Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid- Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: Réseaux et Systèmes distribués (R.S.D)

Thème

Etude et mise en oeuvre d'une solution SDN :

Application de Gestion de VLANs

Réalisé par :

- **Mme BOUIDA Hafida née SAIDI**

Présenté le 12 Juin 2017 devant le jury composé de MM:

- *Mr. BENAMMAR Abdelkrim* — *Président*
- *Mme LABRAOUI Nabila* — *Encadreur*
- *Mr. BEKARA Chakib* — *Examinateur*

Année universitaire : 2016-2017

Résumé

Le SDN – Software Defined Networking – est certainement le sujet chaud qui agite le monde du réseau depuis ces dernières années, la technologie SDN a déclenché un changement radical à long terme dans la conception des réseaux, et le marché s'est rapidement approprié le SDN comme ensemble de solutions/architectures permettant de supprimer les frontières existantes entre les mondes des applications et du réseau. Alors que le déploiement d'applications est toujours plus aisé et dynamique, notamment grâce à la virtualisation et au Cloud.

Dans ce mémoire, nous avons évalué les performances de deux contrôleurs SDN, à savoir, Floodlight et Opendaylight. Nous avons également développé une application de gestion des VLANs dans un environnement SDN.

Mots-clefs : SDN, Contrôleur, Floodlight, Openflow, Openvswitch, VLAN.

Abstract

The SDN - Software Defined Networking - is certainly the hot topic that has been stirring the network world for the past few years, SDN technology has triggered a radical change over the long term in network design, and the market quickly appropriated the SDN As a set of solutions / architectures to remove the existing boundaries between the worlds of applications and the network. While deploying applications is easier and more dynamic, especially with virtualization and the Cloud.

In the context of our project, we evaluate the performance of two SDN controllers: Floodlight and Opendaylight. We also developped a VLAN management application in an SDN environment.

Keywords : SDN, Controller, Floodlight, Openflow, Openvswitch, VLAN.

ملخص

الشبكات المبرمجة ثورة جديدة في البنى التحتيه للشبكات، اخر ما توصل اليه العلم في تكنولوجيا الاتصالات والشبكات وهي شبكات ما تعرف ب (اس دي ان) والمتضمنة في بناء و تشغيل شبكة معلومات فائقة السرعة وعالية الموثوقية والاعتمادية والاستقرار في التشغيل، مستخدمين في ذلك حاسباً ، وبرمجيات .

في هذه المذكرة أثبتت نتائج تحاليل قياس أداء وحدات التحكم فلودلايت و اوبن دايلايت ، فعالية وحدة التحكم فلودلايت كما قمنا في هذه المذكرة بتطوير تطبيق إدارة شبكات محلية ظاهرية

الكلمات المفتاحية : الشبكات المبرمجة، اس دي ان ، التطبيقات ، أوبن فلو ، فلودلايت ، الشبكة المحلية الافتراضية

Remerciements

Ma reconnaissance ainsi que ma dévotion se dirigent tout d'abord vers mon Créateur, le Tout Puissant, qui m'a offert la vie et le bonheur qui va avec. Dieu, qui m'a toujours guidé lors de ma route, Dieu qui m'a doté d'une force, d'un courage et d'une patience afin d'affronter mes obstacles et de mener à bien mes travaux, ce modeste mémoire inclus.

Je tiens à exprimer ma gratitude et mes respects les plus sincères à mon encadreur, Mme LABRAOUI Nabila, pour avoir accepté si généreusement de m'encadrer et pour l'aide qu'elle a bien voulu m'accorder tout au long de mon travail, moyennant ses critiques constructives et ses suggestions pertinentes.

Mes remerciements vont aussi aux membres du jury pour l'honneur d'avoir voulu examiner et évaluer ce modeste travail.

Je remercie également mon mari pour ses précieux conseils et son soutien tout au long de mes études, ainsi que ma mère, pour son amour et son engagement inconditionnel. J 'espère qu'elle sera toujours fière de moi .

Si ce document a pu voir le jour, c'est grâce aux actions conjuguées de plusieurs personnes à qui je dis sincèrement merci .

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Dédicaces

Ce mémoire est dédié principalement à mon défunt papa qui ne sera hélas pas présent pour partager mon stress et ma joie le jour de la soutenance, j'espère simplement qu'il est fier de moi !!

A ma maman même si aucune dédicace ne saurait exprimer ma profonde gratitude et ma vive reconnaissance envers tous les sacrifices qu'elle a faits pour moi.

A mon cher mari, et
A mon Superbe trésor, mes adorables anges :
Rachad et Lina,
vous êtes ma plus grande richesse, Je vous aime !

A mon frère et mes sœurs, mon neveu et mes nièces,
Ainsi qu'à toute ma famille : petits et grands

Hafida

TABLE DES MATIERES

Introduction générale.....1

PARTIE I : État de l'art des technologies de réseaux

Chapitre I : État actuel des réseaux

I.1 Introduction.....3
I.2 L'informatique en nuage.....3
I.3 La Virtualisation.....4
I.4 Cloud computing.....4
I.4.1 Caractéristiques du cloud :.....5
I.5 Un besoin de réseau programmable.....5
I.6 Conclusion.....6

Chapitre II : Vers un réseau programmable : SDN

II.1 Introduction.....7
II.2 Définition et Architecture du SDN.....7
II.2.1 Architecture SDN.....7
II.3 Avantages du SDN.....9
II.3.1 Réseaux programmables.....9
II.3.2 Flexibilité.....9
II.3.3 Politique unifiée.....9
II.3.4 Routage.....9
II.3.5 Gestion du Cloud.....9
II.3.6 Simplification matérielle.....10
II.4 Le protocole OpenFlow dans l'architecture SDN.....10
II.4.1 La genèse d’OpenFlow.....10
II.4.2 Structure d’un commutateur OpenFlow.....11
II.4.3 Table de flux.....11
II.4.4 Messages OpenFlow.....13

II.4.4.1 Messages symétriques....13
II.4.4.2 Messages asynchrones....14
II.4.4.3 Messages contrôleur-commutateur....14
II.5 Quelques Contrôleurs SDN....15
II.5.1 NOX....16
II.5.2 POX....16
II.5.3 Beacon....16
II.5.4 Floodlight....16
II.5.5 OpenDaylight....16
II.6 Conclusion....16

PARTIE II : Mise en place d'un réseau SDN

Chapitre III : Contribution 1 : Évaluation des performances de Floodlight et Opendaylight

III.1. Introduction....18
III.2 Émulateur du réseau Mininet....19
III.2.1 Configuration utilisée....19
III.2.2 Fonctionnement de base du Mininet....19
III.2.2.1 Création d'une topologie avec la ligne de commande....19
III.2.2.2 Topologies personnalisées....20
III.3 Open vSwitch....21
III.3.1 Interagir avec Open vSwitch....22
III.4 Contrôleur SDN....22
III.4.1 Floodlight et Opendaylight....22
a. Fonctionnalité de base des 2 contrôleurs 23
b. Comparaison de floodlight et opendaylight 25
III.4.2 Évaluation des performances de Floodlight et Opendaylight....26
a. Tests de latence :....26
b. Tests de débit :....28
c. Discussion 29
III.5 Conclusion....30

Chapitre IV : Contribution 2 : Réalisation d'une application SDN « getion de VLAN »

IV.1 Introduction aux VLANs.....31
IV.2 Intérêt des VLANs.....31
IV.3 Difficulté des vlans dans les réseaux traditionnels.....32
IV.4 Le SDN, pour une gestion simple des VLANs.....32
IV.5 Implémentation d'une application de gestion des VLANs avec SDN.....33
IV.5.1 Mettre en place une topologie à l'aide du Mininet.....35
IV.5.1.1 Création de topologie avec Mininet 35
a.Scenario 1 35
b. Scenario 2 37
IV.5.1.2 Contrôleur Floodlight.....39
IV.5.2 Conception et développement d'une application de gestion de VLAN.....42
IV.5.2.1 Présentation de l'application.....42
IV.5.2.2 Fonctionnement du VLAN.....45
IV.6 Conclusion.....48

Conclusion générale.....49

Bibliographie 51

Annexe 1 : Installation du floodlight.

Annexe 2 : Installation du Mininet .

Annexe 3 : Comparaison entre Floodlight et Opendaylight.

Liste des Figures

Liste des Tableaux

Liste des Figures

Chapitre I : État actuel des réseaux

Figure I.1 : L'informatique en nuage. 4

Chapitre II : Vers un réseau programmable : SDN

Figure II.1 : Openflow dans l'architecture SDN 8

Figure II.2 : Diagramme de flux des messages OpenFlow.....11

Figure II.3 : Table de flux 12

Figure II.4 : Schématisation du modèle de flux OpenFlow.....13

Chapitre III : Contribution 1 : Évaluation des performances de Floodlight et Opendaylight

Figure III.1 : Schéma d'un réseau SDN simple. 18

Figure III.2 : Différents types de topologie réseau 20

Figure III.3 : Interface web du Floodlight. 24

Figure III.4 : Interface web du Opendaylight. 25

Figure III.5 : Résultats de la latence du Floodlight 27

Figure III.6 : Résultats de la latence d'opendaylight 28

Figure III.7 : Résultats de débit du Floodlight. 29

Figure III.8 : Résultats de débit d'opendaylight 29

Chapitre IV : Contribution 2 : Réalisation d'une application SDN « gestion des VLANS »

Figure IV.1 : VLAN 31

Figure IV.2 : Topologie du scenario 1..... 34

Figure IV.3 : Topologie du scenario 2 35

Figure IV.4 : Création du topologie de scenario 1 36

Figure IV.5 : Exécution de la commande help du Mininet..... 36

Figure IV.6 : sortie des commandes net et dump 37

Figure IV.7: Fichier multiswitch-vlan.py..... 38

Figure IV.8: Topologie du scenario 2.....38

Figure IV.9: Interface web du Floodlight.....40

Figure IV.10: Onglet switches de l'Interface web du Floodlight.....41

Figure IV.11 : Onglet Topology de l'Interface web du Floodlight.....41

Figure IV.12 : Ajout d'un VLAN 43

Figure IV.13 : Exemple d'une règle de flux vlan ajoutée à la table des flux d'un switch..... 43

Figure IV.14 : Modification d'un Vlan 44

Figure IV.15 : Visualiser un vlan..... 45
Figure IV.16 : Scenario 1 : ping entre les hôtes (h2,h4 : vlan10 et h3,h5:vlan20)..... 46
Figure IV.17: Scenario 2 : ping entre les hosts (h2,h6 : vlan10 et h3,h5:vlan20)..... 47

Liste des tableaux

Chapitre III : Contribution 1 : Évaluation des performances de Floodlight et Opendaylight

Tableau III.1 : comparaison entre floodlight et opendaylight. 25

Introduction générale

Avec l'avènement des innovations technologiques récentes telles que la virtualisation des systèmes et le cloud computing, les limites actuelles des architectures réseaux deviennent de plus en plus problématiques pour les opérateurs et les administrateurs réseaux. En effet, depuis déjà plusieurs années, il est communément admis que les architectures IP traditionnelles sont, d'une part, particulièrement complexes à configurer à cause de la nature distribuée des protocoles réseaux et, d'autre part, difficile à faire évoluer en raison du fort couplage qui existe entre le plan de contrôle et le plan de données des équipements d'interconnexion existants.

Le problème est en effet que le manque d'innovation dans l'univers des réseaux a fait peser des contraintes importantes sur le déploiement des applications réseaux. Les constructeurs ont donc fini par se décider à tenter d'apporter aux réseaux le niveau de flexibilité et de simplicité dont ont besoin les entreprises et les fournisseurs de services.

Les principaux domaines d'innovation au cours des dernières années sont à chercher du côté du contrôle centralisé, de la programmabilité, et de la virtualisation. Ces innovations sont regroupées sous l'appellation commune de SDN (ou software-defined network) et visent à résoudre les problèmes que les entreprises rencontrent avec leurs réseaux. Le paradigme SDN est une nouvelle approche qui a pour ambition de répondre à cette rigidité architecturale des réseaux IP actuels, notamment en les rendant plus programmables.

L'objectif de ce travail de fin d'études, est de présenter et identifier ce nouveau paradigme incontournable aux réseaux du futur. C'est notamment le réseau programmable (plus connus sous le nom de SDN), qui permet une ouverture à de futures applications réseau commerciales, prêtes à l'emploi. Il devient ainsi possible de concevoir des réseaux beaucoup plus intelligents et flexibles, et comme nous avons aujourd'hui le choix entre les environnements Windows, linux et Mac, nous aurons à notre disposition différentes plateformes pour les applications réseau.

Dans ce mémoire, nous mesurons les performances des contrôleurs Floodlight et Opendaylight, les résultats d'analyse ont prouvé l'efficacité du contrôleur Floodlight.

Nous avons égalemet développé une application de gestion des VLANs dans un environnement SDN.

Ainsi, ce manuscrit est organisé en quatres chapitres suivis d'une conclusion générale, à savoir :

Dans le premier chapitre : **État actuel des réseaux**, nous essayons de donner une vue générale sur les technologies des réseaux.

Dans le second chapitre : **vers un réseau programmable : SDN**, nous essayons d'aborder la technologie du SDN en détail.

Dans le troisième chapitre : **Contribution 1 : Évaluation des performances de Floodlight et Opendaylight**, nous expliquons principalement les différentes démarches suivies pour déployer un réseau SDN, ensuite nous expliquons la raison pour laquelle on a choisi le contrôleur Floodlight en le comparant avec un autre contrôleur (Opendaylight).

Dans le quatrième chapitre : **Contribution 2 : Réalisation d'une application de getion de VLAN**, nous avons présenté la couche application du SDN, et nous avons choisi le domaine des VLANs afin de démontrer et de prouver l'utilité et l'objectif des réseaux SDN.

Nous terminons ce mémoire par une Conclusion générale dans laquelle nous présentons notre contribution, ainsi que des perspectives envisagées pour ce travail.

PARTIE I :

État de l'art des technologies de réseaux

Dans le monde il y a 10 types de personnes,

ceux qui comprennent le binaire, et les autres.

Chapitre I

État actuel des réseaux

SOMMAIRE

1. Introduction
2. L'informatique en nuage
3. Importance de la virtualisation
4. Cloud Computing
5. Un besoin de réseau programmable
6. Conclusion

I.1 Introduction

Les réseaux à grande échelle sont coûteux à exploiter et les fournisseurs de services sont toujours à la recherche des moyens pour réduire leur capital et les coûts opérationnels, ceci s'explique par l'explosion des différentes technologies, tel que la virtualisation des serveurs, du stockage, ou encore des postes de travail et l'apparition de l'informatique en nuage, mais avec toutes ces innovations, nous rencontrons toujours les problèmes de l'incapacité à faire évoluer les réseaux physiques aussi vite que les montées en charge, ou encore une limitation dans le déploiement automatique des ressources dans des environnements multi-clients.

Dans ce chapitre, nous présentons les technologies des réseaux qui sont utilisées actuellement tel que l'informatique en nuage, la virtualisation, le cloud, et le besoins d'avoir un réseau programmable.

I.2 L'informatique en nuage

Un nuage contient un réseau qui relie un grand nombre d'ordinateurs et de ressources informatiques , mais peu importe ce qu'il y a dedans, car l'utilisateur n'a pas à le savoir , l'essentiel est d'obtenir le service demandé [1].

Selon la définition la plus communément admise, l'informatique en nuage est un modèle pratique, permettant d'établir un accès par le réseau à un réservoir partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement mobilisées et mises à disposition [1].

Les entreprises peuvent utiliser les technologies de l'informatique en nuage pour fournir l'information, ce qui optimise l'utilisation des équipements, augmente la flexibilité et la rapidité et réduit les dépenses en capital. Les technologies de l'informatique en nuage facilitent l'accès des particuliers aux informations et aux contenus en ligne et offrent une plus grande intéractivité [1].

L'utilisation de l'informatique en nuage diffère en fonction des utilisateurs. Les particuliers utilisent en général cette technologie pour les services de messagerie électronique, de stockage de fichiers, de partage d'informations, de paiement et de diffusion des flux de musique et de vidéo [1]. Les entreprises l'utilisent principalement pour les activités de bureautique de base, la collaboration, la gestion de projet et la création d'applications

personnalisées [1].

