République Algérienne Démocratique ET Populaire
Ministère de l'enseignement supérieur et de la Recherche Scientifique.
Université Amar Teliji de Laghouat


Faculté des sciences
Département mathématique et informatique

# Mémoire de Master

**Domaine :** Mathématique et informatique (MI)
**Filière :** Informatique
**Option :** **R**éseaux, **S**ystèmes et **A**pplications **R**épartis (ReSar)

Mémoire de fin d'étude en vue d'obtention de diplôme : Master en Réseaux, Systèmes et Applications Répartis

PRÉSENTÉ PAR :

**DJEKIDEL Sara** **LAKHDARI Fatima**

Thème :

## Etude et analyse de performance des réseaux SDN

*Soutenu publiquement devant le jury composé de :*

| | | |
|---|---|---|
| Mr | Université de Laghouat | Président |
| Mr | Université de Laghouat | Examinateur |
| Mr | Université de Laghouat | Examinateur |
| Mr N.CHAIB | Université de Laghouat | Encadreur |

ANNÉE UNIVERSITAIRE 2015/2016

# *Dédicaces*

*JE dédie affectueusement ce modeste travail :*

***A mes chers parents :***

- ***À ma chère mère*** qui a oeuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

- ***À mon cher père MOHAMED*** qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

***À mes frères et sœurs*** qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

***À mes amis*** et spécialement à **Ilham** et **Noussiba**.

***À tous mes enseignants*** *qui ne m'ont guère privé de leur savoir et de leur bienséance.*

***À mes collègues de la promotion*** .
***À ma meilleure amie et mon binôme*** **Lakhdari Fatima** et à toute sa famille.
***À mes amis Guelouma Fatima, Djoudi Amina et Boussebsi Mbarka*** leurs aides et leurs conseils m'obligent de leur témoigner mon profond respect.

***À mon cher fiancé Hamdi Abdelkader*** pour son soutien permanent, ses encouragements et son aide à surmonter tous les moments difficiles tout au long de la réalisation de mémoire.

*Enfin à tous qui portent le nom* ***DJEKIDEL*** *et à tous ceux qui connaissent.*

*Djekidel Sara*

# *Dédicaces*

*Je dédie affectueusement ce modeste travail :*

***A Ma tendre Mère MERIEM*** : Tu représentes pour moi la source de Tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études.

***A Mon très cher Père FATEH*** : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation le long de ces années.

***A mes chers beaux-parents.***
***A mon cher mari ISMAIL*** :
***A tous mes frères et sœurs, ainsi que leurs enfants.***
***A mes belle sœurs, ma beau-frère.***
***A toute mes très chers amis***
***A tous mes enseignants depuis mes premières années d'études.***
***A tous les membres de ma promotion.***
***A tous ceux qui me sont chers et que j'ai omis de citer.***

*Lakhdari Fatima*

# *Remerciements*

TOUTE *notre gratitude et remerciements au bon* ***DIEU*** *qui nous a donné la force, le courage et la volonté d'élaborer ce travail.*

*Nous adressons notre profond remerciement à Mr* ***CHAIB NOURED-DINE*** *Maitre assistant à l'université de Laghouat pour avoir dirigé ce travail avec tant de bienveillance. Nous le remercions pour ses précieux conseils et sa disponibilité.*

*Nous adressons particulièrement nos remerciements aux membres du Jury.*

*Nous remercions tous le personnel de l'université de Amar TELIJI de Laghouat , l'université qui nous a accueilli bras ouvert , nos remerciements vont particulièrement aux enseignants et administrateurs du département de mathématique et d'informatique.*

*Nos remerciements les plus sincères à toutes les personnes qui nous ont aidé de prés ou de loin à accomplir notre travail.*

*Enfin, nous exprimons nos vifs remerciements à toute notre famille et spécialement à nos parents .*

*Lakhdari Fatima et Djekidel Sara*

# Résumé

DEPUIS quelques temps, un nouveau concept pour l'architecture des réseaux est devenu à la mode, c'est le SDN (Software Defined Networking) qui est né en 2008 du travail des équipes de recherche des Université de Berkeley et Stanford.

Le SDN n'est pas simplement un travail de recherche universitaire, mais bien une nouvelle organisation des réseaux qui intéresse fortement tous les grands acteurs du réseau aujourd'hui.

Les réseaux SDN proposent un nouveau paradigme au niveau du fonctionnement des réseaux informatiques où les routeurs et les commutateurs exécutent des protocoles distribués prédéfinis comme l'OpenFlow .Ce dernier propose de les remplacer par des plates-formes.

Ce mémoire a pour but d'expliquer et définir les réseaux SDN et le protocole OpenFlow. L'objectif n'est pas de faire une analyse technique exhaustive, mais simplement de passer en revue les concepts clés et les éventuels impacts sur les futurs réseaux de données.

**Mots-Clés :** SDN, OpenFlow, table de flux, correspandance, Switch, contrôleur

# Abstract

For some time a new concept for network architecture became fashionable, the SDN (Software Defined Networking) who was born in 2008 the work of research teams from the University of Berkeley and Stanford.

SDN is not simply an academic research work, but a new organization network that heavily involves all major players in the network today.

The SDN networks offer a new paradigm in the operation of computer networks. Where routers and switches running distributed protocols, OpenFlow proposed to replace them by platforms.

This thesis aims to explain and define SDN networks and OpenFlow. The objective is not to make an exhaustive technical analysis, but simply to review key concepts and potential impacts on future data networks.

**Key-words :** SDN, OpenFlow, flow table, matching, Switch,controller

# Table des matières

# Liste des tableaux

# Table des figures

# Glossaire

**SDN** Software Defined Networking

**TLS** Transport Layer Security

**VM** Virtual machine

**IT** Information technology

**NAT** Network address translation

**QoS** Quality of Service

**ONF** Open Networking Foundation

**OF** OpenFlow

**CLI** Command line interface

**SSL** Secure Sockets Layer

**SSH** Secure Shell

**ICMP** Internet Control Message Protocol

**VLAN** Virtual Local Area Network

**DoS** denial of service

**MAC** Media Access Control

**ARP** Address Resolution Protocol

# Introduction générale

RÉCEMMENT, les data centers ont reçu beaucoup d'attentions considérables a cause de ses capacités de stockage pour des grandes quantités de données.

Aujourd'hui les grandes entreprises utilisent des data centers pour leur grande échelle d'application et la technologie de l'information(IT).

La virtualisation et le cloud computing, changent la façon d'utiliser les data-centers. La virtualisation permet une utilisation plus efficace des ressources d'IT avec des niveaux élevés de contrôle, ainsi que le cloud computing s'étend ces avantages en permettant aux organismes de répondre à leurs besoins en utilisant des modèles souples, à la demande et rapide. Ces technologies permettent aux entreprises de répondre aux demandes et de fournir une plus grande souplesse pour les data centers.

Malgré le développement rapide de technologies tel que le cloud computing et la virtualisation, les technologies de réseaux manquent encore de cohérence, parce que la plupart des réseaux actuels ne sont pas développés à considérer la virtualisation et le cloud computing.

Les topologies statiques nécessitent une intervention manuelle pour déployer et migrer des machines virtuelles (VM) qui peuvent rendre les réseaux de devenir un goulot d'étranglement dans le futur de développement de l'IT .

Le SDN (Software Defined Networking) est un nouveau paradigme de gestion de réseau qui apporte des nouvelles fonctionnalités et permet de résoudre de nombreux problèmes difficiles des réseaux conventionnels. Cette approche est basée sur la séparation de l'intelligence du réseau hors de la commutation de paquets et le mettre dans un contrôleur logique centralisé.

Le contrôleur est responsable aux décisions de transfert qui sont placées dans les switchs via des protocoles standards, tels que l'OpenFlow.

La motivation des SDNs consiste à effectuer un système d'exploitation de réseau, où les tâches réseau peuvent se faire sans l'ajout des logiciels supplémentaires pour chaque élément de commutation, permettant de développer des applications qui contrôlent le switch en l'exécutant sur un système d'exploitation de réseau.

Le protocole OpenFlow est mis en place pour unifier l'interface entre la commutation matérielle et le contrôleur dans le paradigme SDN.

L'objectif de ce travail est de faire une étude sur les réseaux SDN et le protocole OpenFlow, c'est pour ça on a organisé le mémoire en trois chapitres comme suit :

- Le premier chapitre est une introduction générale aux réseaux SDN dans lequel nous aborderons toutes les généralités liées à ce type de réseaux ainsi qu'un aperçu de l'architecture SDN.

