mars 2022.
- [27] <https://developer.cisco.com/codeexchange/github/repo/CiscoDevNet/OpenDaylight-Openflow-App/> 13 Avril 2022
- [28] https://infoloup.no-ip.org/virtualbox-debian11-openvswitch-creation/#1-Construction_de_la_VM_depuis_VirtualBox 23 Juin 2022.