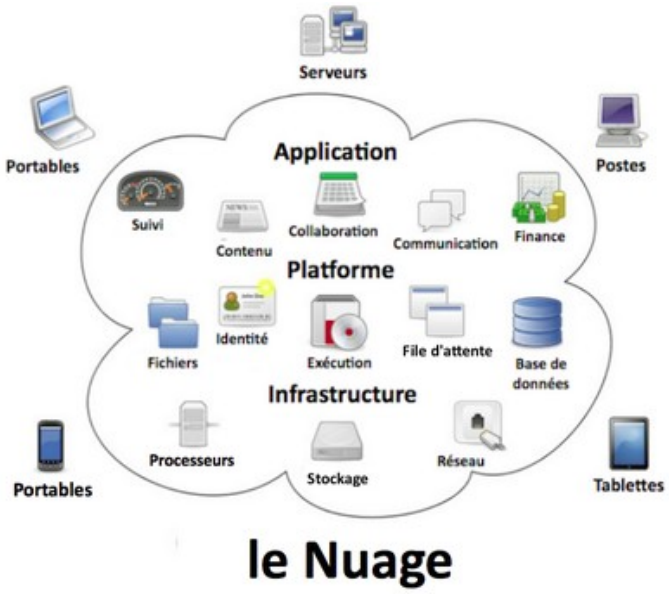


Figure I.1 : L'informatique en nuage [16]

I.3 La Virtualisation

La virtualisation consiste à faire fonctionner un ou plusieurs systèmes d'exploitation comme un simple logiciel sur un ordinateur, au lieu de ne pouvoir en installer qu'un seul par machine [17], elle est utilisée donc pour permettre le fonctionnement de plusieurs machines disposant chacune de leur système d'exploitation spécifique partageant la même infrastructure physique [17].

Le concept couvre différents aspects, on peut ainsi virtualiser les serveurs , le stockage , les applications (simplification de l'administration) ou encore le poste client [17].

A ce stade, la virtualisation et l'informatique en nuage, ces deux sujets vont de pair, les avantages de la virtualisation prennent tous leur sens à l'échelle de l'informatique en nuage.

I.4 Cloud computing

Le Cloud représente un basculement de tendance, au lieu d'obtenir des fonctionnalités et de la puissance de calcul par l'acquisition de matériel et de logiciel, on préfère utiliser de la puissance de calcul et du stockage mise à disposition par un fournisseur via son navigateur web [18].

Le Cloud Computing nous offre aujourd'hui des services sophistiqués (*logiciels en ligne,*

sauvegarde de données…) exploitables à volonté depuis n'importe où dans le monde à partir du moment où l'on dispose d'une connexion Internet [18].

I.4.1 Caractéristiques du cloud :

- **Self-service à la demande :** un utilisateur peut allouer des ressources sans interaction humaine avec le fournisseur [1].
- **Accès réseau élargi :** les ressources sont accessibles via le réseau par des systèmes hétérogènes (clients légers, clients lourds, etc.)[1].
- **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 donnée) des ressources [1].
- **Élasticité:** les ressources peuvent être augmentées ou diminuées facilement, voire 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 [1].
- **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 [1].

I.5 Un besoin de réseau programmable

Au début de l'ingénierie des réseaux, il y avait juste un simple Ethernet ou des règles IP à gérer et le réseau était statique, mais aujourd'hui, les exigences du réseau deviennent de plus en plus complexes, les réseaux sont devenus de plus en plus dynamiques et de nouveaux concepts pour dissocier le contrôle et les données sont développés pour répondre à ces limitations [2].

En effet, les réseaux doivent être réactifs et élastiques car de nombreux ordinateurs sont connectés, déplacés ou déconnectés tel que les réseaux sans fil et service à la demande du Cloud computing [2].

En même temps, il est nécessaire de disposer de périphériques réseau (commutateurs, routeurs) de plus haut niveau et ces périphériques doivent être configurés séparément et proviennent souvent de fournisseurs différents (et donc d'utiliser des commandes propres au

constructeur) [2], et les ingénieurs de réseau doivent faire face à de plus en plus de complexité dans le routage et la gestion des flux, car les flux circulants changent et évoluent rapidement.

Par conséquent, il est difficile de gérer les réseaux et d'appliquer efficacement de nouvelles politiques, car chaque périphérique fonctionne sur un protocole d'un niveau donné et doit être configuré de manière appropriée les uns avec les autres. Donc, il n'est pas facile d'ajouter une règle parce qu'il n'y a pas de moyen standardisé de le faire, mais aussi parce qu'il peut entrer en conflit avec l'un des nombreux appareils sur le chemin. En outre, les règles étaient initialement destinées à être statiques et donc le réseau est statique et mal adapté aux besoins et technologies actuels [2].

Alors, cette série de questions et de difficultés nous montre qu'il faut trouver une solution d'automatisation adaptée aux défis de réseau, c'est le réseau programmable.

I.6 Conclusion

Est-ce que l'informatique en nuage, la virtualisation et le cloud computing résolvent toutes les problématiques? Non, car il reste la problématique d'appliquer des règles réseau, d’où la sécurité n’est qu'un aspect parmi d’autres, notamment des règles de QoS, de routage et de load-balancing. C’est pourquoi le marché planche sur SDN.

Alors dans le chapitre suivant, nous discuterons en premier lieu sur la technologie SDN, puis nous traiteront les différents composants d'un réseau SDN.

Chapitre II

Vers un réseau programmable : SDN

SOMMAIRE

1. Introduction

2. Définition et Architecture du SDN

3. Avantages du SDN

4. Le Protocole OpenFlow dans l'Architecture SDN

5. Quelques contrôleur SDN

6. Conclusion

II.1 Introduction

L'idée de SDN est de rendre le réseau programmable, c'est-à-dire de permettre aux applications d'intéragir directement avec les réseaux, avec une coopération à double sens : l'application informe le réseau du comportement désiré, le réseau informe les applications de ses capacités. Le tout permettant d'établir un service avec des conditions définies a priori [20].

Dans une telle agitation, les administrateurs réseau sont le plus souvent perdus. Qu'est-ce que le SDN ? Qu'est-ce qu'OpenFlow ? Qu'est-ce que cela apporte pour nous ?

Dans ce chapitre, nous discuterons en premier lieu sur la technologie SDN et ces avantages, puis nous traiterons les différents composants d'un réseau SDN.

II.2 Définition et Architecture du SDN

SDN signifie littéralement Software Defined Networking, c'est-à-dire le réseau défini par logiciel. On comprend donc immédiatement que le sujet est vaste et qu'il va être difficile d'avoir une définition unique [3].

La définition académique, consistait à voir le SDN comme une architecture qui découplait les fonctions de contrôle et de transfert des données du réseau afin d'avoir une infrastructure physique complètement exempte de tout service réseau [3].

Dans ce modèle, les équipements réseau se contentent d'implémenter des règles, injectées par les applications, de traitement des flux de données. Plus besoin d'avoir sur ces équipements de protocoles de routage : une entité intelligente, appelée « **contrôleur** » voit le réseau dans sa globalité et injecte directement les règles de traitement des données sur chaque équipement [3].

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

- Permettre aux applications de programmer le réseau afin d'en accélérer le déploiement ;
- Permettre au réseau de mieux identifier les applications transportées pour mieux les gérer (qualité de service, sécurité, ingénierie de trafic...).

II.2.1 Architecture SDN

Le SDN présente une architecture réseau où le plan de contrôle est totalement découplé

du plan de données, cela est illustré par la figure II.1.

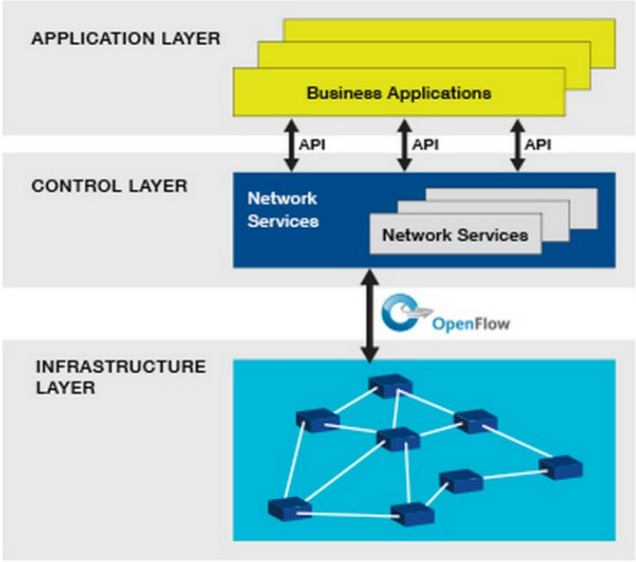


Figure II.1 :Openflow dans l'architecture SDN [3]

La figure II.1 présente l'architecture SDN et la place du protocole Openflow, alors nous observons que SDN se compose de 3 couches :

- **La couche infrastructure** contient les éléments de réseau responsable de l'acheminement du trafic et qui supportent le protocole OpenFlow qu'ils partagent avec le contrôleur [3].
- **La couche de contrôle** est logicielle, basée sur le contrôleur SDN qui offre une visibilité globale du réseau et des équipements d'infrastructure [3].
- **La couche applicative** apporte l'automatisation des applications à travers du réseau, et à l'aide des interfaces programmables [3].

Ainsi, dans une approche SDN, la charge de calcul associée au contrôleur est en grande partie retirée des routeurs. Le contrôleur, est donc chargé du calcul et de la mise à jour des tables de routage [4].

II.3 Avantages du SDN

II.3.1 Réseaux programmables

Avec SDN, il est plus simple de modifier les stratégies réseau car il suffit de changer une politique de haut niveau et non de multiples règles dans divers équipements de réseau [2].

De plus, la centralisation de la logique dans un tel contrôleur entièrement personnalisé avec des connaissances globales et une puissance de calcul élevée, simplifient le développement des fonctions plus sophistiquées. Cette aptitude à programmer le réseau est l'élément clé du SDN [2].

II.3.2 Flexibilité

SDN apporte également une grande flexibilité dans la gestion du réseau. Il devient facile de rediriger le trafic, d'inspecter des flux particuliers, de tester de nouvelles stratégies ou de découvrir des flux inattendus [2].

II.3.3 Politique unifiée

Avec son contrôleur, SDN garantit également une politique réseau unifiée et à jour, puisque le contrôleur est responsable de l'ajout de règles dans les commutateurs, il n'y a aucun risque qu'un administrateur de réseau ait oublié un commutateur ou installé des règles incohérentes entre les dispositifs.

En effet, l'administrateur va simplement spécifier une nouvelle règle et le contrôleur adaptera la configuration pour envoyer des règles cohérentes dans chaque dispositif pertinent [2].

II.3.4 Routage

SDN peut également être utilisé pour gérer les informations de routage de manière centralisée en déléguant le routage et en utilisant une interface pour le contrôleur [2].

II.3.5 Gestion du Cloud

SDN permet également une gestion simple d'une plateforme cloud. En effet, la dynamique apportée par SDN traite des problèmes spécifiques aux clouds tels que l'évolutivité, l'adaptation, ou des mouvements de machines virtuelles [2].

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

II.3.6 Simplification matérielle

SDN a tendance d'utiliser des technologies standard et de base pour contrôler les équipements du réseau, tandis que la puissance de calcul n'est requise qu'au niveau du contrôleur. Ainsi, les équipements de réseau deviendront des produits à bas prix offrant des interfaces standard [2].

Avec ce type de matériel, il serait également simple d'ajouter de nouveaux périphériques, puisqu'ils ne sont pas spécialisés, de les connecter au réseau et de laisser le contrôleur les gérer conformément à la politique définie. Ainsi, le réseau devient facilement évolutif dès que le contrôleur est évolutif [2].

II.4 Le protocole OpenFlow dans l'architecture SDN

OpenFlow est un protocole de lien entre le plan de contrôle et le plan de données. L'échange de messages se fait au cours d'une session **TCP** établie via le port **6633** du serveur contrôleur .

Openflow est donc une composante du SDN, son développement a commencé en 2007 dans le cadre d'une collaboration entre les mondes de l'université et des affaires. Etablie à l'origine par l'université de Stanford et l'université de Californie à Berkeley,

Un switch OpenFlow est déterminé par une ou plusieurs tables de flux (flow table). Chaque table s'assimile à un ensemble de flux [4]. Lorsqu'un paquet parvient au switch, les valeurs contenues dans ses en-têtes sont comparées aux différentes règles enregistrées dans la table de flux du switch [4].

II.4.1 La genèse d'OpenFlow

L'histoire d'OpenFlow est intéressante et permet de mieux comprendre son rôle fondamental dans la conception de l'architecture SDN [6].

OpenFlow a été initié comme un projet à l'université de Stanford lorsqu'un groupe de chercheurs exploraient la manière de tester de nouveaux protocoles dans le monde IP (créer un réseau expérimental confondu avec le réseau de production) mais sans arrêter le trafic du réseau de production lors des tests [6].

C'est dans cet environnement que les chercheurs à Stanford ont trouvé un moyen de séparer le trafic de recherche du trafic du réseau de production qui utilisent le même réseau IP.

OpenFlow est donc un protocole ouvert (open protocol) qui permet aux administrateurs de réseau de programmer les tables de flux (flow tables) dans leurs différents

commutateurs, chacun avec son ensemble de fonctionnalités et caractéristiques de flux [6].

II.4.2 Structure d'un commutateur OpenFlow

Les commutateurs openflow contiennent des tables de flux qui sont utilisées pour effectuer des fonctions de transfert indiquées dans les en-têtes de paquets [6].

À l'aide de la table de flux, une des actions suivantes est exécutée :

A. Relayer le paquet sur un port de sortie,

B. Supprimer le paquet,

C. Passer le paquet au contrôleur. Le paquet est encapsulé dans un message OpenFlow **PACKET_IN**.

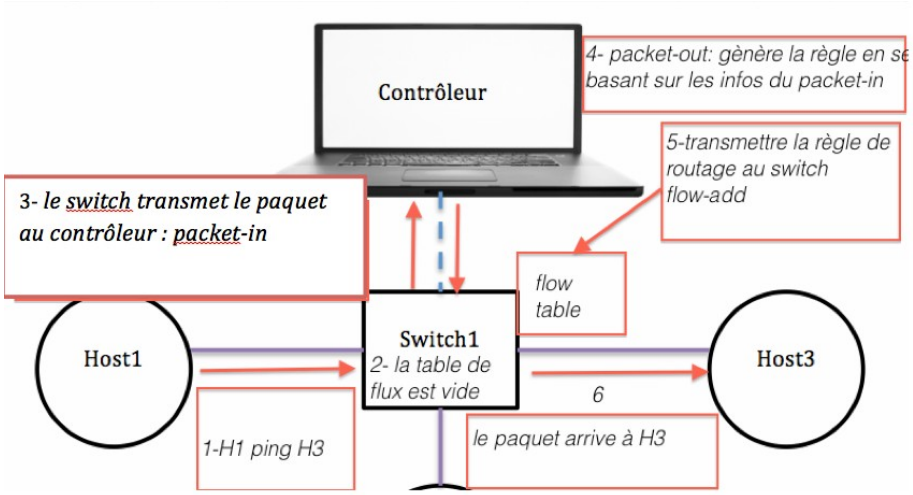


Figure II.2 : Diagramme de flux des messages OpenFlow [8].

II.4.3 Table de flux

Une table de flux "Flow Table" est composée de plusieurs entrées de flux , chacune est structurée comme suit : [10]

Champ de Correspondance	Instructions	Compteurs

- **Champ de correspondance (Match fields)** : Utilisé lors de la recherche de l'entrée correspondante au paquet. Ils sont constitués essentiellement des entêtes des paquets et des ports d'entrées [10].

- **Compteurs (Counters)** : Servent essentiellement à garder des statistiques sur les flux pour ensuite décider si une entrée de flux est active ou non . Pour chaque table, chaque flux , chaque port, des compteurs de statistiques sont maintenus [10]
- **Instructions** : Représentent l'ensemble des instructions OpenFlow qui servent à modifier le traitement que va subir le paquet. Les instructions supportées sont : [10]

  1. Apply-Actions : Pour appliquer les actions sur le paquet immédiatement.
  2. Clear-Actions : Pour supprimer une liste des actions du paquet.
  3. Write-Actions : Ajouter une liste d'actions au paquet.
  4. Write-Metadata : Ajouter des données utiles pour le séquencement entre les tables OpenFlow.
  5. Goto-Table : Indique que le paquet doit être acheminé vers une table d'indice supérieur.