- Dans le deuxième chapitre on parle sur le protocole OpenFlow, le Switch OpenFlow avec ses composants, les étapes de correspondance, ensuite le contrôleur OpenFlow.
- Dans le troisième chapitre nous décrivons sur la plateforme Mininet, son architecture, son fonctionnement, les différents messages Open flow et l'évaluation de performance du deux contrôleur POX et NOX.
  Nous présentons les différents paramètres utilisés pour l'emulation, ainsi que le suivi des mesures de performances adoptées pour son évaluation. Ces résultats de l'emulations vont nous permettre de comprendre son comportement.

- Enfin nous terminerons ce mémoire par donner une conclusion générale et tracer des axes pour de futurs travaux.

# Chapitre 1

# Les réseaux SDN

## 1.1 Généralités sur les réseaux SDN

### 1.1.1 Introduction

Les réseaux informatiques sont généralement construits à partir d'un grand nombre de périphériques réseau tels que les routeurs, les commutateurs et de nombreux types de boîtiers de médiation (à savoir, des appareils qui manipulent le trafic ayant des autres usages que la transmission de paquets, tel qu'un firewall) sur lesquelles sont appliqués de nombreux protocoles complexes.

Les réseaux et data center traditionnels ont atteint un point de rupture. ils n'ont pas été conçus pour les applications dévoreuses de bande passante qui prolifèrent aujourd'hui ni même pour les exigences croissantes en matière de vitesse, d'évolutivité et de résilience . Dans ce contexte, les réseaux définis par logiciel (SDN) offrent une approche révolutionnaire conçue pour surmonter tous ces obstacles.

Dans ce chapitre, nous allons essayer de présenter les réseaux SDN, en commençant par définir ce nouveau type de réseaux et ses objectifs, puis nous allons décrire l'architecture de l'SDN et les différents modéles pour le SDN.

### 1.1.2 Les réseaux SDN

Selon la définition officielle d'Open Networking Foundation (ONF)[Fou12] le SDN est une architecture de réseau émergeant où le contrôleur de réseau est découplé au transfert de paquet et il est directement programmable.

Une autre définition de SDN est la suivante : le SDN est une nouvelle approche de la conception, de l'architecture et de la gestion des réseaux. Le concept de base du SDN consiste à découpler le plan de contrôle (le cerveau) et le plan de données (les muscles) du réseau, où le plan de contrôle peut contrôler plusieurs dispositifs voir 1.1 [Fou16].

La solution SDN est une solution Cloud intégrale et flexible qui nous permet de libérer toute la puissance et les capacités d'analyser les réseaux.

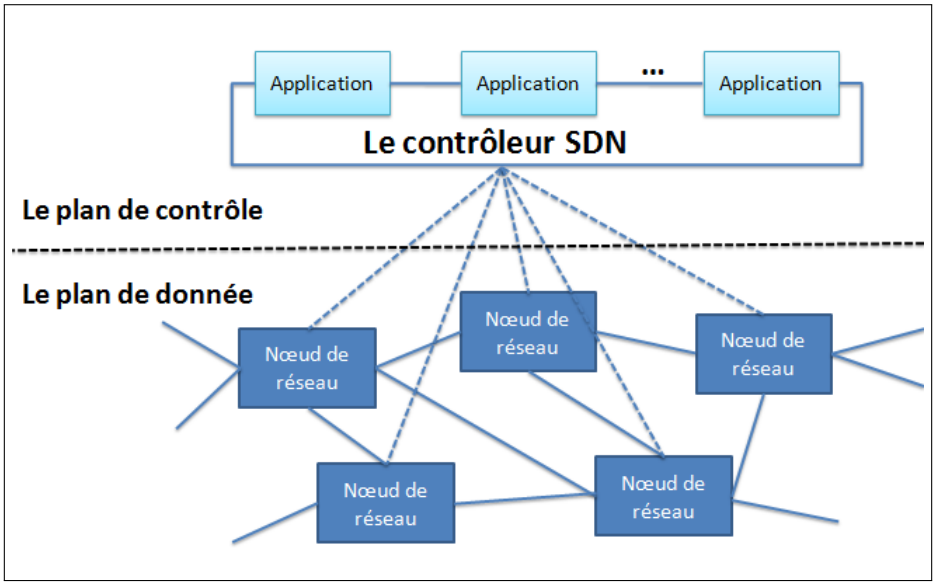


**Figure** 1.1 – *SDN « Software Defined Networking »*

### 1.1.3 L'architecture des réseaux convontionnels

Les réseaux d'aujourd'hui sont isolés et séparés physiquement pour répondre aux besoins de l'industrie, organismes et de services des utilisateurs, où nous avons le plan de contrôle et le plan de données sont dans le même périphérique comme le montre la figure 1.2.

Ce genre d'architecture du réseau a bien fonctionné dans le passé, mais aujourd'hui avec le monde virtualisé, il sera difficile voire impossible pour un réseau conventionnel de répondre aux nouvelles exigences virtuelles.

Avec les budgets limités d'aujourd'hui, les entreprises de technologie de l'information cherchent à virtualiser la plupart de leurs serveurs.

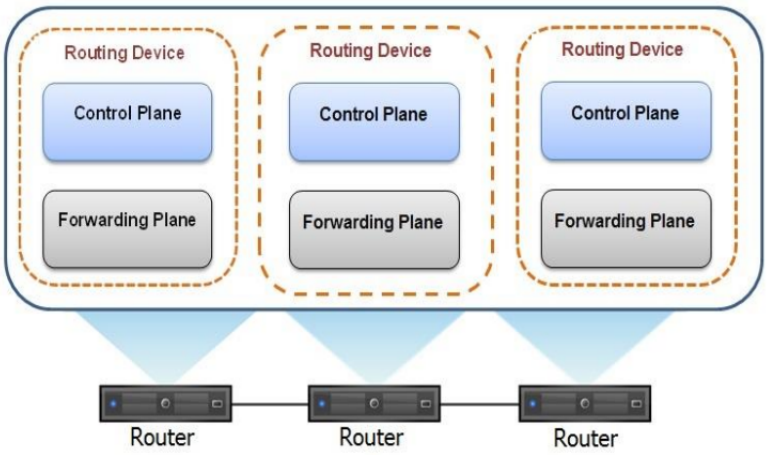


**Figure** 1.2 – *L'architectures des réseaux conventionnels*

### 1.1.4 SDN vs réseaux conventionnels

Nous pouvons distinguer les différences dans les points suivants :[M.T]

- L'unité de commande centralisée :
  - A une vue de bout-en-bout de l'ensemble du réseau.
  - Connaît les chemins et les capacités.
  - Peut calculer des chemins en fonction des adresses sources et de destinations.
  - Utilise des chemins d'accès réseau pour différents types de trafic.
  - Réagit rapidement à l'évolution des conditions et des contraintes.
- Séparation entre le plan de contrôle et le plan de données (la figure 1.3).
- Modèle réseau basé sur le contrôle.
- Système de calcul distribué – Système de calcul centralisé.

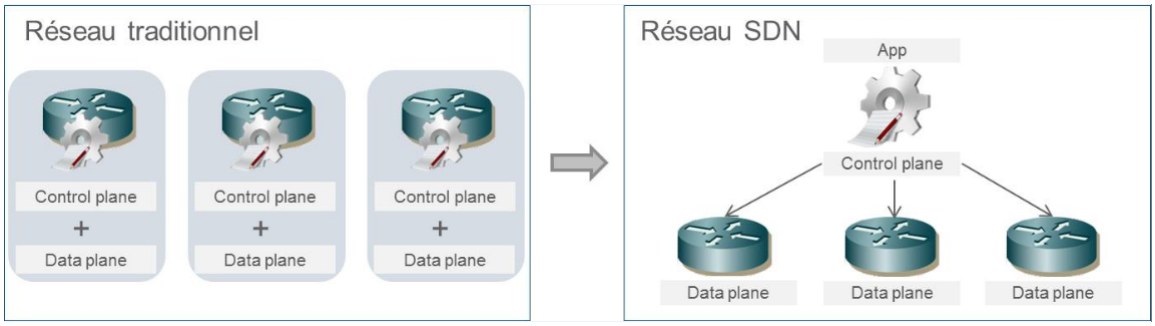


**Figure** 1.3 – *Réseaux conventionnels vs Réseaux SDN [TMX+15]*

### 1.1.5 Les avantages des SDNs

Le concept SDN va changer la façon dont les ingénieurs réseaux construisent et exploitent leurs réseaux pour atteindre les exigences de l'entreprise. Avec le SDN, les réseaux sont devenus des standards ouverts, non-propriétaires, et facile à programmer et à gérer.

Le SDN donne aux entreprises et aux opérateurs plus de contrôles sur leurs réseaux, et leur permettent d'adapter et d'optimiser leurs réseaux pour réduire l'ensemble du coût de maintien du réseau.

Les principaux avantages de SDN peuvent être résumées ci-dessous :

- **La simplicité de gestion du réseau :**

  avec SDN, le réseau peut être considéré et géré comme un seul noeud qui va transférer les tâches de gestion de réseau, par défaut compliqués, pour être abstraite dans un lieu où il sera plus facile de gérer les interfaces.

- **Le déploiement du service rapide :**

nouvelles fonctionnalités et applications peuvent être déployées de manière rapide en quelques heures au lieu de plusieurs jours.

- **La configuration automatique :**

  les tâches de configuration manuelles telle que l'attribution de VLAN et la configuration QoS « Qualité of Service » peuvent être provisionnées automatiquement.

- **La virtualisation de réseau :**

  Comme les serveurs et la virtualisation du stockage sont devenus plus déployé qu'avant, les réseaux peuvent eux aussi bénéficier a être virtualisé ainsi grâce au SDN.

- **La réduction de la dépense de fonctionnement :**

  en tirant profit de l'automatisation du déploiement du réseau, un changement sur le réseau n'a jamais été facile. Par conséquent,cela permet de réduire le coût de fonctionnement du réseau.

  – La programmabilité permet de répondre aux besoins d'automatisation du cloud.

  – Il est possible d'introduire une nouvelle fonctionnalité très facile, il n'est pas nécessaire d'implémenter sur des Switchs, il suffit de l'implémenter sur le contrôleur.

## 1.2 Architecture d'un réseau SDN

### 1.2.1 Les couches d'un réseau SDN

L'architecture de l'SDN est structurée en trois couches, comme la figure 1.4 présente :

- Les ressources réseau constituées par l'ensemble des éléments de réseaux physiques ou virtuels (routeurs et commutateurs).
- Une couche de contrôle contenant des fonctions de programmation des ressources réseau .
- Une couche de services applicatifs.

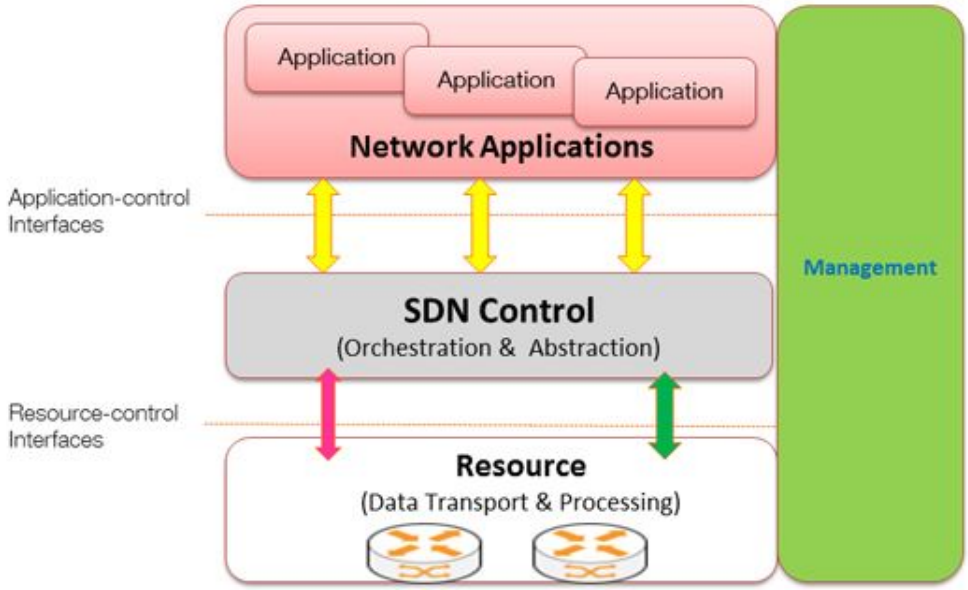


**Figure** 1.4 – *Les couches d'un réseau SDN [SVAS15]*

### 1.2.2 Les plans logiciel du réseau

Le réseau SDN a séparé le logiciel en quatre couches ou plans comme la figure 1.5 montre :

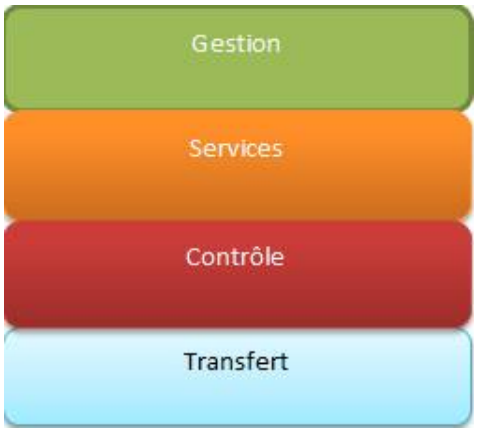


**Figure** 1.5 – *Les plans réseau [TMX+15]*

#### a. Le plan de Transfert

Le plan de transfert représente les muscles du réseau, son rôle est d'acheminer les paquets réseau et déplacer les données rapidement .

#### b. Le plan de Contrôle

Le plan de contrôle est le cerveau, son rôle est de Contrôler la topologie du réseau et de décider comment diriger le flux du trafic réseau.

**c. Le plan de Services**

Si le plan de transfert ne peut pas faire des traitements plus approfondies, dans ce cas le plan services effectue ce traitement approfondi et exécutant les opérations complexes sur les données réseau, ce plan est propice à l'innovation de réseau.

**d. Le plan de Gestion**

Si le plan de contrôle connaît toutes les informations d'un réseau le plan de gestion doit informer chaque périphérique qu'il doit faire donc le rôle principal de ce plan est indiquant comment un périphérique de réseau doit interagir avec le reste du réseau.

## 1.3 Modèle de programmation

Le réseau SDN ne peut pas être basée sur le modèle impératif parce que dans ce modèle un programme est obligé de connaître exactement toute la suite des actions à réaliser pour achever une action, c'est pour ça le SDN est basé sur le modèle déclaratif. Dans ce modèle un programme va faire tout le nécessaire pour arriver à l'objectif.

Le SDN est basé sur ce second modèle a cause de ces raisons[J.D15] :

- il est plus simple à mettre en œuvre puisqu'on va utiliser une intelligence déjà présente sur les équipements .
- il est plus robuste puisque les équipements réseau eux-mêmes peuvent réagir.
- il est préféré par les constructeurs qui acceptent plus facilement d'exposer l'intelligence de leurs équipements .

## 1.4 Différents modèles pour le SDN

Selon le type d'application, l'interaction avec le SDN se change selon le modèle de programmation, donc on peut noter ses différents modèles de programmabilité : [J.D15]

- **Programmabilité individuelle de chaque équipement**

  Dans ce modèle une application interagit directement avec chaque équipement via des API "Application Programming Interface".L'application est centralisée ou peut être localisée directement sur l'équipement réseau pour réaliser des tâches spécifiques.

- **Programmabilité via un contrôleur**

  Dans ce modèle, une application donne un ordre abstrait et global à un contrôleur, qui à son tour traduit cette requête en une suite d'ordre auprès des équipements du réseau concerné.

- **Création d'un réseau virtuel au-dessus du réseau physique**

  Dans ce modèle, les applications créent leur propre réseau « overlay », le réseau d'overlay assure l'intégralité des services. On parle également de virtualisation des fonctions réseau (NFV – Network Function Virtualisation) quand les routeurs, commutateurs, etc. sont des éléments virtualisés sur des serveurs.(la figure 1.6)

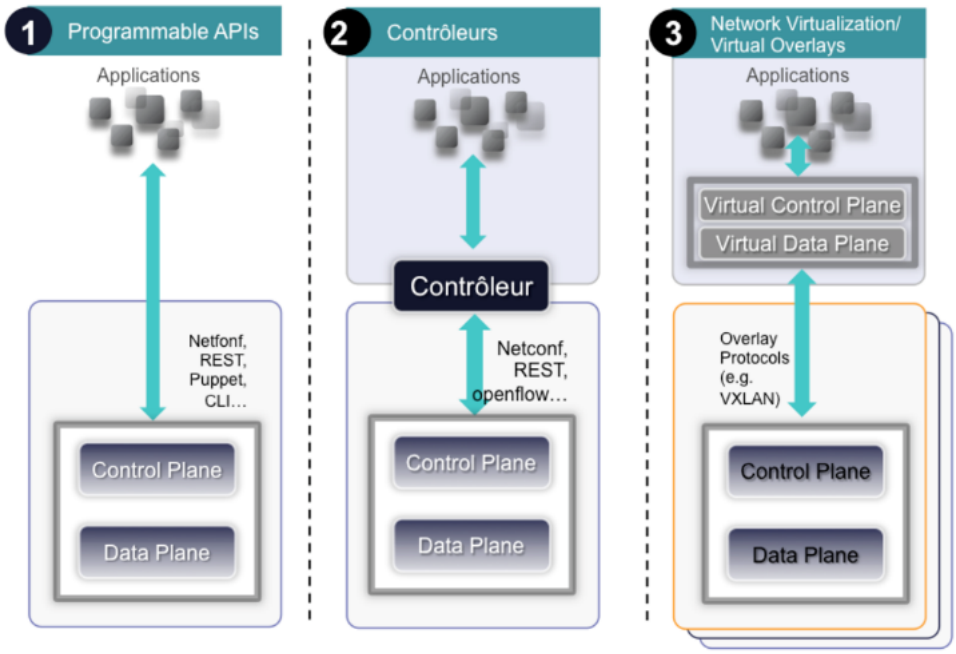


**Figure** 1.6 – *Les différents modèles SDN [M.T]*

## 1.5 Conclusion

L'avenir des réseaux dépend de plus en plus sur le logiciel, qui va accélérer le rythme de l'innovation pour les réseaux comme les domaines de stockage et de calcul. Dans ce chapitre nous avons présenté une étude sur les réseaux SDN, l'architecture SDN et tous les concepts relié a ce nouveau type de réseau. Dans le chapitre suivant nous allons étudier le plat-forme qui permet d'implementer les réseaux SDN.