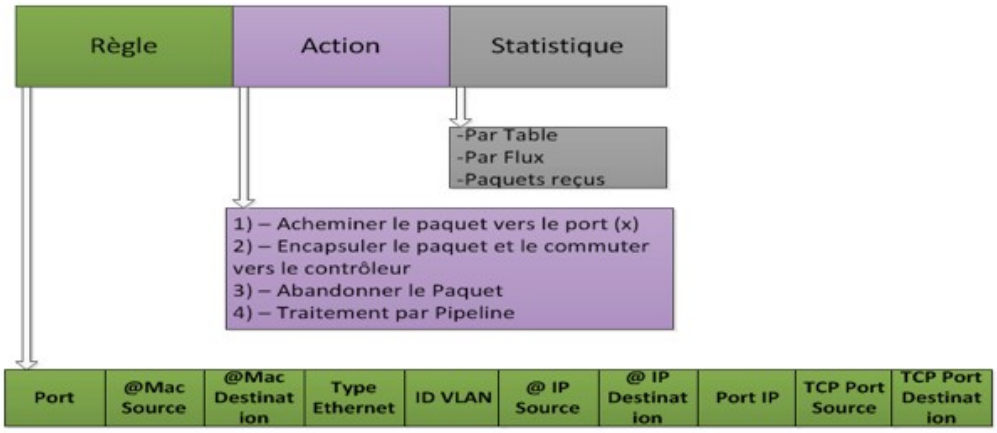


Figure II.3 : table de flux [10]

Les instructions d'une entrée de flux peuvent explicitement diriger le paquet vers une autre table de flux "Flow Table" via l'instruction "Goto". Si le paquet ne correspond à aucune entrée de flux, le comportement du switch vis-à-vis ce paquet dépendra de la configuration de la table. Le comportement par défaut est d'envoyer le paquet au contrôleur avec un message **PACKET_IN**. La table peut aussi spécifier que dans ce cas, le paquet doit continuer son chemin vers la table suivante [10]. Le contrôleur répondra généralement par un **PACKET_OUT** donnant l'instruction à suivre.

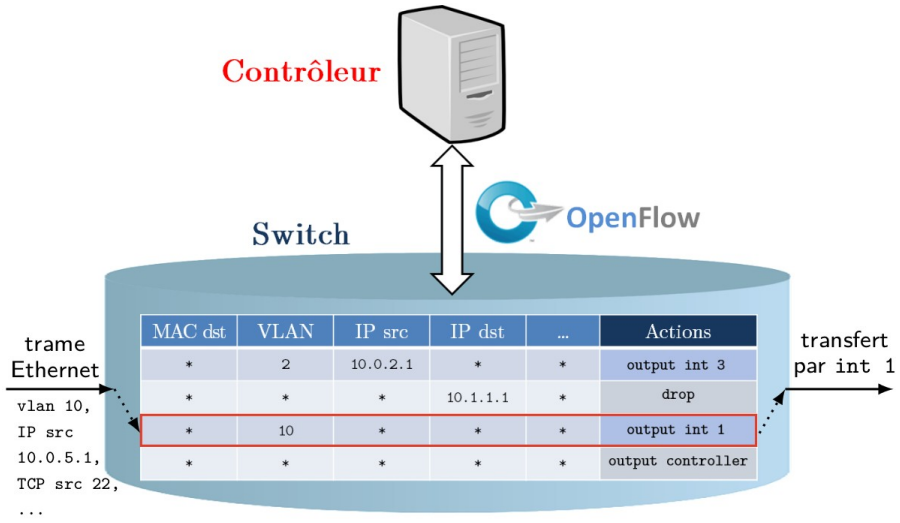


Figure II. 4 : Schématisation du modèle de flux OpenFlow [5]

La figure ci-dessus illustre le traitement d'un paquet parvenant à un switch OpenFlow contenant une seule table de flux : parmi les différents champs d'en-têtes de la trame, l'appartenance au VLAN 10 fait correspondre cette trame à un des flux du switch, dont l'action associée consiste en un transfert par l'interface 1 [5].

II.4.4 Messages OpenFlow

Les messages OpenFlow entre le contrôleur et le commutateur sont transmis via un canal sécurisé, implanté via une connexion TLS sur TCP. Le commutateur initie la connexion TLS lorsqu'il connaît l'adresse IP du contrôleur. Il existe trois catégories de message : symétrique, contrôleur-commutateur et asynchrone [6].

II.4.4.1 Messages symétriques

Les messages symétriques peuvent être émis indifféremment par le contrôleur ou le commutateur sans avoir été sollicité par l'autre entité. Par exemple, on trouve les messages **HELLO** qui sont échangés une fois que la canal sécurisé a été établi [6].

Les messages **ECHO** sont utilisés par n'importe quel entité (contrôleur, commutateur) pendant le fonctionnement du canal pour s'assurer que la connexion est toujours en vie et afin de mesurer la latence et le débit courants de la connexion. Chaque message

ECHO_REQUEST doit être acquitté par un message **ECHO_REPLY** [6].

II.4.4.2 Messages asynchrones

Les messages asynchrones sont émis par le commutateur au contrôleur sans que le commutateur ait été sollicité par le contrôleur. Par exemple, on peut citer le message **PACKET_IN** qui est utilisé par le commutateur pour passer les paquets de données au contrôleur pour leur prise en charge (lorsque aucune entrée de flux ne correspond au paquet entrant ou lorsque l'action de l'entrée correspondante spécifie que le paquet doit être relayé au contrôleur) [6]. Si le commutateur dispose d'une mémoire suffisante pour mémoriser les paquets qui sont envoyés au contrôleur, les messages **PACKET-IN** contiennent une partie de l'en-tête (par défaut 128 octets) , les commutateurs qui ne supportent pas de mémorisation interne (ou ne disposant plus de mémoire) émettent le paquet entier au contrôleur dans le message **PACKET-IN** [6].

Le commutateur peut informer le contrôleur qu'une entrée de flux a été supprimée de la table de flux via le message **FLOW_REMOVED**. Cela survient lorsque aucun paquet entrant n'a de correspondance avec cette entrée pendant un temporisateur spécifié par le contrôleur lors de la création de cette entrée au niveau de la table de flux du commutateur [6].

Le message **PORT_STATUS** est utilisé afin de communiquer un changement d'état du port (le lien est hors service). Finalement le commutateur utilise le message **ERROR** pour notifier des erreurs au contrôleur [6].

II.4.4.3 Messages contrôleur-commutateur

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 [6]*.

a. Les messages ***switch configuration*** 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 [6].
- Le contrôleur utilise la paire de message **FEATURES_REQUEST** et **FEATURES_REPLY** afin d'interroger le commutateur au sujet des fonctionnalités qu'il supporte.

- La paire de message **GET_CONFIG_REQUEST** et **GET_CONFIG_REPLY** est utilisée afin d'obtenir la configuration du commutateur [6].

b. Les messages ***command from controller*** 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 [6].
- Le contrôleur modifie les entrées de flux existantes dans le commutateur via le message **FLOW_MOD**.
- **PORT_MOD** est utilisé pour modifier l'état d 'un port OpenFlow.

c. Des statistiques sont obtenues du commutateur par le contrôleur via la paire de message **STATS_REQUEST et STATS_REPLY.**

d. 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 commutateur via **QUEUE_GET_CONFIG_REQUEST** acquitté par **QUEUE_GET_CONFIG_REPLY** pour apprendre quelle est la configuration des files d'attente associées à un port afin de pourvoir acheminer des paquets sur des files d'attente spécifiques et ainsi fournir à ces paquets un niveau de QoS désiré [6].

e. Le message **BARRIER_REQUEST** 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 commutateur. La confirmation est retournée par le commutateur via la réponse **BARRRIER_REPLY** [6].

II.5 Quelques Contrôleurs SDN

Le contrôleur SDN permet d'implémenter un changement sur le réseau en traduisant une demande globale en une suite d'opérations sur les équipements réseau (ajouts d'états Openflow, configuration en CLI…), les ordres sont donnés au contrôleur par une application via une API dite **« Northbound »** ou nord. Le contrôleur communique avec les équipements via une ou plusieurs API dites **« Southbound »** ou sud. Openflow se positionne comme une API sud agissant directement sur le plan de données [3], il existe plusieurs contrôleurs SDN, tel que :

II.5.1 NOX

Initialement développé chez Nicira, NOX est le premier contrôleur OpenFlow. C'est Open-source et écrit en C ++ . Il est actuellement à la baisse: il n'y a pas eu de changements majeurs depuis mi 2012 [2].

II.5.2 POX

POX est le plus jeune frère de NOX . C'est un contrôleur open-source écrit en Python, et comme NOX, fournit un cadre pour le développement et le test d'un contrôleur OpenFlow, mais les performances POX sont nettement inférieures à celles des autres contrôleurs et ne convient donc pas au déploiement d'entreprise [2].

II.5.3 Beacon

Beacon est un contrôleur Java connu par sa stabilité. Il a été créé en 2010 et est toujours maintenu, il a été utilisé dans plusieurs projets de recherche. En raison de ses performances, c'est une solution fiable pour l'utilisation dans des conditions réelles. Ce contrôleur a également été utilisé dans d'autres projets tels que Floodlight ou OpenDaylight [2].

II.5.4 Floodlight

Floodlight est un contrôleur open-source OpenFlow basé sur Java, pris en charge par BigSwitch Networks. Il est sous licence Apache [8]. Il est facile à configurer et à montrer aussi de grandes performances. Avec toutes ses fonctionnalités, Floodlight est plus une solution complète [2].

II.5.5 OpenDaylight

OpenDaylight est un projet de la Fondation Linux pris en charge par l'industrie. C'est Un framework open source pour faciliter l'accès au logiciel de définition de réseau (SDN) . Comme Floodlight, il peut également être Considéré comme une solution complète [2].

II.6 Conclusion

Au cours de cette première partie, nous avons fourni une base théorique sur les réseaux définis par logiciels (SDN), en présentant l'architecture et les avantages de ce dernier, puis nous avons traité les composants essentiels de cette solution afin d’appliquer ses concepts à notre contexte.

La deuxième partie de ce mémoire sera réservée à notre propre travail qui sera scindé en deux chapitres contenant chacun une contribution.

PARTIE II :

Mise en place d'un réseau SDN

Nous ne pouvons pas prédire où nous conduira
la Révolution Informatique.

Tout ce que nous savons avec certitude,
c'est que, quand on y sera enfin,
on n'aura pas assez de RAM.

- Dave Barry

Chapitre III

Contribution 1 :

Évaluation des performances de Floodlight et Opendaylight

SOMMAIRE

1. Introduction

2. Outil d'émulation du réseau : Mininet

3. Switch openflow

4. Contrôleur SDN

5. Conclusion

III.1. Introduction

Essentiellement le SDN, ou réseau programmable, sépare la partie contrôle de la partie infrastructure du réseau qui traite et achemine les données.

Autrement dit, l’application indique au contrôleur SDN ses prérequis, et le contrôleur relaie la demande à une infrastructure qui n’a plus besoin d’être homogène, ainsi, c’est le contrôleur qui devient le **véritable cerveau** du réseau en apportant les bénéfices suivants :[5]

√ Transformer le réseau en une ressource qui peut être facilement programmée.

√ Garantir la qualité de service, indépendamment de l’infrastructure en place.

√ Gérer aisément et à faible coût le réseau, puisque les administrateurs du réseau n’ont à gérer que les paramètres et les règles, uniquement au niveau du contrôleur central, et non plus au niveau de chaque équipement [5].

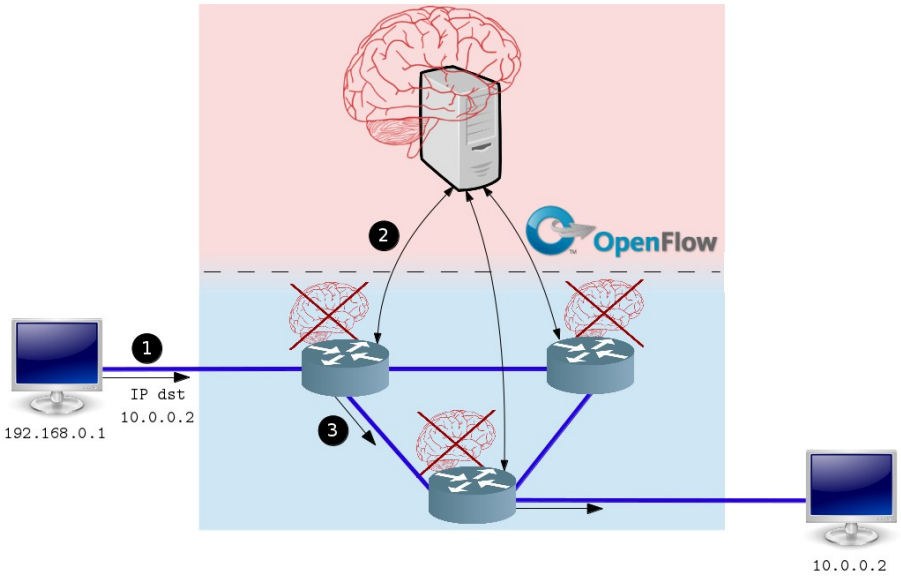


Figure III.1 : Schéma d’un réseau SDN simple [5]

La figure III.1 illustre le routage d’un paquet entre deux réseaux distants via une implémentation de SDN avec le protocole OpenFlow. Les routeurs discriminent les paquets et déterminent leur interface de sortie, mais ce comportement découle de règles émises par le contrôleur. Le routeur qui réceptionne le paquet de l’envoi (1) signale l’évènement au contrôleur et reçoit en retour des règles au cours d’un échange (2), afin de décider du transfert (3) vers le routeur suivant approprié [5].

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Ce chapitre représente la première contribution de notre travail, nous commençons d'abord par une présentation de Mininet et open vswitch, puis nous effectuons une comparaison entre les contrôleurs Floodlight et Opendaylight.

III.2 Émulateur du réseau Mininet

Mininet est un émulateur de réseau, il permet de créer une topologie réseau qui se compose d'un ensemble de hosts, de switchs, de contrôleurs et liens virtuels [8]. Il fournit la capacité de créer des hôtes, les commutateurs et contrôleurs via :

- Ligne de commande ,
- Interface interactive,
- Script Python.

III.2.1 Configuration utilisée

Afin de bien visualiser le fonctionnement du SDN, nous avons choisi d'utiliser une machine virtuelle téléchargée depuis :

https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip

et accessible via VirtualBox, elle utilise le système d'exploitation Linux (Ubuntu-14.04.4).

III.2.2 Fonctionnement de base du Mininet

Mininet permet avec un simple jeu de commande de réaliser des réseaux virtuels en utilisant la commande **mn** [8].

III.2.2.1 Création d'une topologie avec la ligne de commande

Mininet fournit, en plus de la topologie «minimal », la topologie «single», «linear» et «tree» comme montré dans la figure III.2. Pour charger l’une de ces topologies, on utilise l’option **«--topo»** [8], Par exemple: `$ sudo mn --topo single`

«single» tout court donne la même topologie que minimal, c’est-à-dire créer un hôte relié avec un switch, mais on peut ajouter également comme argument un chiffre, qui indique le nombre d'hôtes à créer [8].

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

Si nous souhaitons personnaliser une topologie déjà créée, nous appliquons une des commandes citées ci-dessous :

- Ajouter un switch : self.addSwitch(‘s1’)
- Ajouter un hôte : self.addHost(‘h1’)
- Ajouter un lien : self.addLink(h1, s1)

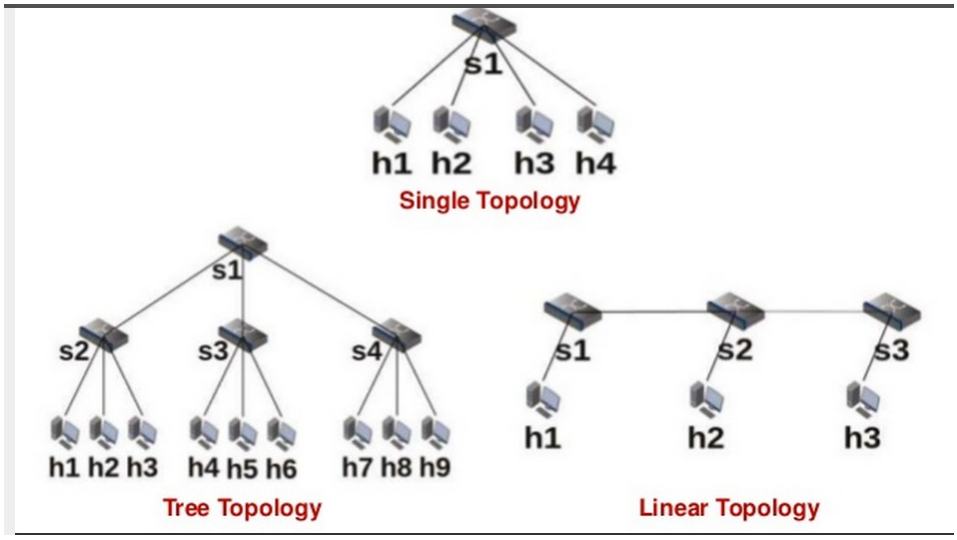


Figure III.2 : Différents types de topologie réseau [8]

Après la configuration, nous devons tester si les paquets sont routés correctement avec la commande **ping** qui est un bon moyen pour vérifier la connectivité :

Mininet> h1 ping h2 **ou bien** Mininet> pingall

et pour analyser les règles insérées dans chaque commutateur, nous utilisons la commande dpctl: Mininet> dpctl dump-flows , et d'une façon générale, pour accéder à toutes les commandes utilisées sous Mininet, il suffit de taper : Mininet> help

III.2.2.2 Topologies personnalisées

Si nous souhaitons travailler avec une topologie autre que tree, linear ou simple, et même sans la ligne de commande, il faut créer une topologie personnalisée à l’aide des scripts Python. Par exemple, si on veut créer 2 hôtes h1 et h2 , les deux sont connectés à un switch s1.