# Chapitre 2

# OpenFlow

---

## 2.1 Généralités sur OpenFlow

### 2.1.1 Introduction

OpenFlow est une technologie standard utilisée pour gérer le trafic entre les switchs Ethernets, les routeurs et les points d'accès sans fil.

Le protocole OpenFlow est la base d'un SDN standardisé et basé sur des switchs Ethernet avec une table de flux interne et une interface standardisée pour ajouter et supprimer des entrées de flux.

L'idée de base derrière OpenFlow est qu'on peut connecter plusieurs commutateurs ensemble pour créer un flux ensuite gérer l'infrastructure complète, établir les politiques et gérer le type de trafic.

Dans ce chapitre on va présenter d'abord le protocole OpenFlow, le Switch OpenFlow avec ses composants, les étapes de correspondance, ensuite le contrôleur OpenFlow.

### 2.1.2 Historique d'OpenFlow

Le développement d'OpenFlow a commencé en 2007 dans le cadre d'une collaboration entre les mondes de l'université et des affaires. Établie à l'origine par l'université de Stanford et l'université de Californie à Berkeley [HP16].

OpenFlow est géré par l'ONF (Open Networking Foundation) qui est une organisation consacrée à la promotion et l'adoption du SDN à travers des standards ouverts de développement.

Depuis sa sortie, plusieurs entreprises et projets open source comme le projet OpenDaylight soutiennent OpenFlow. Autres sociétés comme Cisco et Brocade offrent des contrôleurs OpenFlow .

En janvier 2013, le géant industriel japonais de l'informatique et de la télécommunication NEC a dévoilé un commutateur virtuel pour les Microsoft Windows.

### 2.1.3 Évolution d'OpenFlow

La version 1.1 du protocole OpenFlow a été libérée le 28 février 2011, en décembre 2011, le jury d'ONF a approuvé la version 1.2 et la publiait en février 2012. La version actuelle d'OpenFlow est 1.4.

En mai 2011 l'université de l'Indiana conjointement avec l'ONF a lancé un laboratoire d'interopérabilité du SDN pour tester comment les différents vendeurs du

SDN bien fonctionnent avec les produits d'OpenFlow.

En février 2012 Big Switch networks a libéré le projet floodlight (un contrôleur OpenFlow sous la licence d'Apache) et sa suite OpenFlow-based SDN a été annoncé en novembre de la même année, qui contient un contrôleur commercial et une commutation virtuelle.

En février 2012, HP a dit qu'il soutient la norme sur 16 de ses produits de commutation Ethernet.

En avril 2012, Urs Hölzle décrit comment le réseau interne de l'entreprise Google avait été complètement repensé au cours des deux années précédentes pour s'exécuter sous OpenFlow avec l'amélioration substantielle de l'efficacité.

## 2.2 Le Switch OpenFlow

La plupart des Switchs et routeurs Ethernet modernes contiennent des tables de flux (construites sur des mémoires adressables par contenu) qui fonctionne à haut débit pour implémenter des firewalls, NAT (Network Adresse Translation), QoS (Quality of Service) et pour collecter des statistiques.

Un ensemble intéressant de fonctions sont implémentées sur la plupart des routeurs et des Switchs. OpenFlow exploite cet ensemble de fonctions et fournit un protocole ouvert pour programmer les tables de flux sur différents Switchs et routeurs.

Les versions du protocole les plus implémentées couramment sur les Switchs actuels sont v1.0 et v1.3 [M.T], la version 1.4 peut être envisagée dans sa globalité comme une extension de la v1.0. Donc la description qui suit porte sur la version 1.4, sauf mention contraire.

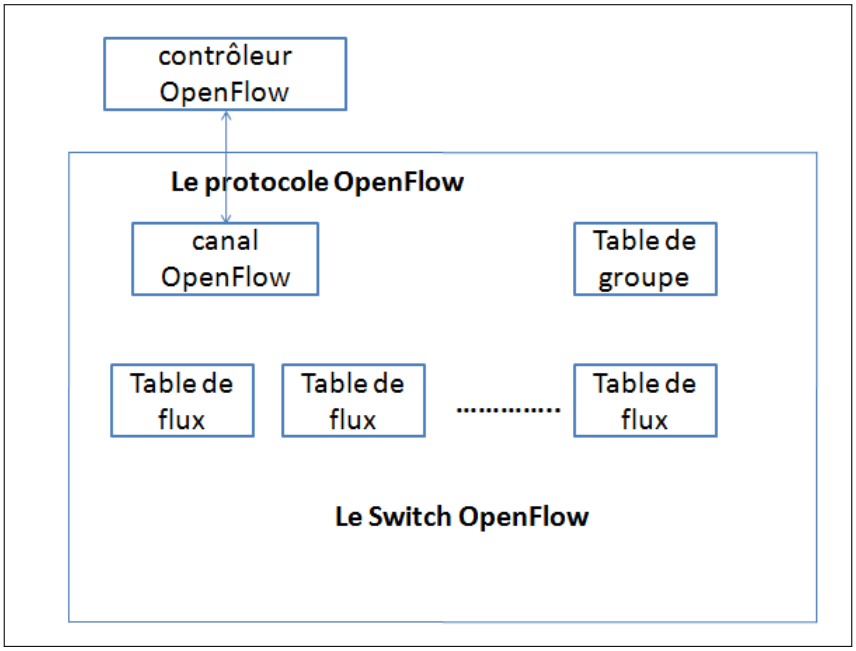


**Figure** 2.1 – *Les principaux composants d'un Switch OpenFlow [Fou13]*

### 2.2.1 Les composants d'un Switch OpenFlow

Le Switch OpenFlow se compose d'une ou plusieurs tables de flux et une table de groupe, qui sert à la recherche et le transfert des paquets et un canal OpenFlow voir la figure 2.1.

Le Switch communique avec le contrôleur et le contrôleur gère le Switch via le protocole OpenFlow. En utilisant le protocole OpenFlow, le contrôleur peut ajouter, mettre à jour et supprimer des entrées dans les tables de flux.

Chaque table de flux dans le Switch contient un ensemble des entrées de flux, chaque entrée de flux est constituée d'un champ de correspondance, des compteurs, et un ensemble d'instructions pour appliquer aux paquets correspondants.

La correspondance commence à la première table de flux et peut continuer aux tables de flux supplémentaires. Les entrées de flux font la correspondance des paquets par ordre de priorité.
Si une entrée correspondante est trouvée, les instructions associées à l'entrée de flux spécifique sont exécutées. Sinon, le résultat dépend de la configuration de l'entrée de flux de la table-miss. Par exemple, le paquet peut être transmis au contrôleur sur le canal OpenFlow, supprimé, ou peut continuer à la table de flux suivante.

Les instructions associées à chaque entrée de flux soit contenues des actions ou modifient le traitement de pipeline.

Les actions inclues dans les instructions décrivent le transfert et la modification de paquets. Les instructions de traitement de pipeline permettent aux paquets d'être envoyés aux tables suivantes pour plus de traitements et permettre à l'information,

sous forme de métadonnées, à communiquer entre les tables.

Le traitement de pipeline s'arrête lorsque le jeu d'instructions associé à une entrée de flux ne précise pas une table suivante , dans ce cas, le paquet est modifié et transmis.

Les entrées de flux peuvent transmettre à un port. Cela est généralement un port physique, mais il peut aussi être un port logique défini par le Switch ou d'un port réservé défini par le protocole OpenFlow.

### 2.2.2 Les ports d'OpenFlow

Les ports d'OpenFlow sont les interfaces réseau pour la transmission des paquets entre le traitement d'OpenFlow et le reste des réseaux. Les Switchs OpenFlow connectés logiquement à les autres via leurs ports d'OpenFlow.