Voici le script Python à écrire :

```
class Test_Topo(Topo):
        def __init__(self):
                "Create P2P topology"
                # Initialize topology
                Topo.__init__(self)
                # Add hosts and switches
                h1 = self.addHost('h1')
                h2 = self.addHost('h2')
                s1 = self.addSwitch('s1')
                # Add links
                self.addLink(h1,s1)
                self.addLink(h2,s1)
topos = {     'toptest': (lambda: Test_Topo())    }
```

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

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

III.3 Open vSwitch

Open vSwitch est bien adapté pour fonctionner comme un commutateur virtuel dans les environnements Virtuels. En plus,il supporte plusieurs technologies de virtualisation basées sur Linux, y compris Xen, KVM et VirtualBox [20].

Open vSwitch est donc une implémentation logicielle d’un switch ethernet. Concrètement il est constitué d’un service (**ovs-vswitchd**) et d’un module kernel (**openvswitch_mod**). Le service permet de commuter effectivement les paquets vers les bons ports virtuels, alors que le module kernel permet de capturer le trafic provenant des interfaces réseau [21].

Pour fonctionner comme n’importe quel switch, Open vSwitch utilise la notion de ports. Chaque port est constitué d’une ou plusieurs interfaces, qui correspondent à des interfaces du système hôte (logiques ou physiques).

III.3.1 Interagir avec Open vSwitch

- Pour connaître l’état global d’un switch : **ovs-vsctl show**

```
root@isfc-linux-02:~# ovs-vsctl show
b3709076-9b43-4f67-90a3-ac88d1f71d01
    ovs_version: "1.4.0+build0"
root@isfc-linux-02:~#
```

- La création d’un switch se fait par la commande :

  ovs-vsctl add-br <nom du virtuel switch>

```
root@isfc-linux-02:~# ovs-vsctl add-br vswitch-01
root@isfc-linux-02:~#
root@isfc-linux-02:~# ovs-vsctl show
b3709076-9b43-4f67-90a3-ac88d1f71d01
    Bridge "vswitch-01"
        Port "vswitch-01"
            Interface "vswitch-01"
                type: internal
    ovs_version: "1.4.0+build0"
root@isfc-linux-02:~#
```

- Pour ajouter un flux directement dans la table de flux sans avoir utiliser un contrôleur, nous utilisons la commande : **dpctl**

  mininet> dpctl add-flow in_port=1,actions=output:2

  mininet> dpctl add-flow in_port=2,actions=output:1

  Cette commande permet de transférer les paquets arrivant à port1 vers port 2 et vice versa.

- Pour vérifier le contenu de la table de flux : **$ dpctl dump-flows**

III.4 Contrôleur SDN

Les deux principaux piliers de l'approche SDN sont la programmabilité des équipements réseaux et la mise en œuvre d'un point de contrôle centralisé [9].

Dans les SDN, le plan de contrôle est placé dans un contrôleur centralisé qui a une visibilité sur l’ensemble du réseau, y compris les hôtes qui s’y connectent, les applications qui disent aux contrôleurs comment programmer le réseau utilisent des interfaces de programmation « **NorthBound** » tandis que les API et protocoles utilisés par le contrôleur pour communiquer avec les équipements du réseau sont dits « **SouthBound** » [9].

III.4.1 Floodlight et Opendaylight

Nous avons choisi d'étudier et de comparer deux contrôleurs : Floodlight et Opendaylight afin d'évaluer leurs paramètres de performance tel que le débit et la latence.

a. Fonctionnalité de base des 2 contrôleurs :

Floodlight est un contrôleur de réseau défini par un logiciel, soutenu par la société Big Switch, il offre un point de gestion central pour les réseaux OpenFlow et il peut gérer des périphériques tel que open vswitch de manière transparente. Il prend également en charge une large gamme de commutateurs physiques OpenFlow, de sorte qu'il simplifie grandement la gestion du réseau [11]. Il est sous licence Apache et écrit en java [8] et il a pour mission :

- Installer / Supprimer les règles de transfert sur les commutateurs
- Découverte de la topologie : besoin de savoir à quoi ressemble le réseau (Link Layer Discovery Protocol (LLDP)).
- Statistiques : besoin de savoir ce qui se passe dans le réseau [11].

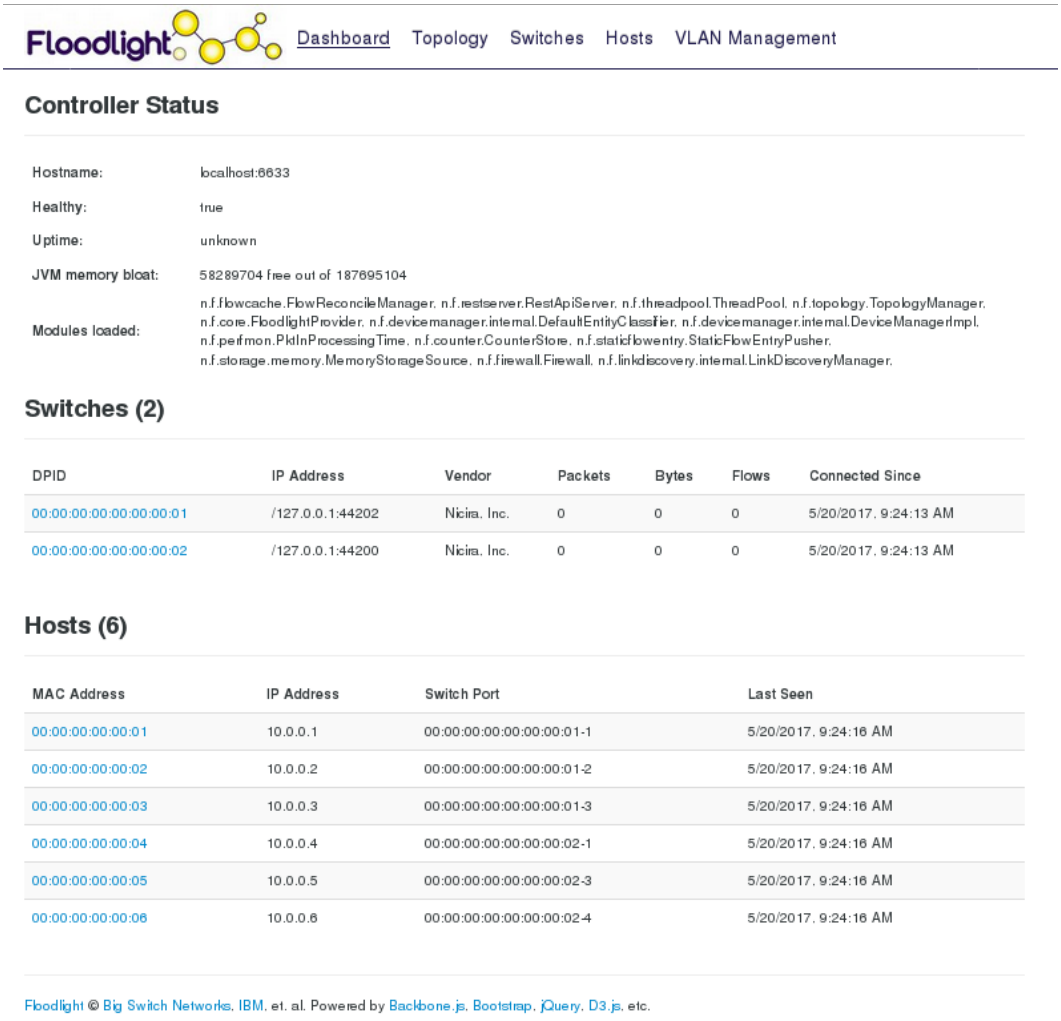


Figure III.3 : Interface web du Floodlight.

OpenDaylight est un projet open source pris en charge par IBM, Cisco, Juniper, VMWare et plusieurs autres grands fournisseurs de réseau. OpenDaylight est une plate-forme de contrôleur SDN implémentée en Java, il peut être déployé sur n'importe quelle plate-forme matérielle et système d'exploitation prenant en charge Java [13].

Le projet OpenDaylight repose sur des principes de développements ouverts et transparents, il vise à réunir les acteurs du réseau pour travailler sur des solutions communes. Big Switch Networks, Cisco, Citrix, Ericsson, IBM, Juniper Networks, Microsoft, Red Hat ou encore VMware sont ainsi les membres fondateurs d’OpenDaylight [13].

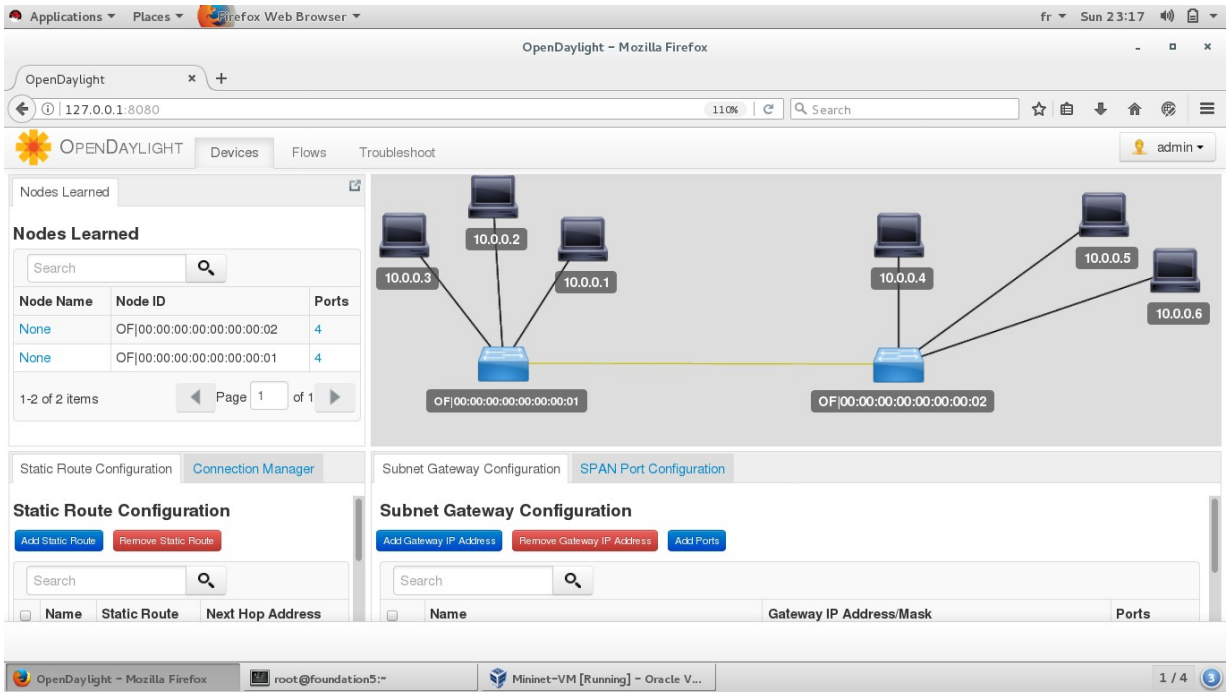


Figure III.4 : Interface web du Opendaylight

b. Comparaison entre Floodlight et Opendaylight :

Caractéristiques	**Floodlight**	**OpenDaylight**
Développé par	Big Switch Networks	Linux Foundation
Soutenu par	Big Switch Networks	Cisco, HP, IBM , Juniper, VMWare, etc.
Ecrit en Langage	java	java
Langages supportés	Java, Python et n'importe quel langage qui supporte Rest API	java
Open source	oui	oui
Interface utilisateur	web	web
Virtualisation	Mininet, OpenVswitch	Mininet, OpenVswitch
Interface	southbound (OpenFlow), northbound (Java, REST)	southbound (OpenFlow et autres protocoles), northbound (Java RPC)

Plateforme	Linux, Mac, Windows	Linux, Windows
Documentation	Site officiel	Moyen
Activité de liste de mailing	très élevé	Moyen

Tableau III.I : comparaison entre Floodlight et Opendaylight [12].

III.4.2 Évaluation des performances de Floodlight et Opendaylight:

Dans cette section, nous fournissons une analyse des performances de Floodlight et Opendaylight. Nous décrivons d'abord le test d'évaluation basé sur le débit et la latence, puis nous discuterons brièvement de nos observations.

Afin de réaliser ces tests sur les contrôleurs SDN, nous avons utilisé un laptop Lenovo, équipé d'un processeur Intel Core i3 et qui utilise Redhat enterprise 7 comme système d'exploitation. Nous avons également installé dans cette machine Floodlight et Opendaylight. Nous avons par la suite exécuté l'outil **Cbench** qui est un outil standard utilisé pour évaluer les contrôleurs SDN depuis une machine virtuelle.

Nous avons testé le débit et la latence des contrôleurs avec différents nombres de commutateurs : 8, 16 et 32. On a supposé que chaque commutateur supporte 1000 @MAC . Chaque test a été exécuté pour un minimum de 10 Minutes et répétés 3 fois.

a. Tests de latence :

La latence du contrôleur signifie combien du temps il faut pour traiter un seul paquet. À cet effet, nous avons exécuté Cbench en mode latence qui envoie un packet-In au contrôleur et attend une réponse avant d'envoyer un autre paquet.

La ligne de commande utilisée pour mesurer la latence est la suivante :

```
$ cbench   -c   10.0.2.2   -p   6633   -m  10000  -l  3  -s   16  -M   1000
```

À partir des résultats, nous avons observé qu'avec 8 commutateurs, Floodlight produit en moyenne de 8149 réponses /sec par commutateur. L'augmentation du nombre de commutateurs a entraîné une légère augmentation du nombre de réponses, de 8149 à 8 commutateurs à 8155 avec 16 commutateurs et à 8238 avec 32 commutateurs (Ces résultats sont détaillés dans la partie annexe).

Ces résultats sont également cohérents. Cependant, nous avons noté que la variabilité des résultats des tests augmente avec une augmentation du nombre de commutateurs. Les résultats des tests de latence du Floodlight sont résumés dans la Figure III.5.

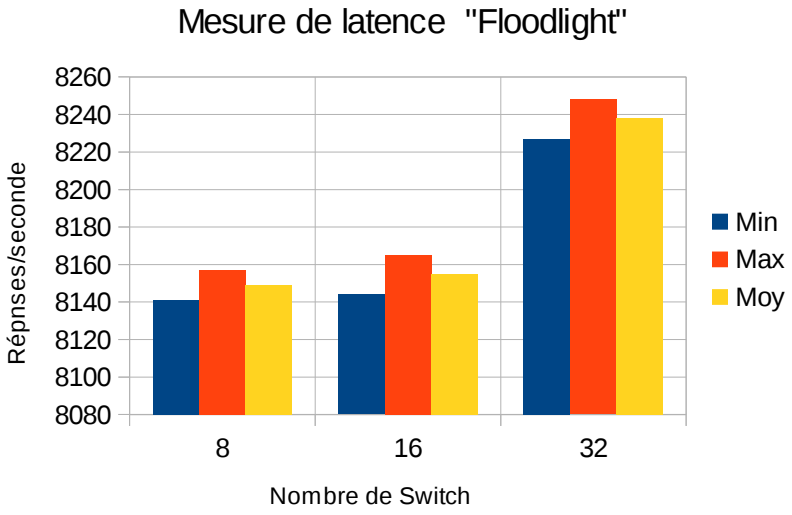


Figure III.5 : Résultats de la latence du Floodlight

Par rapport à Floodlight, le nombre de réponses sur Opendaylight était incohérent. Les réponses enregistrées étaient 3677 réponses/sec, 3211 réponses/sec et 3357 réponses/sec pour 8, 16 et 32 commutateurs respectivement (Ces résultats sont détaillés dans la partie annexe). Nous avons également remarqué que les performances de latence de Floodlight augmentent lorsque le nombre de commutateurs augmente, mais pour le contrôleur Opendaylight, il affiche un comportement inattendu car le nombre de réponses diminue lorsque le nombre de commutateurs augmente. Les résultats du mode de latence Opendaylight sont résumés dans la Figure III.6.