L'ensemble des ports d'OpenFlow ne peut pas être identique à l'ensemble des interfaces de réseau fournies par le matériel de commutation, certaines interfaces de réseau peuvent être désactivées pour OpenFlow, et le Switch OpenFlow peut définir des ports supplémentaires [Fou13].

Les paquets d'OpenFlow sont reçus sur un port d'entrée et traités par le pipeline d'OpenFlow qui peut les transmettre à un port de sortie.

Un Switch OpenFlow doit prendre en charge trois types de ports d'OpenFlow :

ports physiques, ports logiques et ports réservés.

#### a. Les ports physiques

Les ports physiques d'OpenFlow sont des ports définis par le Switch qui correspond à l'interface matérielle de Switch.

Par exemple, sur un Switch Ethernet, les ports physiques mappent un à un aux interfaces Ethernet.

Dans certains déploiements, le Switch OpenFlow peut être virtualisé sur le matériel du Switch.

Dans ces cas, un port physique peut représenter une tranche virtuelle de l'interface matérielle de Switch.

**b. Les ports logiques**

Les ports logiques d'OpenFlow sont des ports définis par le Switch qui ne correspond pas directement à l'interface matérielle de Switch.

Les ports logiques peuvent inclure des paquets d'encapsulation et peuvent mapper aux divers ports physiques. Le traitement fait par le port logique doit être transparent par rapport au traitement d'OpenFlow et ces ports doivent interagir avec le traitement d'OpenFlow comme des ports physiques.

La seule différence entre les ports physiques et logiques, c'est qu'un paquet associé à un port logique peut avoir un champ de métadonnées supplémentaires appelé Tunnel-ID et quand un paquet reçu sur un port logique, ce paquet est envoyé au contrôleur.

**c. Les ports réservés**

Les ports réservés d'OpenFlow sont définis par le protocole OpenFlow. Ils ont spécifié des actions génériques de transmission telles que l'envoi au contrôleur ou l'inondation.

Un Switch OpenFlow est tenu de supporter les ports réservés qui sont marqués « obligatoire » ci-dessous [Fou13] :

- **Obligatoire :ALL :** il représente tous les ports que le Switch peut utiliser pour transmettre un paquet spécifique. Peut-être utilisé seulement comme un port de sortie. Dans ce cas, une copie du paquet est envoyée à tous les ports standard.
- **Obligatoire :CONTROLLER :** il représente le canal de contrôle avec le contrôleur OpenFlow. Peut-être utilisé comme un port d'entrée ou comme un port de sortie. Lorsqu'il est utilisé comme un port de sortie, encapsuler le paquet dans un message PACKET-IN et l'envoyer à l'aide de protocole OpenFlow. Lorsqu'il est utilisé comme un port d'entrée, cela identifie un paquet provenant du contrôleur.
- **Obligatoire : TABLE :** il représente le début de pipeline d'OpenFlow. Ce port est valable uniquement dans une action de sortie dans la liste des actions de message PACKET-OUT.
- **Obligatoire : IN PORT :** il représente le port d'entrée d'un paquet. Il peut être utilisé uniquement comme un port de sortie pour envoyer le paquet par son port d'entrée.
- **Obligatoire : ANY :** c'est une valeur spéciale utilisée dans certaines commandes OpenFlow lorsqu'aucun port n'est spécifié (c'est à dire le port est avec des caractères génériques). Il ne peut pas être utilisé comme un port d'entrée, ni comme un port de sortie.

### 2.2.3 Les tables d'OpenFlow

Cette section décrit les éléments de table de flux et la table-miss, ainsi que la mécanisme de correspondance [Fou09].
La figure 2.2 présente les flux de paquets au cours du traitement de pipeline

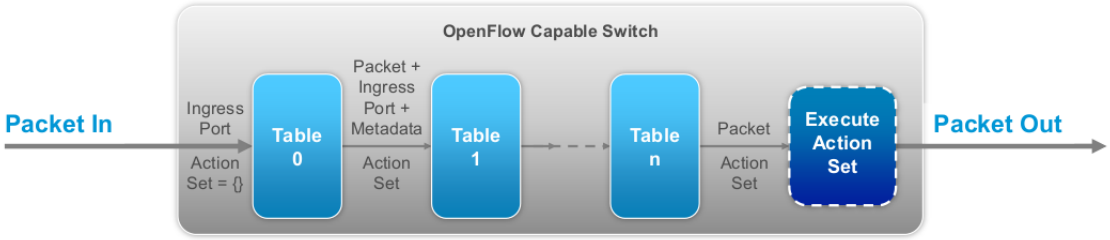


**Figure** 2.2 – *Les flux de paquets à travers le traitement de pipeline [Fou13]*

#### a. Le traitement de pipeline

Le traitement de pipeline OpenFlow définit comment les paquets interagissent avec ces tables de flux voir figure 2.2. Un Switch OpenFlow est requis d'avoir au moins une table de flux, et peut éventuellement avoir plusieurs tables de flux.

Un Switch OpenFlow avec seulement une table de flux est valide, dans ce cas le pipeline est très simplifié.

Les tables de flux d'un Switch OpenFlow sont numérotées séquentiellement, commençant à 0. Le traitement de pipeline commence toujours à la première table de flux : le paquet est d'abord comparé aux entrées de flux du 1ére table de flux numéro 0.

Les autres tables de flux peuvent être utilisées selon le résultat de la correspondance dans la première table.

Lors du traitement par une table de flux, le paquet est mis en correspondance avec les entrées du table de flux pour sélectionner une entrée de flux.

Si une entrée de flux est trouvée, le jeu d'instructions inclus dans cette entrée de flux est exécuté. Ces instructions peuvent diriger explicitement le paquet à une autre table de flux (en utilisant l'instruction Goto), où le même processus se répète à nouveau. Une entrée de flux peut diriger un paquet à une table de flux d'un numéro qui est supérieur à son propre numéro de table de flux.

Évidemment, les entrées du flux de la dernière table du pipeline ne peuvent pas inclure l'instruction Goto. Si la correspondance de l'entrée de flux ne dirige pas les paquets vers une autre table de flux, le pipeline s'arrête à cette table. Lorsque le pipeline s'arrête, le paquet est traité avec l'action associée définie et généralement

transmis .

Le pipeline d'OpenFlow et les diverses opérations d'OpenFlow traitent les paquets d'un type spécifique avec les spécifications définies pour ce type de paquet.

### b. La table de flux

Une table de flux est constituée par des entrées de flux[Fou09].

| Match Fields | Priority | Counters | Instructions | Timeouts | Cookie |
|---|---|---|---|---|---|

TABLE 2.1 – *Les Principaux éléments d'une entrée dans une table de flux [Fou13]*

Chaque entrée de la table de flux voir la table 2.1 contient [SA14] :

- **Match Field :** il fait la correspondance sur les paquets. Il est composé par des ports d'entré et des entêtes des paquets et éventuellement les métadonnées spécifiées par une table précédente.
- **Priority :** il est correspond à la priorité de l'entrée de flux.
- **Counters :** ils font la mise à jour lorsque les paquets sont mis en correspondance.
- **Instructions :** ils modifient les actions ou le traitement de pipeline.
- **Timeouts :** c'est un montant maximal de temps ou le temps d'inactivité avant que le flux est expiré par le Switch.
- **Cookie :** c'est une valeur de données opaque choisie par le contrôleur. Elle peut être utilisée par le contrôleur pour l'obtention des statistiques de flux, la modification et la suppression de flux.

### c. La table-miss

Chaque table de flux doit prendre en charge une entrée de flux de la table-miss pour traiter le manque de cette table. L'entrée de flux de la table-miss spécifie comment traiter les paquets inégalés par d'autres entrées de flux dans la table de flux, et peut par exemple envoyer des paquets au contrôleur ou supprimer des paquets.

### d. La correspondance

La figure 2.3 est un organigramme qui montre les étapes de correspondance :

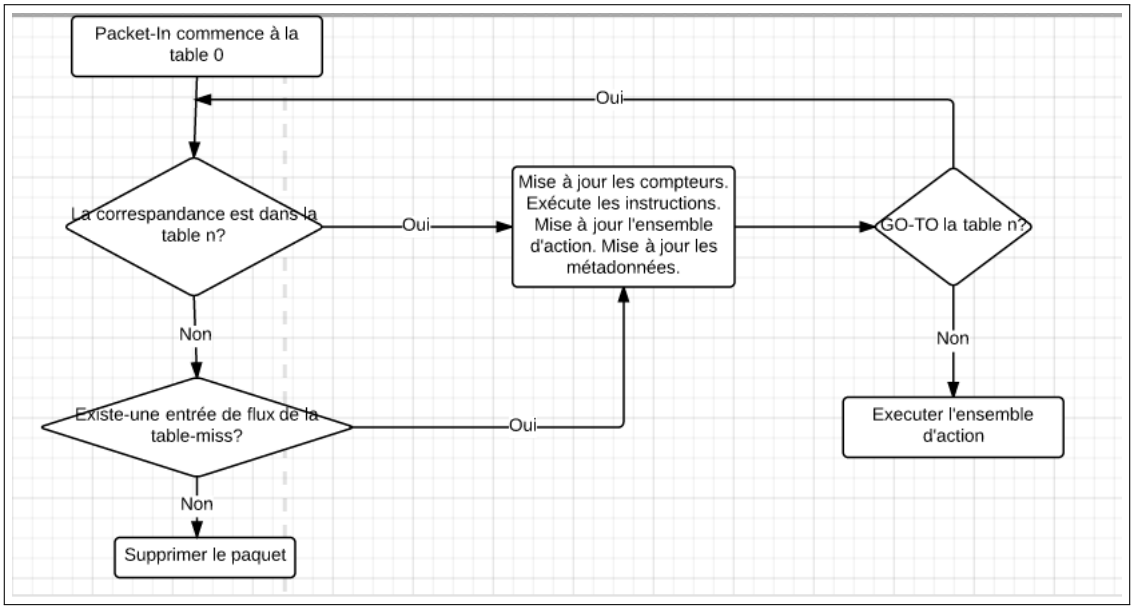


**Figure** 2.3 – *Organigramme détaillant les flux des paquets via un Switch OpenFlow [Fou13]*

À la réception d'un paquet, le Switch OpenFlow exécute les fonctions indiquées à la figure 2.3. Le Switch commence en effectuant une recherche de table dans la première table de flux, et en se basant sur le pipeline, il peut effectuer une recherche de table dans les autres tables de flux [SA14].

Les champs de correspondance utilisés pour la recherche de table dépendent de type de paquet, et incluent différents champs d'en-tête de paquet, comme l'adresse de source ou l'adresse de destination.

En plus des en-têtes de paquets, les correspondances peuvent également être effectuées sur le port d'entrée et les champs de métadonnées.

Les métadonnées peuvent être utilisées pour transmettre des informations entre les tables dans un Switch.

Un paquet correspond à une entrée de table de flux si les valeurs dans les champs de correspondance de paquets utilisés pour la recherche sont définies dans l'entrée de table de flux.

Le paquet est comparé à la table et seulement l'entrée de flux avec une priorité plus élevée qui correspond au paquet doit être sélectionnée. Les compteurs associés à l'entrée de flux sélectionné doivent être mise à jour et le jeu d'instructions inclus dans l'entrée de flux sélectionné doit être appliqué.

S'il y a plusieurs flux d'entrées correspondant avec la même priorité, l'entrée de flux sélectionné n'est pas explicitement définie.

**e. L'ensemble d'action**

Un ensemble d'action est associé à chaque paquet. Cet ensemble est vide par défaut, une entrée de flux peut modifier l'action définie à l'aide d'une instruction Write-action ou une instruction Clear-action .

L'ensemble d'action est effectué sur les tables de flux lorsque l'ensemble d'une entrée de flux d'instructions ne pas contenir une instruction Goto-table, le traitement de pipeline s'arrête et les actions dans l'ensemble de l'action sont exécutées.

### 2.2.4 Implémentaion des Switch OpenFlow

La table 2.2 présente les implémentations des Switchs OpenFlow et les versions compatibles ainsi que la table 2.3 présente les fabricants de ces Switchs[BMX+14] :

| **Switch logiciel** | **Implementation** | **Version** |
|---|---|---|
| Open vSwitch | C/Python | v1.0 |
| Pantou/OpenWRT | C | v1.0 |
| Ofsoftswitch13 | C/C++ | v1.3 |
| Indigo | C | v1.0 |

TABLE 2.2 – *Implémentation des Switchs logiciels compatibles avec OpenFlow*

| **Fabricant** | **Le modele de Switch** | **Version** |
|---|---|---|
| Hewlett-Packard | 8200zl, 6600, 6200zl,5400zl, et 3500/3500yl | v1.0 |
| Brocade | NetIron CES 2000 Series | v1.0 |
| IBM | RackSwitch G8264 | v1.0 |
| NEC | PF5240 PF5820 | v1.0 |
| Pronto | 3290 et 3780 | v1.0 |
| Juniper | Junos MX-Series | v1.0 |
| Pica8 | P-3290, P-3295, P-3780 et P-3920 | v1.3 |

TABLE 2.3 – *Les principaux Switchs OpenFlow et leur fabricants*

## 2.3 Le contrôleur OpenFlow

Un contrôleur ajoute ou supprime les entrées de flux dans la table de flux pour le compte d'expériences. Par exemple, un contrôleur statique pourrait être une simple application exécutée sur un PC pour établir de façon statique des flux pour interconnecter un jeu d'instructions de tests pour la durée de l'expérience [FC16]. Dans ce cas, les flux ressemblent à ceux de VLAN des réseaux actuels. De la même

manière, OpenFlow est une généralisation des VLAN.

On peut aussi imaginer des contrôleurs plus sophistiqués qui peuvent dynamiquement ajouter ou supprimer des flux au cours de l'expérience. Dans un modèle d'utilisation, un chercheur pourrait contrôler le réseau complet de Switchs OpenFlow et être libre de décider comment les flux sont traités.

Un contrôleur plus sophistiqué pourrait supporter de multiples chercheurs, chacun avec différents comptes et permissions, leur permettant d'exécuter de multiples expériences indépendantes sur des jeux différents de flux. [FC16]

### 2.3.1 Les principaux contrôleurs OpenFlow

**a. Beacon**

C'est un contrôleur OpenFlow rapide, multi-plateforme, modulaire qui est basé sur Java [A.F16] .

**b. Floodlight**

C'est un contrôleur supporté par BigSwitch, Il est sous licence Apache et écrit en Java [A.F16].

**c. NOX et POX**

Ils sont les deux premiers contrôleurs OpenFlow. NOX est implémenté en C++, ainsi que POX implémenté en Python [A.F16] .

**d. Ryu**

C'est un contrôleur SDN capable de configurer les équipements réseaux en utilisant OpenFlow, NetConf et OF-config[A.F16].

la table 2.4 présente les principaux contrôleurs et leurs implémentations :

| Contrôleur | Implémentation | Open Source | Dévelopeur |
|---|---|---|---|
| POX | Python | Oui | Nicira |
| NOX | Python/C++ | Oui | Nicira |
| MUL | C | Oui | Kulcloud |
| Maestro | Java | Oui | Rice University |
| Beacon | Java | Oui | Stanford |
| Floodlight | Java | Oui | BigSwitch |
| SNAC | C++ | Non | Nicira |
| Ryu | Python | Oui | NTT, OSRG group |

TABLE 2.4 – *Implémentation des contrôleurs OpenFlow [BMX+14]*

## 2.4 Le canal OpenFlow

Le canal OpenFlow est l'interface qui relie chaque Switch OpenFlow à un contrôleur. Via cette interface, le contrôleur configure et gère le Switch, et envoie des paquets au Switch [Fou13].

### 2.4.1 Les différents messages d'OpenFlow

Le protocole OpenFlow prend en charge trois types de messages, contrôleur-switch, asynchrone et symétrique [Fou13].

Chaque type avec plusieurs sous-types. Les messages contrôleur-switch sont initiés par le contrôleur et utilisés pour gérer l'état du Switch.

Les messages asynchrones sont initiés par le Switch et utilisés pour mettre à jour des informations sur l'état de réseau.

Les messages symétriques sont initiés soit par le Switch ou le contrôleur et envoyés sans sollicitation.

Les types de messages utilisés par OpenFlow sont décrits ci-dessous :

**a. Les messages contrôleur-switch**

- **Features :** ces messages sont en mode (request/reply), le contrôleur envoie une demande des caractéristiques au Switch. Ce dernier envoie une réponse des caractéristiques qui spécifie les capacités prises en charge par le Switch.

- **Configuration :** ce message permet au contrôleur de définir et d'envoyer des requêtes de configuration au Switch. Le Switch répond à la requête et envoie les informations nécessaires au contrôleur.

- **Modify-State :** il est utilisé pour ajouter, supprimer ou modifier les flux dans les tables de flux et pour définir les propriétés du port d'un Switch.
- **Read-State :** ces messages permettent au contrôleur de recueillir des statistiques sur les tables de flux, les ports et les entrées de flux .
- **Packet-out :** ils sont utilisés par le contrôleur pour envoyer des paquets à partir d'un port spécifié sur le Switch, et pour transmettre les paquets reçus via des messages Packet-in.
- **Send-Packet :** le contrôleur utilise ces messages pour envoyer des paquets sur un port spécifié dans le Switch. Soit parce qu'il ne dispose pas d'une correspondance pour toute entrée dans la table de flux ou il correspond à une action qui commande le Switch pour l'envoyer au contrôleur.