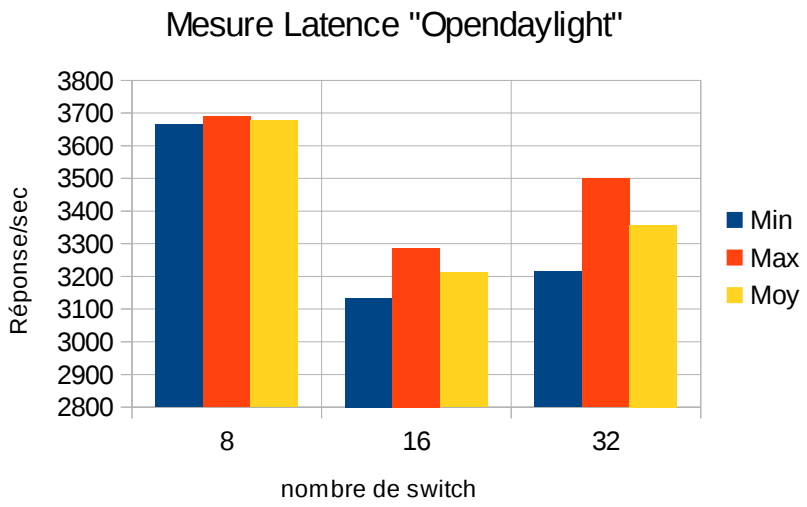


Figure III.6: Résultats de la latence d'opendaylight

Nous avons également remarqué que pour opendaylight, Cbench a échoué à produire un résultat dans plusieurs tests.

b. Tests de débit :

Dans les tests en mode débit, le contrôleur est évalué par rapport à la quantité de paquets qu'il peut traiter en une seconde. Nous avons également utilisé Cbench pour tester ce cas, dans lequel il envoie un trafic de contrôle, puis enregistre le nombre de réponses . Nous avons testé la performance du débit du contrôleur avec la ligne de commande suivante :

```
$ cbench -c 10.0.2.2 -p 6633 -m 10000 -l 3 -s 16 -M 1000 -t
```

Les scénarios de test étaient les mêmes que les tests de latence, les réponses moyennes / sec de Floodlight avec 8 commutateurs donnent 87808 réponses/sec. Lorsque nous avons augmenté le nombre de commutateurs à 16 et 32, le nombre de réponses par seconde a augmenté respectivement pour 90430 et 93096. Un résumé des résultats du débit Floodlight est illustré à la Figure III.7.

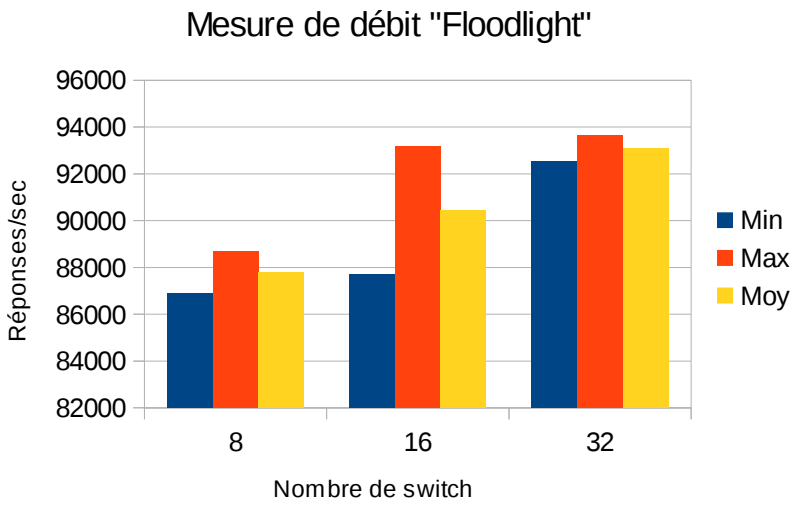


Figure III.7: Résultats de débit du Floodlight

Opendaylight a fourni des réponses très faibles en comparaison à Floodlight, et Cbench a échoué à produire un résultat dans plusieurs tests et a montré des réponses nulles. Les réponses moyennes par commutation enregistrées sur le contrôleur Opendaylight étaient respectivement de 21, 0 et 0 pour 8, 16 et 32 commutateurs. Ces résultats sont présentés dans la figure III.8 .

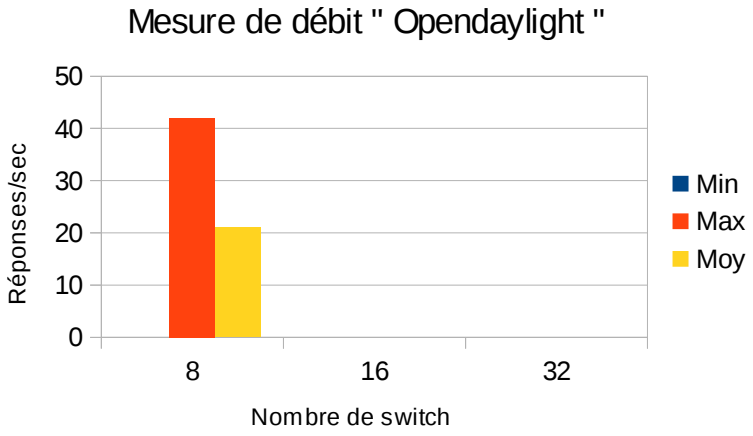


Figure III.8: Résultats de débit d'opendaylight

c. Discussion :

Selon les résultats, nous avons remarqué que Opendaylight a plusieurs problèmes de performance. Nous avons également constaté que de nombreux tests Opendaylight ne

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

produisaient aucun résultat. C'est Évident qu'un grand nombre de tests échouent si nous augmentons le nombre de commutateurs, la raison possible est que le protocole OpenFlow utilisé avec Opendaylight fonctionne mal. Suite à ces résultats de simulation, nous avons opté pour le choix du controlleur floodlight pour réaliser notre simulation SDN ainsi que l'application de gestion des vlans.

III.5 Conclusion

Nous avons présenté dans ce chapitre les différents outils utilisés pour déployer un réseau SDN, ainsi qu'une analyse comparative des contrôleurs Opendaylight et Floodlight. Nous avons constaté que le benchmarking d'Opendaylight avec Cbench n'a pas beaucoup de succès, pour cela on a opté pour un contrôleur Floodlight.

Dans la suite, nous allons démontrer comment mettre en place un réseau SDN, en utilisant Floodlight, ainsi comment utiliser les applications réseau (VLAN) pour faciliter la gestion.

Chapitre IV

Contribution 2 :

Réalisation d'une application SDN « gestion des VLANS »

SOMMAIRE

1. Introduction aux vlans

2. Intérêt des vlans

3. Difficulté des vlans dans les réseaux traditionnels

4. Le SDN, Solution pour une gestion simple des vlans

5.Implémentation d'une application de gestion des vlans avec SDN

6. Conclusion

IV.1 Introduction aux vlans

Pour débuter, il est important de comprendre ce qu’est un VLAN. Le terme «LAN» est utilisé pour désigner un réseau local [22], la lettre V en avant du terme «LAN» signifie «virtual» , alors un VLAN est un réseau local virtuel. Le concept de VLAN est utilisé afin d’avoir plusieurs réseaux indépendants sur le même équipement réseau. Cela évite d’avoir des équipements réseaux différents dans une entreprise lorsque nous voulons que deux départements ou fonctionnalités ne soient pas sur le même réseau ou vu l’un de l’autre. Pour des fins de sécurité, cela est très pratique [22].

Le meilleur exemple que nous pouvons donner c’est celui d'une entreprise qui possède un système téléphonique IP et qui ne veut pas que l’ensemble de son réseau informatique ait une influence sur la qualité de celui-ci. Alors, un réseau virtuel sera créé pour la téléphonie IP et un autre pour les ordinateurs [22].

Il est possible de faire cohabiter plusieurs VLAN sur un même équipement réseau. Il est également possible de faire communiquer deux VLAN ensemble [22].

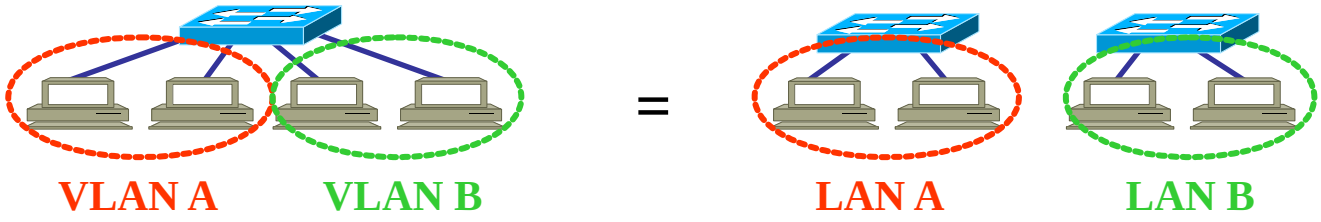


Figure IV.1 : VLAN

IV.2 Intérêt des VLANS

Il existe plusieurs intérêts d'avoir des VLANS dans un organisme. Par contre, il n’est pas nécessaire d’avoir des VLANS lorsque nous avons un petit réseau avec très peu de fonctionnalité. Voici quelques exemples de besoins qui nécessitent l’utilisation des réseaux virtuels [22] :

- Améliorer la gestion du réseau,
- Optimiser la bande passante,

- Séparer les flux,
- Fragmentation : réduire la taille d’un domaine de broadcast,
- Sécurité : permet de créer un ensemble logique isolé pour améliorer la sécurité. Le seul moyen de communiquer entre des machines appartenant à des VLAN différents est alors de passer par un routeur [22].

IV.3 Difficulté des VLANs dans les réseaux traditionnels

Dans les réseaux traditionnels, la configuration d'un VLAN reste un processus fastidieux, complexe et exposé aux erreurs, car les administrateurs réseau doivent configurer manuellement beaucoup de périphériques réseau d'une façon individuelle en utilisant la CLI au niveau du périphérique [14].

Il serait aussi plus difficile si ces périphériques étaient situés dans un emplacement plus vaste (par exemple, différents étages d'un grand bâtiment) [14].

Par conséquent, la gestion des VLAN est l'une des tâches les plus difficiles auxquelles sont confrontés les administrateurs réseau. Malgré que les constructeurs des périphériques réseau, tel que Cisco, ont pris en charge des nouveaux protocoles pour mieux gérer les VLANS (tel que VTP), mais ils sont encore très limité en termes de portée et de fonctionnalité [14].

Nous croyons que pour résoudre ces problèmes, il est nécessaire d'utiliser un outil de gestion de VLAN centralisé qui permet non seulement aux administrateurs réseau de configurer facilement les VLANS, mais aussi d'afficher une vue virtuelle pour la surveillance et le dépannage de chaque VLAN dans le réseau.

IV.4 Le SDN, Meilleure solution pour les VLANs

Par rapport aux réseaux traditionnels, l'introduction de SDN et OpenFlow peut simplifier la gestion des VLANs tout en offrant une grande flexibilité [14] :

- Améliorer l'efficacité du réseau grâce à une gestion et un contrôle centralisé, ainsi qu'un niveau élevé d'automatisation. Le contrôleur peut définir des règles et fournir un contrôle d'accès [14].
- Améliorer la disponibilité du service.
- Fournir une analyse avancée de toutes les ressources afin qu'on puisse facilement

surveiller et contrôler ces ressources et prendre des décisions stratégiques.

- Réduction des dépenses d'investissement , ainsi que maximiser l'utilisation des ressources [14].

Alors, bien que SDN et OpenFlow puissent remplacer toutes les infrastructures de gestion de réseau actuelles, l'utilisation du VLAN est encore nécessaire, et même les administrateurs de réseau continuent à les utilises afin d'avoir des réseaux sécurisés [14].

Donc, SDN et OpenFlow seraient une bonne solution pour faciliter la gestion des VLANS dans les réseaux d'entreprise et de campus.

IV.5 Implémentation d'une application de gestion des VLANs avec SDN

L'objectif de notre projet est d'avoir une application centralisée, qui peut aider l'administrateur réseau à configurer et à gérer le VLAN d'une manière souple et efficace. Les principales démarches de notre travail sont les suivantes:

- Mettre en place d'une topologie à l'aide d'un outil d'émulation basé sur Mininet, et d'un contrôleur Floodlight.
- Conception et développement d'une application de gestion de VLAN via une interface web (GUI).

Nous avons basé dans les différentes parties de nos tests sur 2 scenarios :

a. Scenario 1 : Utilisation d'un seul switch

Dans ce scenario, nous avons créé une topologie réseau , montrée dans la figure IV.2, qui se compose d'un seul switch connecté avec un contrôleur Floodlight , et avec plusieurs hôtes (de 4 à 6).

Puis nous avons configuré 2 VLANs: **vlan10** et **vlan20**, chacun se compose de 2 hosts, et évidemment le test du fonctionnement de ces VLANs a été fait via la commande **ping**.

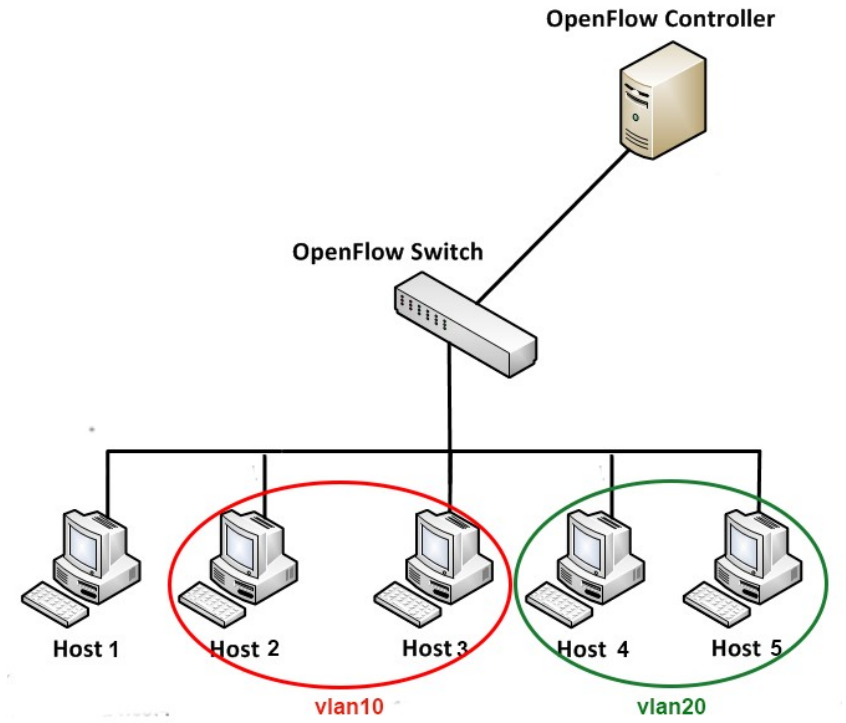


Figure IV.2 : Topologie du scenario 1

b. Scenario 2 : Utilisation de 2 switchs

Ici, nous avons créé une autre topologie réseau , montrée dans la figure IV.3, qui se compose de 2 switchs, chacun est connecté avec plusieurs hôtes (de 2 à 4), les 2 switches sont reliés entre eux et relié aussi avec un contrôleur Floodlight, puis nous avons configuré 2 VLANs : **vlan10** et **vlan20.**

Vlan10 se compose d'un hôte connecté au 1er switch et d'un autre hôte connecté au 2ème switch, et c'est pareil pour le vlan20. Nous avons testé par la suite la connectivité des hôtes d'un VLAN avec la commande ping.

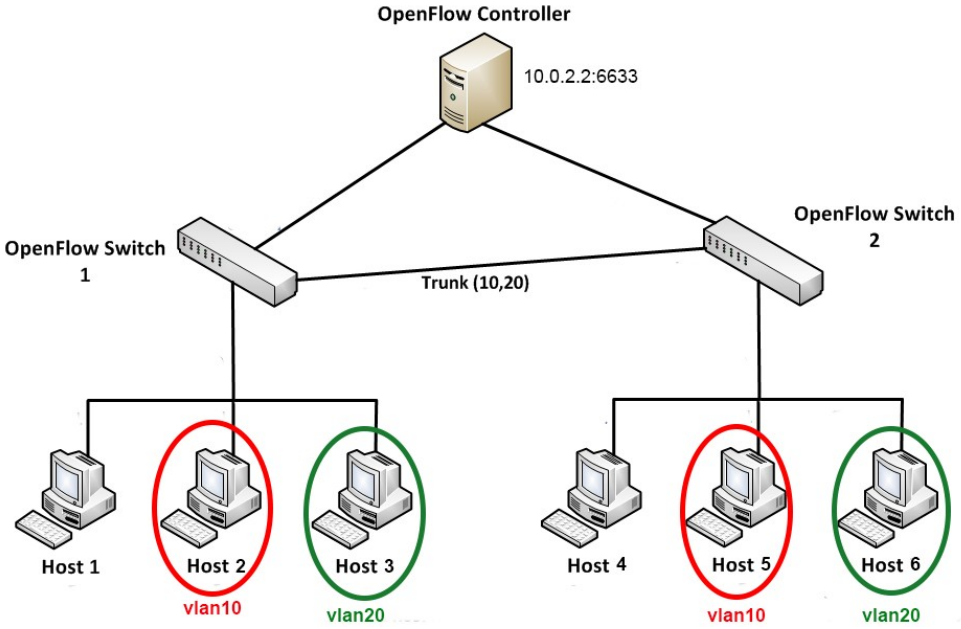


Figure IV.3 : Topologie du Scenario 2

IV.5.1 Mettre en place une topologie à l'aide du Mininet

IV.5.1.1 Création de topologie avec Mininet :

a.Scenario 1 :

pour créer la topologie du scénario 1 , nous avons exécuté la ligne de commande suivante :

```
$ sudo mn    --topo=single,5  --mac   --switch=ovs    --controller=remote,ip=10.0.2.2,port=6633
```

```
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --topo single,5  --mac  --switch ovs --controller=
remote,ip=10.0.2.2,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet>
```

Figure IV.4 : Création du topologie de scenario 1

Aprés avoir créer la topologie, nous pouvons la tester en exécutant des différentes commandes, nous pouvons utiliser la commande **help** pour avoir cette liste de commandes , comme montre la figure IV.5.

```
mininet>
mininet>
mininet>
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mininet> _
```

Figure IV.5 : Exécution de la commande help du Mininet.

Par exemple, la figure IV.6 nous affiche la sortie de la commande **dump** (la configuration de la topologie) ainsi que la sortie de la commande **net** (qui nous montre les liens entre hôtes et switch) .

```
mininet>
mininet>
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1490>
<Host h2: h2-eth0:10.0.0.2 pid=1493>
<Host h3: h3-eth0:10.0.0.3 pid=1495>
<Host h4: h4-eth0:10.0.0.4 pid=1497>
<Host h5: h5-eth0:10.0.0.5 pid=1499>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,
s1-eth5:None pid=1504>
<RemoteController{'ip': '10.0.2.2', 'port': 6633} c0: 10.0.2.2:6633 pid=1484>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet>
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0 s1-eth5:
h5-eth0
c0
mininet>
mininet>
mininet>
mininet>
```

Figure IV.6 : sortie des commandes net et dump

b. Scenario 2 :

pour créer la topologie du scénario 2 , nous avons utilisé un fichier python : **multiswitch-vlan.py**, puis nous avons appelé ce fichier dans la ligne de commande ci-dessous :

$ sudo mn --mac --custom multiswitch-vlan.py --topo=mytopo --switch=ovs --controller=remote,ip=10.0.2.2,port=6633

```
from mininet.topo import Topo
class MyTopo( Topo ):
    "Simple topology example."
    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        Host1 = self.addHost( 'h1' )
        Host2 = self.addHost( 'h2' )
        Host3 = self.addHost( 'h3' )
        Host4 = self.addHost( 'h4' )
        Host5 = self.addHost( 'h5' )
        Host6 = self.addHost( 'h6' )
        Switch1 = self.addSwitch('s1')
        Switch2 = self.addSwitch('s2')
        # Add links
        self.addLink( Host1, Switch1 )
        self.addLink( Host2, Switch1 )
        self.addLink( Host3, Switch1 )
        self.addLink( Host4, Switch2 )
        self.addLink( Switch1, Switch2 )
        self.addLink( Host5, Switch2 )
        self.addLink( Host6, Switch2 )
topos = { 'mytopo': ( lambda: MyTopo() ) }
~
~
"multiswitch-vlan.py" 27L, 878C                          1,1           All
```

Figure IV.7 : Fichier multiswitch-vlan.py

```
mininet@mininet-vm:~/mininet/perso-test$
mininet@mininet-vm:~/mininet/perso-test$ sudo mn --mac  --custom multiswitch-vla
n.py  --topo=mytopo --controller=remote,ip=10.0.2.2,port=6633
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
mininet>
```

Figure IV.8 : Topologie du scenario 2

IV.5.1.2 Contrôleur Floodlight

IV.5.1.2.1 Pré-requis :

- Pour étudier et tester un réseau SDN, nous avons utilisé une machine Linux (Redhat Enterprise linux 7), et installé les outils de développement JDK et eclipse.
- Puis, nous avons téléchargé et installé Floodlight. (voir la partie Annexe.)

IV.5.1.2.2 Exécution du Floodlight :

Pour démarrer Floodlight, nous exécutons le fichier **floodlight.jar** avec la commande suivante : **# java -jar floodlight.jar**

et via un navigateur web, nous accèdons à l'URL suivant : **http://localhost:8080/ui/index.html**

Les figures ci-dessous, représentent l'interface web du contrôleur Floodlight, nous avons choisi d'étudier dans cette partie une topologie du scenario 2 qui constitue de 2 switch, chacun est relié avec 3 hôtes.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

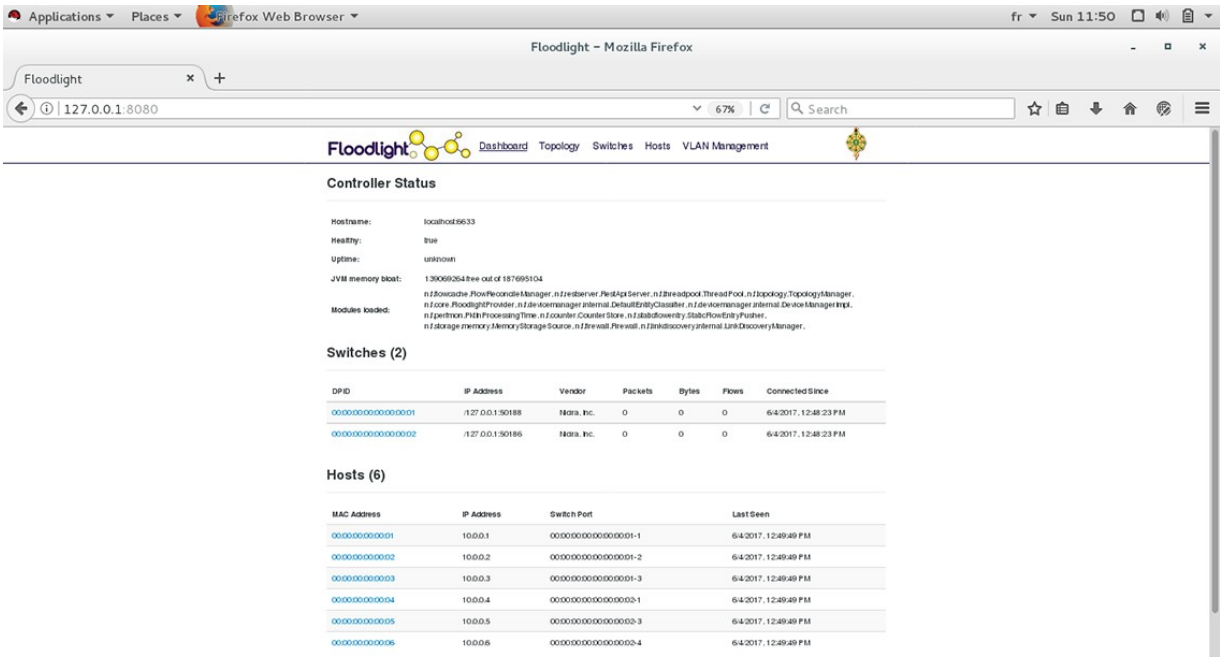


Figure IV.9 : Interface web du Floodlight

Une fois la topologie est créée avec mininet, tout le détail sera affiché dans la page d'accueil de Floodlight comme montre la figure IV.9.

Si on clique sur le lien switches, on aura l'adress IP et l'adress MAC du switch ainsi que le nombre de ports et des informations sur ces derniers tel que le flux circulant dedans (figure IV.10).

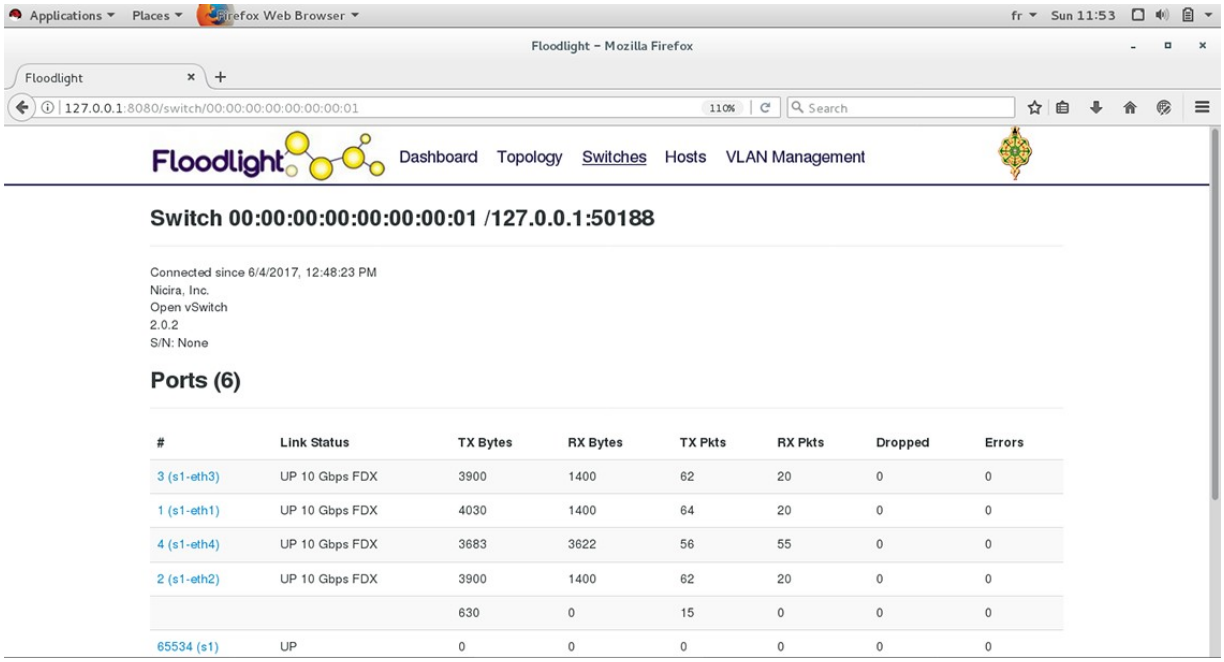


Figure IV.10 : Onglet switches de l'Interface web du Floodlight

Et si on clique sur le lien topology, on aura :

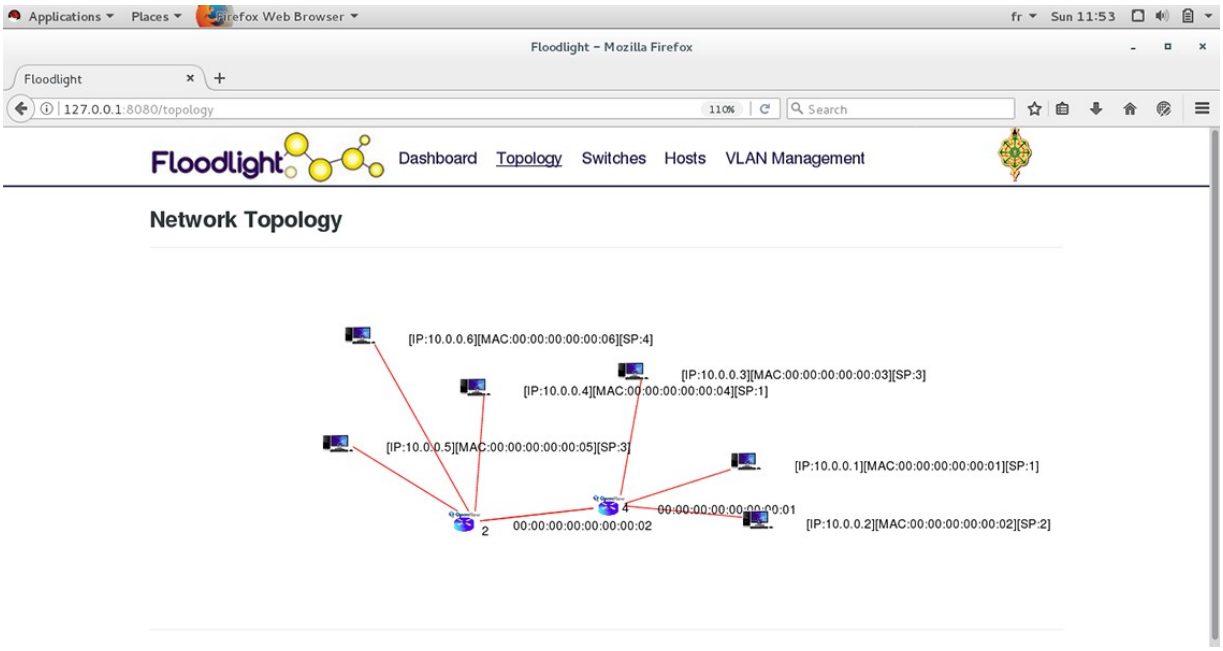


Figure IV.11 : Onglet Topology de l'Interface web du Floodlight

IV.5.2 Conception et développement d'une application de gestion de VLAN

IV.5.2.1 Présentation de l'application

Notre application «Vlan Management » permet de réaliser toutes les tâches de gestion de VLAN, y compris l'ajout d'un nouveau réseau VLAN, la suppression d'un VLAN ou de tous les VLAN, l'édition d'un VLAN existant et l'affichage de la topologie des réseaux VLAN.

Ce module est accessible via l'interface web du Floodlight, dans la partie header figure le lien **« VLAN Management »**, nous avons utilisé le langage javascript, qui peut être intégré directement au sein des pages web, pour y être exécuté sur le poste client.

Généralement, JavaScript sert à contrôler les données saisies dans des formulaires HTML, nous avons intégré le JavaScript dans la méthode **AJAX** (***A****synchronous* ***J****avascript* ***A****nd* ***X****ML*) qui sert à modifier le contenu des pages web, des scripts en langage JavaScript vont envoyer des demandes au serveur Web en utilisant l'objet **XMLHTTPRequest.**

Le transfert de données est géré par le JavaScript, en utilisant la technologie de formatage de données, **JSON** (*JavaScript Object Notation*) qui est un format de données textuel, permet de représenter des informations structurées.

a. Ajout d'un VLAN

Nous utilisons cette fonction afin de créer un nouveau réseau VLAN. Une fois ce dernier est créé, un ensemble de règles de flux sont installés dans la table de flux du switch.

Les informations requises pour la création sont : le nom et l'ID du VLAN, les ports auxquels les hôtes VLAN sont connectés. Nous avons également défini deux types de ports : **tagged** et **untagged ports**.

Les ports tagged (étiqueté) sont des ports où un identifiant VLAN sera ajouté dans l'en-tête du paquet. Les ports untagged sont des ports où l'ID VLAN sera supprimé de l'en-tête du paquet. Les ports de transfert sont les ports du commutateur qui n'ont pas d'hôte VLAN, mais qui sont impliqués dans le chemin d'accès au VLAN.

Chaque communication entre 2 d'hôtes consiste à deux entrées de flux, une pour envoyer le paquet dans un sens et l'autre utilisée dans l'autre sens. Pour cela, deux actions clés ont été utilisées pour la configuration des flux, **set-vlan-id** qui correspond à un tagged port et

strip-vlan qui correspond à un port untagged.