**b. Les messages asynchrones**

- **Packet-in :** un message Packet-in contient un paquet qui doit être envoyé au contrôleur.
- **Port-status :** ce message est envoyé au contrôleur si un port de Switch change son état.
- **Error :** si une erreur qui s'est passé au niveau de Switch, le contrôleur pourrait être avisé par ce type de message.

**c. Les messages symmétriques**

- **Hello :** ces messages sont échangés entre le Switch et le contrôleur lorsque la connexion est démarée.
- **Echo :** ces messages sont en mode (request/reply) où ils peuvent être envoyés soit par le Switch ou le contrôleur et doit retourner un message (echo-reply) pour indiquer la latence et la bande passante de la connexion.
- **Vendor :** ces messages permettent à certains vendeurs de créer un message personnalisé qui permet aux Switchs OpenFlow d'offrir des fonctionnalités supplémentaires.

### 2.4.2 Les étapes de connexion d'un Switch OpenFlow au contrôleur

**a. Le premier scénario**

Tout d'abord, il faut renseigner sur l'adresse IP du contrôleur au niveau du Switch. Lors du démarrage du Switch, le Switch OpenFlow envoie un paquet OFPT-HELLO avec le numéro de version d'OpenFlow supporté. Le contrôleur vérifie la version d'OpenFlow supporté par le Switch et lui répond par un OFPT-HELLO en indiquant la version d'OpenFlow. La connexion est ainsi établie voir 2.4.

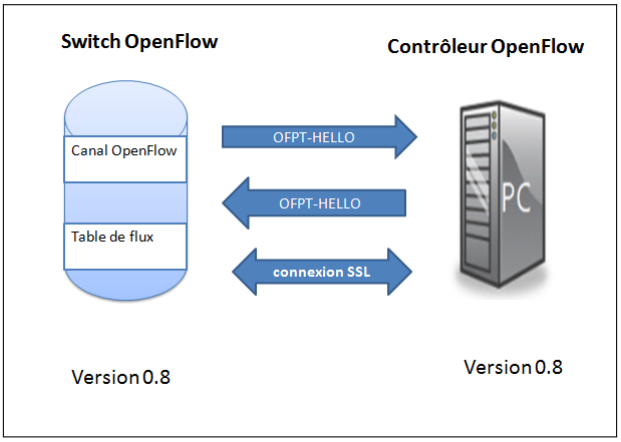


**Figure** 2.4 – *Connexion au contrôleur[FC16]*

### b. Le deuxième scénario

Comme dans le scénario précédant le Switch OpenFlow envoie son paquet OFPT-HELLO avec la version, le contrôleur s'aperçoit qu'il ne supporte pas la version OpenFlow du Switch. Il lui retourne donc un paquet OFPT-ERROR en indiquant que c'est un problème de compatibilité voir 2.5.

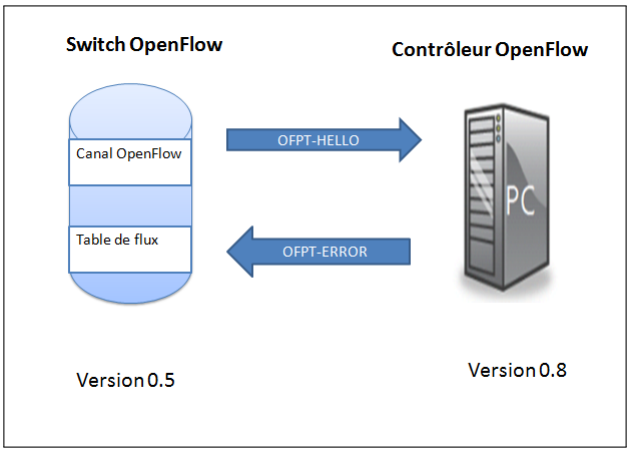


**Figure** 2.5 – *L'échec de connexion [FC16]*

### c. Le troisième scénario

Comme dans les scénarios précédents le Switch envoie un paquet OFPT-HELLO à son contrôleur, si celui-ci ne répond pas il tente alors de joindre les autres contrôleurs qui lui ont été paramétrés. S'il ne parvient pas à les joindre, il se met alors en mode "EMERGENCY MODE". Le Switch utilise sa table de flux par défaut, si toutefois un paquet ne correspond à aucun enregistrement dans la table il le supprime.

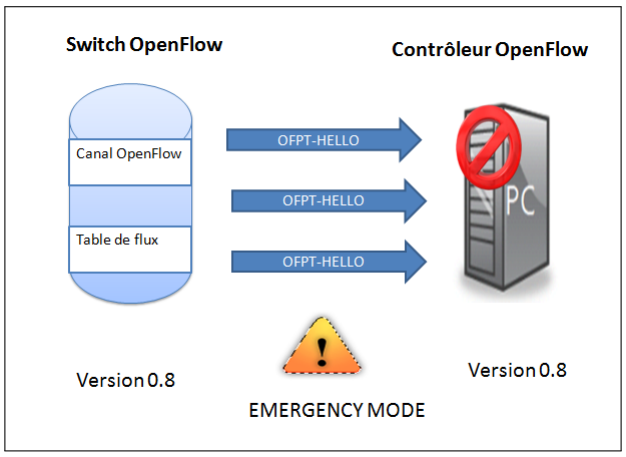


**Figure** 2.6 – *Contrôleur injoignable[FC16]*

## 2.5 Faiblesses du protocole OpenFlow

### 2.5.1 Perméabilité de la liaison contrôleur-switch

Du point de vue de la sécurité, la centralisation du contrôle est un point critique du réseau, le contrôleur doit être l'unique source des consignes OpenFlow envoyées aux Switchs. la sécurité de l'installation repose alors sur l'authentification des communications.

Cependant, la prise en charge de TLS (Transport Layer Security) qui est exigée pour OpenFlow 1.0 a été rendue optionnelle à partir d'OpenFlow 1.1[M.T].
Jusqu'à mi-2014, très peu de Switchs et aucun contrôleur libre n'assuraient ce service. En l'absence de mécanisme d'authentification, la centralisation au niveau du contrôleur permet à tout attaquant ayant gagné accès au réseau d'administration d'agir à tout moment sur le paramétrage du Switch et de configurer le réseau de production comme souhaité.

Afin de filtrer les commandes d'un administrateur légitime, il est possible pour un attaquant de se placer en man-in-the-middle, aussi l'écoute de messages OpenFlow non chiffrés qui lui permet d'obtenir de nombreux renseignements sur les réseaux .

### 2.5.2 Risques de déni de service côté Switch

Il est possible de provoquer un DoS sur un Switch OpenFlow en le soumettant à un trafic soutenu. Un Switch supprime une partie de l'ensemble des paquets qui lui parviennent s'il est incapable de les traiter .

Une approche propre au protocole vise à surcharger la mémoire-tampon du Switch (s'il en dispose) avec des paquets en attente d'être envoyés au contrôleur. La mémoire pourrait saturer pour deux raisons principales : la réception intensive de paquets à envoyer au contrôleur, et l'absence de réponse du contrôleur aux

PACKET-IN[M.T].

### 2.5.3 Risques de déni de service côté contrôleur

Un risque important se présente par rapport au système de maîtres-esclaves. Sans mécanisme d'authentification, il est simple pour un contrôleur adverse d'usurper le rôle de maître, ou du moins d'empêcher qu'un contrôleur légitime qui souhaite être maître puisse paramétrer le Switch.

En effet, quand bien même ce contrôleur légitime effectuerait une requête de rôle à chaque fois qu'il serait averti de son passage au statut d'esclave, il suffirait à l'adversaire d'adopter exactement le même comportement pour empêcher le contrôleur légitime d'effectuer des actions de configuration[M.T].

## 2.6 Conclusion

Dans ce chapitre, nous avons présenté le protocole OpenFlow et on détaille le contrôleur et le Switch ainsi que le mécanisme de correspondance.

Nous estimons que OpenFlow est un compromis pragmatique qui permet aux chercheurs d'exécuter des expériences sur des Switchs et des routeurs hétérogènes de manière uniforme, sans la nécessité des fournisseurs pour exposer le fonctionnement interne de leurs produits, ou pour les chercheurs d'écrire des logiciels de contrôle spécifique à chaque vendeur.

Dans le chapitre suivant nous allons étudier les performances du protocole OpenFlow dans l'environemment Mininet .