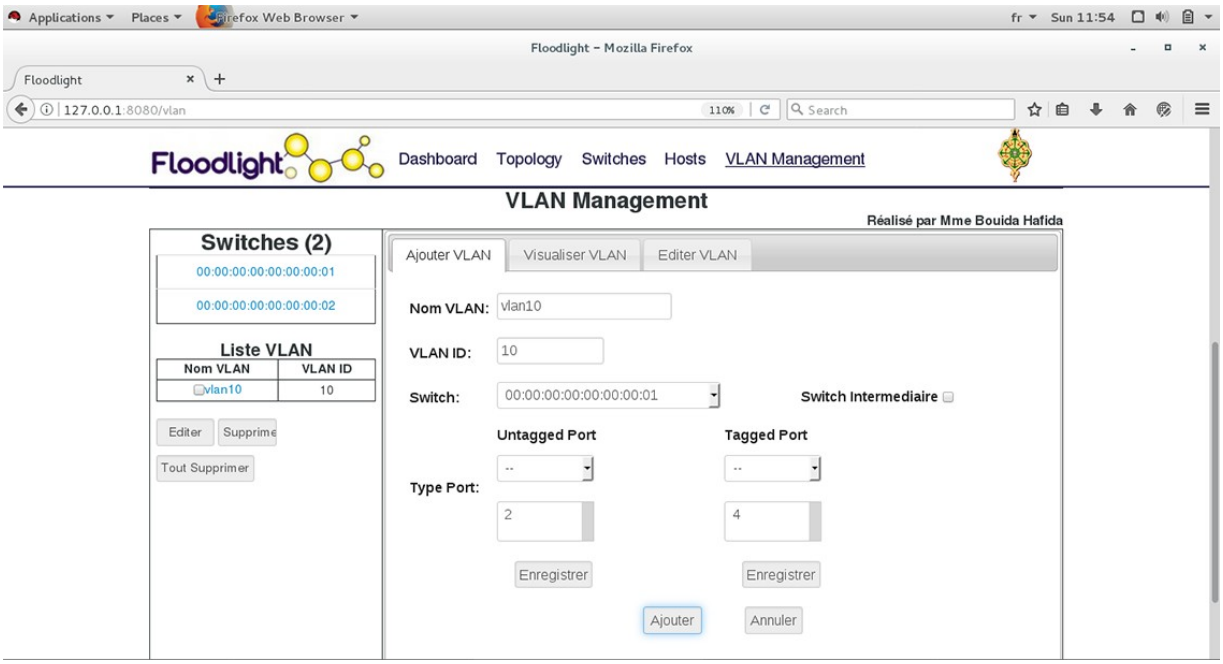


Figure IV.12 : Ajout d'un VLAN

H1 at port 1 Openflow Switch 01 (DPID=00:0a:6c:3b:e5:f9:7c:80)

"switch": "00:0a:6c:3b:e5:f9:7c:80","name":"VLAN 10-1","ingress-port":"1","actions":"set-vlan-id=10,output=2"

"switch": "00:0a:6c:3b:e5:f9:7c:80","name":"VLAN 10-2","ingress-port":"2", "vlan-id":"10","actions":"strip-vlan,output=1"

H2 at port 2 of Openflow Switch 03 (DPID = 00:0a:6c:3b:e5:f9:99:74)

"switch": "00:0a:6c:3b:e5:f9:99:74","name":"VLAN 10-5","ingress-port":"2","actions":"set-vlan-id=10,output=3"

"switch": "00:0a:6c:3b:e5:f9:99:74","name":"VLAN 10-6","ingress-port":"3", "vlan-id":"10","actions":"strip-vlan,output=2"

Figure IV.13 : Exemple d'une règle de flux VLAN ajoutée à la table des flux d'un switch

L'exemple illustré dans la Figure IV.13 montre que le port numéro 1 du Switch 01 et le numéro de port 2 du switch 03 sont des ports tagged (étiqueté), tandis que le port numéro 2 du switch 01 et le numéro de port 3 du switch 03 sont des ports untagged (non étiqueté).

b. Suppression d'un VLAN:

Cette fonction permet à un administrateur réseau de supprimer facilement le réseau VLAN en un seul clic sur l'interface graphique Web, au lieu d'avoir à le supprimer manuellement dans les commutateurs individuels. Cela faisant que toutes les entrées de flux dans tous les commutateurs appartenant au VLAN seront supprimées en même temps.

c. Modification d'un VLAN:

Cette fonction permet aux administrateurs réseau de modifier un VLAN existant, tel que ajout ou suppression d'un nouvel hôte d'un VLAN existant.

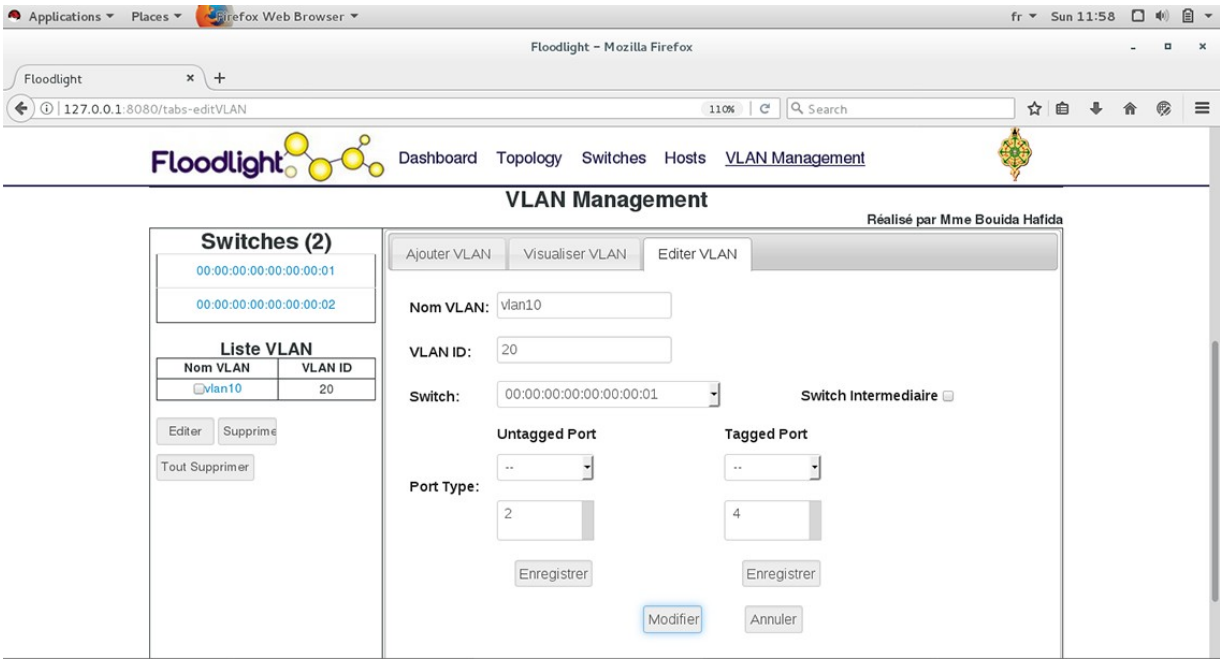


Figure IV.14 : Modification d'un VLAN

d. Visualisation d'un VLAN:

Cette fonction permet d'afficher tous les réseaux VLAN ainsi que leurs informations. Elle fournit également une vue séparée pour afficher plusieurs topologies de réseau VLAN simultanément.

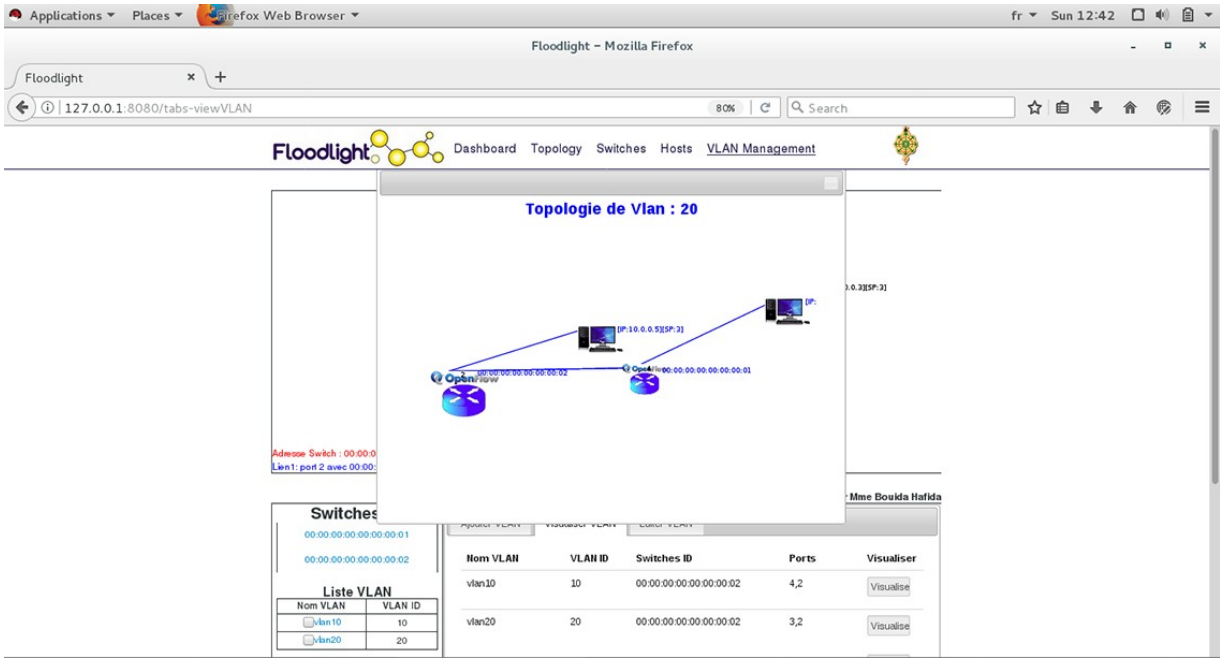


Figure IV.15 : Visualiser un VLAN

IV.5.2.2 Fonctionnement du vlan

Dans cette section, nous configurons deux VLANs vlan10 et vlan20 de 2 manières différentes :

a. Scenario 1 : Utilisation d'un seul switch

Une fois la topologie est créé (figure IV.2), et le VLAN aussi (figure IV.12), nous testons si les hosts appartenant à un seul VLAN (vlan10 ou vlan20) communiquent entre eux, vlan10 se compose du hôte 2 et hôte 3, et vlan20 se compose du hôte 4 et hôte 5.

Alors, nous observons dans la figure IV.16, que h2 peut pinger que h3 (et vice versa) et aussi h4 peut pinger que h5 (et vice versa), nous concluons donc que les 2 VLANs sont fonctionnels.

```
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X
h2 -> X h3 X X
h3 -> X h2 X X
h4 -> X X X h5
h5 -> X X X h4
*** Results: 80% dropped (4/20 received)
mininet>
```

Figure IV.16 : ping entre les hôtes (h2,h3: vlan10 et h4,h5:vlan20)

b. Scenario 2 : Utilisation de 2 switchs

Comme le scénario précédent, la figure IV.17 montre que les hôtes appartenant à un VLAN donné (h2 de Switch1 et h5 de Switch2: vlan10 , h3 de switch1 et h6 de switch2:vlan20) communiquent entre eux, d'où nous concluons que les 2 VLANs sont aussi fonctionnels.

```
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
mininet>
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
mininet>
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X h4 X X
h2 -> X X X h5 X
h3 -> X X X X h6
h4 -> h1 X X X X
h5 -> X h2 X X X
h6 -> X X h3 X X
*** Results: 80% dropped (6/30 received)
mininet>
```

Figure IV.17 : Scenario2 , ping entre les hosts (h2,h5: vlan10 et h3,h6:vlan20)

En résumé, nous pouvons confirmer, à partir des différents tests vu précédemment, que le SDN permet de faciliter la gestion des réseaux.

IV.6 Conclusion

Dans ce chapitre, nous avons exploité mininet, Floodlight et OpenFlow. Notre contribution principale est le développement d'une application de gestion des VLANs dans un environnement SDN. Notre application fournit une interface web interactive qui nous aide à gérer, créer, modifier et aussi visualiser un réseau VLAN, elle offre une flexibilité d'utilisation à un administrateur réseau.

En fournissant une vue topologique séparée pour chaque réseau VLAN, l'administrateur réseau peut surveiller facilement plusieurs réseaux VLAN en même temps.

De nos jours, VLAN n'est pas seulement présent sur le campus et les réseaux d'entreprise, mais est également largement utilisé pour gérer un réseau dans le système de cloud computing. Nous croyons que notre système peut être facilement adapté pour être utilisé dans cet environnement.

Conclusion générale

Découvrir le SDN a été un vrai plaisir. L'automatisation ouvre tellement de nouvelles perspectives, c'est le pouvoir de renforcer l'agilité, la sécurité ou encore les capacités du réseau grâce à de nouvelles solutions logicielles embarquant toujours plus d'intelligence. Car c'est bien là que réside le point fort du réseau qui désormais n'évolue plus avec la lourdeur du matériel mais avec la célérité du logiciel. Le SDN va changer donc le quotidien des administrateurs réseau, moins pris par des opérations répétitives et fastidieuses, ils pourront se consacrer à des tâches présentant plus de valeur ajoutée pour leur entreprise.

Grâce alors au SDN, les réseaux sont maintenant programmables, et la mise à disposition d'interfaces de programmation d'applications va permettre de programmer les équipements du réseau en utilisant différents langages.

Dans ce travail de fin d'étude, nous avons présenté une nouvelle technologie dans le domaine des réseaux : SDN. Notre démarche se base d'abord sur une étude comparative des contrôleurs floodlight et opendaylight, ensuite nous avons testé l'efficacité de cette nouvelle technologie en développant une application de gestion des vlans dans un environnement SDN, ce qui nous a permis de prouver ces avantages ainsi que son efficacité.

Ce projet a été pour nous une chance et une formidable opportunité de découvrir un réseau reposant sur les technologies SDN avec une console centrale et des outils « intelligents », capables de prendre en charge une grande partie de ces tâches fastidieuses.

L'automatisation opérée par le SDN est la chose la plus étonnante que nous avons pu voir dans notre projet. Nous avons, par exemple, passé peu de temps à créer une topologie réseau en utilisant des outils d'émulation puis connecté cette topologie avec un contrôleur SDN, l'automatisation limitant alors les risques d'erreurs humaines.

Le SDN remet en question le modèle des grands acteurs du réseau : s'ils n'intègrent pas le SDN à leur stratégie, ils pourraient se retrouver cantonnés à vendre des boîtiers de commutation. Les constructeurs réseaux doivent donc intégrer le SDN à leur stratégie.

Ainsi, pouvoir proposer des équipements ou technologies basées sur SDN devient une

priorité et nous citons quelques rapprochement ou rachat de petites startups spécialisées dans le SDN par les grands acteurs des réseaux et du logiciel : **Oracle** rachetant la startup Xsigo qui offre des services réseaux basés sur le SDN, **Juniper** s'alliant à la société Contrail, **Cisco** se rapprochant de la société Insiemi. Des acteurs comme **HP** se sont intéressés très tôt au SDN et se positionnent aujourd'hui parmi les leaders du marché.

Perspectives

Découvrir le SDN a été un vrai plaisir. L'automatisation ouvre tellement de nouvelles perspectives. Pour cela, il serait intéressant dans le futur d'étendre notre application de gestion de VLAN afin d'assurer le routage inter-vlan au niveau de la couche 3 (réseau), ou bien étudier la qualité de services au sein du SDN, la sécurité, le firewall ou encore étudier le SDN dans un environnement du cloud computing.

C'est bien là que réside le point fort du réseau SDN, désormais il n'évolue plus avec la lourdeur du matériel mais avec la célérité du logiciel.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Bibliographie

[1] Xavier Buche, **« Cloud universitaire avec OpenStack »**, mémoire d'ingénieur CNAM, Université Lille 1, Soutenu le 1 er décembre 2015.

[2] Thomas Paradis, **« Software-Defined Networkin »**, mémoire de Master, School of Information and Communication Technology KTH Royal Institute of Technology Stockholm, Sweden, le 20 Janvier 2014.

[3] Jérôme Durand, **« Le SDN pour les nuls »**, Montpellier, JRES 2015.

[4] Traore Issa, Kouassi Brou Médard, et Atta Ferdinand, **« Etude du nomadisme dans un Cloud éducatif administré par la technologie SDN/OpenFlow »**, Institut de recherches mathématique Université Félix Houphouët-Boigny, conférence WACREN 2016.

[5] Maxence Tury, **« Les risques d'OpenFlow et du SDN »**, ANSSI 2014.

[6] Enric Caceres, **« Le Protocole OpenFlow dans l'Architecture SDN »**, EFORT 2016.

[7] Ahmed Sonba et Hassan Abdalkreim, **« Performance Comparison Of the state of the art Openflow Controllers »**, mémoire de Master , Université Halmstad Sweden, Décembre 2014.

[8] Ouafae IFAKREN, « **Software Defined Network** », Rapport du semestre, hepia Genève 2015-2016.

[9] Messaoud Aouadj, **« AirNet, le modèle de virtualisation " Edge-Fabric " comme plan de contrôle pour les réseaux programmables »**, Thèse du doctorat de l'université de TOULOUSE, 2016.

[10] Karim Idoudi, «**IMPLÉMENTATION D'UN PLAN DE CONTRÔLE UNIFIÉ POUR LES RÉSEAUX MULTICOUCHES IP / DWDM**», Université du québec Montréal, 2014

[11] Brendan Tschaen, **« Floodlight Tutorial »**, CPS514, 23 Septembre 2015.

[12] Shiva Rowshanrad, Vajihe Abdi et Manijeh Kechtgari, **« Performance evaluation of SDN Controllers »** , Université de technologie Shiraz, Iran, IIUM, 2 Novembre 2016

[13] Zuhran Khan Khattak, Muhammad Awais et Adnan Ikbal, **« Performance evaluation of Opendaylight SDN Controller »**, Namal college, Mianwali, Pakistan, 2013

[14] Van-Giang Nguyen, Young-Han Kim, **«SDN-Based Enterprise and Campus Networks»** JIPS, Septembre 2016.

Webographie

[15] Les bases du SDN : http://www.lemagit.fr/actualites/2240228195/Les-bases-du-SDN-comprendre-les-atouts-de-la-programmabilite-et-du-controle-centralise , consulté le 06 Février 2017.

[16] Le nuage informatique : https://nouvelleinternet.wordpress.com/, consulté le 20 Février 2017.

[17] Qu'est-ce que la virtualisation ? : http://www.piloter.org/techno/support/virtualisation.htm, consulté le 20 Février 2017.

[18] Qu'est ce que le Cloud Computing ? : https://www.anthedesign.fr/autour-du-web/cloud-computing/ , consulté le 21 Février 2017.

[19] Open Source Floodlight : http://www.enterprisenetworkingplanet.com/netos/open-source-floodlight-extends-software-defined-networking-to-openstack.html , consulté le 29 Mars 2017.

[20] Documentation open vSwitch : http://docs.openvswitch.org/en/latest/intro/what-is-ovs/, consulté le 15 Avril 2017.

[21] Open vSwitch : http://www.linusnova.com/2013/04/open-vswitch/, consulté le 15 Avril 2017.

[22] Pourquoi avoir des VLAN : http://www.servlinks.com/blog/pourquoi-avoir-des-vlan/ consulté le 19 Mai 2017.

[23] Le SDN : https://www.riskinsight-wavestone.com/2013/10/le-sdn-ce-quil-faut-savoir-sur-le-cisco-killer/ consulté le 05 Mai 2017.

Annexe 1 : Installation du Floodlight

1. Pré-requis

- OS Recommandé : n'importe quelle distribution Linux, pour notre cas, nous avons travaillé avec le système Redhat Enterprise Linux 7.
- Installation de JDK, et ant.

```
# yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
# yum install ant
```

2. Téléchargement du floodlight

- Télécharger le source à partir du site github
- Lancer la commande ant (Ant est un projet du groupe Apache-Jakarta. Son but est de fournir un outil écrit en Java pour permettre la construction d'applications (compilation, exécution de tâches post et pré compilation, ...).

```
$ git clone git://github.com/floodlight/floodlight.git
$ cd floodlight
$ ant
```

3. Exécution du Floodlight

- Avant de lancer la commande ci-dessous, la commande java doit être ajouter à la variable d'environnement PATH.
- Exécuter le fichier **floodlight.jar** généré par ant.

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

4. Accéder au contrôleur Floodlight

- Ouvrir un navigateur web,
- Saisir l'URL ci-dessous :

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

5. Test du Floodlight :

- Il faut d'abord créer une topologie réseau en utilisant Mininet, puis lancer la commande pingall.
- On peut insérer des règles OpenFlow via une ligne de commande lancée dans la même machine où on a installé Floodlight en utilisant la commande **curl.**

```
curl -d '{"switch": "00:00:00:00:00:00:00:01", "name":"static-flow1",
"cookie":"0", "priority":"32768", "dst-mac":"00:00:00:00:00:01",
"active":"true", "actions":"output=5"}'
http://127.0.0.1:8080/wm/staticflowpusher/json
```

- Pour lister les flux insérés dans la table de flux:

```
curl -s http://localhost:8080/wm/staticflowpusher/list/all/json
```

- Pour supprimer un flux de la table de flux:

```
curl -s http://localhost:8080/wm/staticflowpusher/clear/all/json
```

```
curl -X DELETE -d '{"name":"static-flow1"}'
http://<controller_ip>:8080/wm/staticflowentrypusher/json
```

Annexe 2: Installation de Mininet

1. Pré-requis

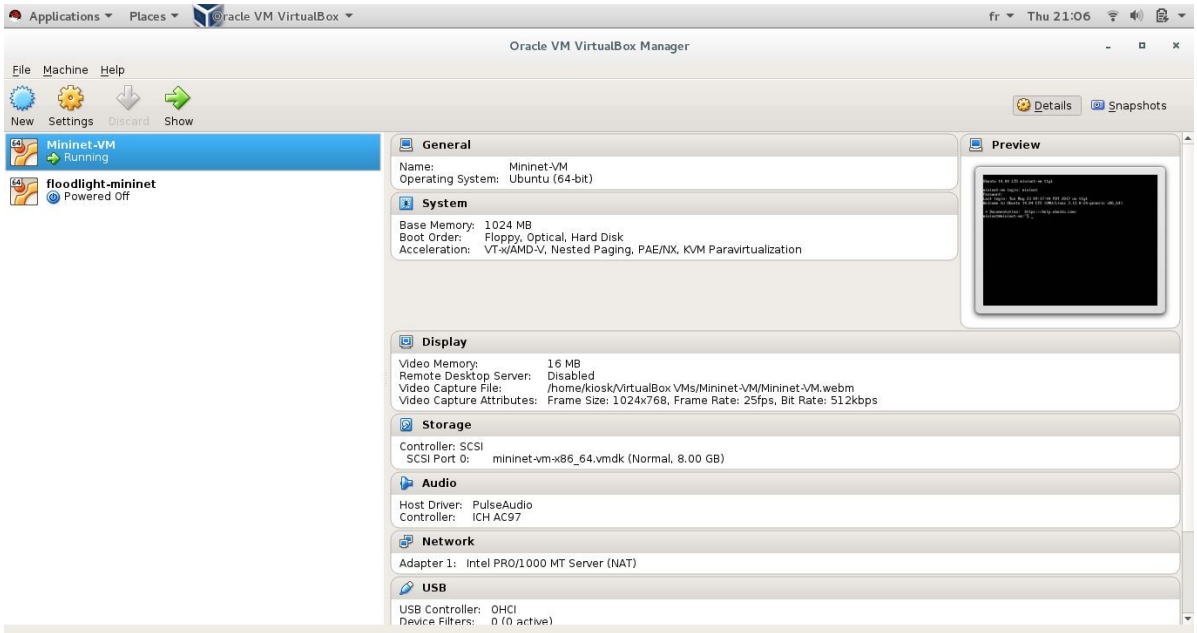
- La façon la plus simple pour commencer avec Mininet est de télécharger une machine virtuelle préinstallée sous Ubuntu.
- Cette machine virtuelle comprend Mininet lui-même, tous les binaires OpenFlow et les outils pré-installés, et ajuste la configuration du noyau pour prendre en charge les grands réseaux Mininet.

2. Téléchargement

- Télécharger la machine virtuelle à partir du lien :

```
$ https://github.com/mininet/mininet/wiki/Mininet-VM-Images
```

- Télécharger et installer un système de virtualisation. Le système le plus recommandé et que nous l'avons utilisé est VirtualBox, il est gratuit et il peut fonctionner sous windows et sous linux.



- Utiliser le compte : mininet avec le mot de passe mininet pour se connecter au système.

Annexe 3: Comparaison entre floodlight et opendaylight

1. Installation de l'outils cbench :

- Afin de d'évaluer les performances de Floodlight et Opendaylight, nous avons utilisé l'outils **cbench.**
- Cbench (controller benchmarker) est un programme pour tester les contrôleurs OpenFlow . Cbench émule un ensemble de commutateurs connecté à un contrôleur, en envoyant des messages Packet-in et surveillant les flux .
- Pour installer cbench, nous avons suivi la procédure suivante :

```
$ sudo apt-get install autoconf automake libtool libsnmp-dev libpcap-dev
$ git clone git://gitosis.stanford.edu/oflops.git
$ cd oflops; git submodule init && git submodule update
$ git clone git://gitosis.stanford.edu/openflow.git
$ cd openflow; git checkout -b release/1.0.0 remotes/origin/release/1.0.0
$ wget http://hyperrealm.com/libconfig/libconfig-1.4.9.tar.gz
$ tar -xvzf libconfig-1.4.9.tar.gz
$ cd libconfig-1.4.9
$ ./configure
$ sudo make && sudo make install
$ cd ../../netfpga-packet-generator-c-library/
$ sudo ./autogen.sh && sudo ./configure && sudo make
$ cd ..
$ sh ./boot.sh
./configure --with-openflow-src-dir=<absolute path to openflow branch>; make
$ sudo make install
$ cd cbench
```

2. Exécution du cbench :

- Les options les plus utilisées avec cbench sont :

M	Nombre de MACs / hôtes à émuler par émulate per switch
s	Nombre de switches à émuler
t	throughput (mode débit)

- La ligne de commande utilisée est :

```
./cbench -c localhost -p 6633 -m 10000 -l 10 -s 16 -M 1000 -t
```

3. Résultat de l'exécution du cbench :

- Nous avons lancé la commance cbench depuis une machine virtuelle (machine distante), et nous avous installé floodlight et opendaylight sur la machine locale.
- Le résultat obtenu est décrit ci-dessous :

a. Mesure du débit « Floodlight »

```
$ cbench -c 10.0.2.2 -p 6633 -m 10000 -l 3  -M 1000 -s 8 -t
cbench: controller benchmarking tool
running in mode 'throughput'
connecting to controller at 10.0.2.2:6633
faking 8 switches offset 1 :: 3 tests each; 10000 ms per test
with 1000 unique source MACs per switch
learning destination mac addresses before the test
starting test with 0 ms delay after features_reply
ignoring first 1 "warmup" and last 0 "cooldown" loops
connection delay of 0ms per 1 switch(es)
debugging info is off
04:02:33.271 8   switches: flows/sec:  102114  101527  92474  88871  86941  88714  100042  100863   total = 76.150671 per ms
04:02:43.372 8   switches: flows/sec:  115087  115087  109534  115088  102300  115088  102300  112643   total = 88.706295 per ms
04:02:53.475 8   switches: flows/sec:  102305  107471  113739  102304  109739  115091  115091  103488   total = 86.911614 per ms
RESULT: 8 switches 3 tests min/max/avg/stdev = 86911.61/88706.30/87808.95/897.34 responses/s
```

```
$ cbench -c 10.0.2.2 -p 6633 -m 10000 -l 3  -M 1000 -s 16 -t
cbench: controller benchmarking tool
running in mode 'throughput'
connecting to controller at 10.0.2.2:6633
faking 16 switches offset 1 :: 3 tests each; 10000 ms per test
with 1000 unique source MACs per switch
learning destination mac addresses before the test
starting test with 0 ms delay after features_reply
ignoring first 1 "warmup" and last 0 "cooldown" loops
connection delay of 0ms per 1 switch(es)
debugging info is off
04:03:25.385 16  switches: flows/sec:  54222  38757  41172  44935  39706  54221  51282  38970  40921  45613  48875  37089
                                       47856  48770  36359  51157   total = 71.989204 per ms
04:03:35.490 16  switches: flows/sec:  63941  63942  63942  57906  51154  51154  63942  51154  63942  63942  56356  63273
                                      51153  51154  63941  51154   total = 93.171617 per ms
04:03:45.594 16  switches: flows/sec:  53402  51150  51150  51150  51150  63938  51150  63938  51150  51150  56514  51150
                                      51151  63937  51150  63938   total = 87.689906 per ms
RESULT: 16 switches 3 tests min/max/avg/stdev = 87689.91/93171.62/90430.76/2740.86 responses/s
```

b. Mesure du débit « Opendaylight »

```
cbench -c 10.0.2.2 -p 6633 -m 10000 -l 3  -M 1000 -s 8 -t
cbench: controller benchmarking tool
running in mode 'throughput'
connecting to controller at 10.0.2.2:6633
faking 8 switches offset 1 :: 3 tests each; 10000 ms per test
with 1000 unique source MACs per switch
learning destination mac addresses before the test
starting test with 0 ms delay after features_reply
ignoring first 1 "warmup" and last 0 "cooldown" loops
connection delay of 0ms per 1 switch(es)
debugging info is off
05:03:12.904 8   switches: flows/sec:  9337  12  12  12  12  0  0  1   total = 0.937949 per ms
05:03:23.983 8   switches: flows/sec:  464  0  0  0  0  0  0  0   total = 0.042268 per ms
05:03:34.108 8   switches: flows/sec:  0  0  0  0  0  0  0  0   total = 0.000000 per ms
RESULT: 8 switches 3 tests min/max/avg/stdev = 0.00/42.27/21.13/21.13 responses/s

$ cbench -c 10.0.2.2 -p 6633 -m 10000 -l 3  -M 1000 -s 16 -t
cbench: controller benchmarking tool
running in mode 'throughput'
connecting to controller at 10.0.2.2:6633
faking 16 switches offset 1 :: 3 tests each; 10000 ms per test
with 1000 unique source MACs per switch
learning destination mac addresses before the test
starting test with 0 ms delay after features_reply
ignoring first 1 "warmup" and last 0 "cooldown" loops
connection delay of 0ms per 1 switch(es)
debugging info is off
05:12:21.464 16 switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0   total = 0.000 per ms
05:12:32.259 16 switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0   total = 0.000 per ms
05:12:42.837 16 switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0   total = 0.000 per ms
RESULT: 16 switches 3 tests min/max/avg/stdev = 0.00/0.00/0.00/0.00 responses/s
```

c. Mesure de latence « Floodlight »

```
$ cbench -c 10.0.2.2 -p 6653 -m 10000 -l 3  -M 1000 -s 8
cbench: controller benchmarking tool
running in mode 'latency'
connecting to controller at 10.0.2.2:6633
04:42:01.240 8   switches: flows/sec:  6853  6798  6649  6535  6434  6287  4431  4134   total = 4.811791 per ms
04:42:11.342 8   switches: flows/sec:  11645  11605  11532  11508  11394  11211  6465  6061   total = 8.141659 per ms
04:42:21.443 8   switches: flows/sec:  11680  11592  11571  11516  11439  11243  6434  6110   total = 8.157898 per ms
RESULT: 8 switches 3 tests min/max/avg/stdev = 8141.66/8157.90/8149.78/8.12 responses/s
```

```
$ cbench -c 10.0.2.2 -p 6653 -m 10000 -l 3  -M 1000 -s 16
cbench: controller benchmarking tool
   running in mode 'latency'
   connecting to controller at 10.0.2.2:6633
04:44:03.578 16  switches: flows/sec:  5373  5845  5123  5539  5316  5116  5684  5402  5238  5479  4976
                                       4909  5000  4837  3017  2797   total = 7.963961 per ms
04:44:13.680 16  switches: flows/sec:  5494  5483  5464  5457  5456  5427  5429  5421  5403  5385  5364
                                       5332  5306  5212  3132  2899   total = 8.165984 per ms
04:44:23.783 16  switches: flows/sec:  5498  5481  5485  5463  5441  5416  5422  5399  5368  5356  5346
                                       5306  5263  5169  3119  2915   total = 8.144115 per ms
RESULT: 16 switches 3 tests min/max/avg/stdev = 8144.12/8165.98/8155.05/10.93 responses/s
```

d. Mesure de latence « Opendaylight »

```
$ cbench -c 10.0.2.2 -p 6653 -m 10000 -l 3  -M 1000 -s 8
cbench: controller benchmarking tool
   running in mode 'latency'
   connecting to controller at 10.0.2.2:6633
05:00:50.575 8   switches: flows/sec:  37073  9  9  9  1  1  1  1   total = 3.710400 per ms
05:01:00.676 8   switches: flows/sec:  36644  0  0  0  0  0  0  0   total = 3.664399 per ms
05:01:10.778 8   switches: flows/sec:  36899  0  0  0  0  0  0  0   total = 3.689899 per ms
RESULT: 8 switches 3 tests min/max/avg/stdev = 3664.40/3689.90/3677.15/12.75 responses/s

$ cbench -c 10.0.2.2 -p 6653 -m 10000 -l 3  -M 1000 -s 16
cbench: controller benchmarking tool
   running in mode 'latency'
   connecting to controller at 10.0.2.2:6633
04:59:29.873 16  switches: flows/sec:  27836  81  81  81  81  81  81  81  81  77
                                       73  73  29  29  29  29   total = 2.882279 per ms
04:59:39.974 16  switches: flows/sec:  32874  0  0  0  0  0  0  0  0  0  0  0
                                       0  0  0  0   total = 3.287399 per ms
04:59:50.076 16  switches: flows/sec:  31348  0  0  0  0  0  0  0  0  0  0  0
                                       0  0  0  0   total = 3.134799 per ms
RESULT: 16 switches 3 tests min/max/avg/stdev = 3134.80/3287.40/3211.10/76.30 responses/s
```