# Chapitre 3

# Simulation et Analyse

---

## 3.1 Introduction

Afin de comprendre le rôle d'OpenFlow et de ses éléments constitutifs, il est important de faire une étude sur les contrôleurs OpenFlow, ses performances et les différents messages échangés.

Ce chapitre est composé, principalement de deux parties :
la première partie décrit l'architecture de Mininet et son fonctionnement, la deuxiéme partie décrit les messages OpenFlow et leurs structures. Elle présente aussi les deux contrôleur NOX et POX et leur rôle dans la gestion de ces messages. Ensuite, nous présentons les résultats de simulation (Analyse, Discussion).

## 3.2 Outils de l'émulation

### 3.2.1 Mininet

Mininet [Tea16] est un émulateur de réseau qui permet de créer des hosts, switch, contrôleurs et liens virtuels, il fournit la capacité de les créer via :

- Ligne de commande.
- Interface utilisateur interactive.
- Application Python.

Mininet prend en charge la recherche, le développement, le prototypage, les tests, le débogage, et toutes autres tâches qui pourraient bénéficier d'avoir un réseau expérimental complet sur un ordinateur.

#### a. Configuration

- Un ordinateur portable (HP 630,Intel(R) Core(TM) i3 CPU M 380 @ 2.53 GHz , cores = 2, threads = 4)
- Mininet 2.2.1 on Ubuntu 14.04, 32 bits
- Virtualbox 4.3.22

#### b. Architecture de Mininet

Mininet est une plateforme qui permet de virtualiser un réseau sur un seul noyau, elle émule les switchs et les hosts avec des simples processus. La figure suivante illustre un cas simple où les deux hosts h1 et h2 communiquent entre eux grâce aux câbles ethernet virtuels (Veth Pair), h1 et h2 sont connectés à un seul (virtuel) switch S1 voir 3.1.

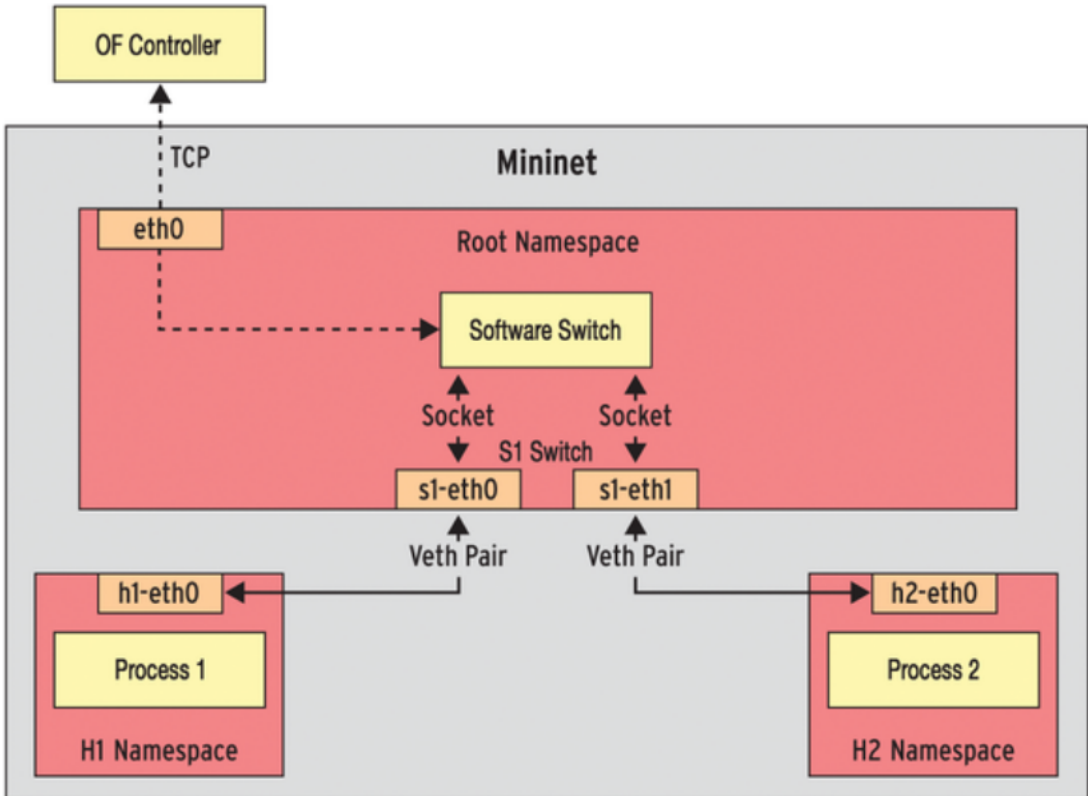


**Figure** 3.1 – *L'architecture de Mininet [O.I15]*

### c. Architecture de Mininet dans VirtualBox

La figure 3.2 montre comment le PC communique avec la VM Mininet via le protocole SSH.

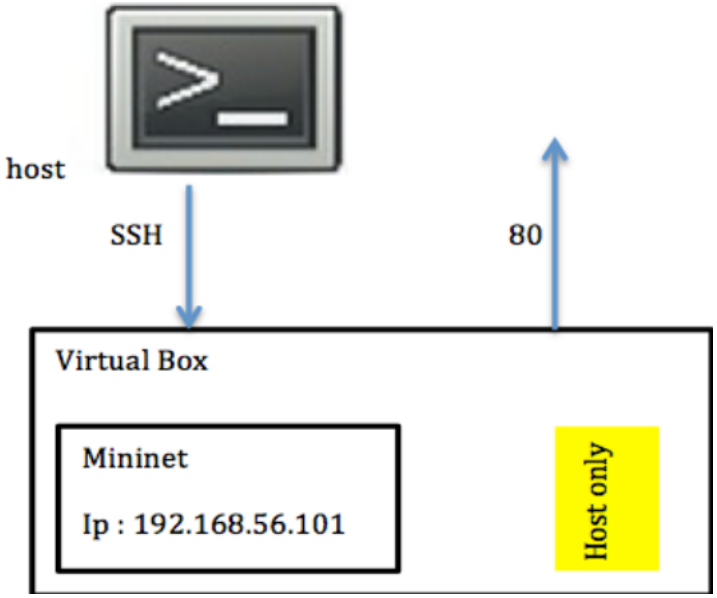


**Figure** 3.2 – *L'architecture de Mininet dans VirtualBox [O.I15]*

## 3.3 Démonstration des messages échangés dans le réseau OpenFlow

Afin de démontrer les messages expliqués dans le chapitre précédent, l'émulateur de réseau Mininet permettant d'émuler les deux hosts connectés à un switch et un contrôleur, voir figure 3.3 ci-dessous.

Pour cette démonstration, nous expliquons tout d'abord l'établissement de connexion switch-contrôleur, puis la communication Host-Host à travers le switch et le contrôleur.

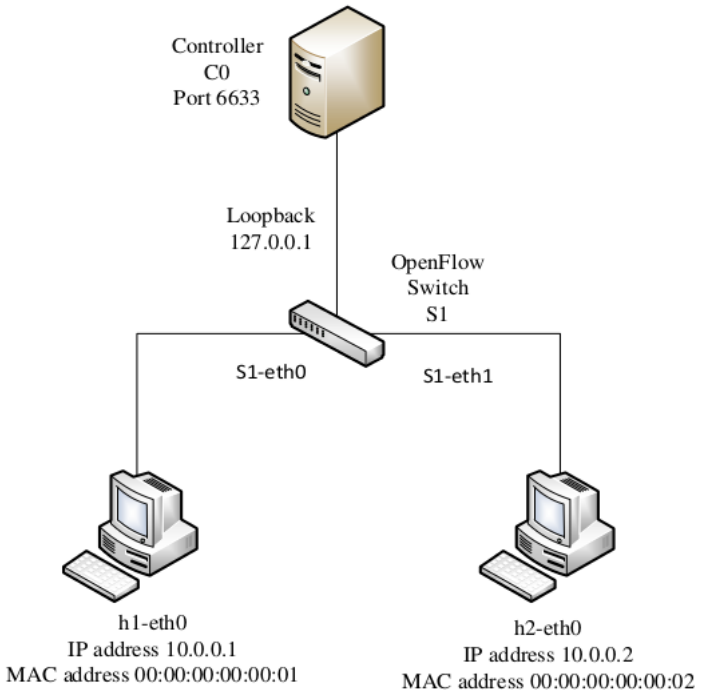


**Figure** 3.3 – *La topologie de réseau en utilisant Mininet [SA14]*

### 3.3.1 L'échange de messages entre un switch et un contrôleur

Lorsqu'un switch se connecte à un réseau OpenFlow, il établit une connexion TCP avecl'adresse IP du contrôleur (127.0.0.1) et un port par défaut de 6633.

Les deux côtés commencent à échanger des messages **Hello** qui incluent le plus grand nombre de version OpenFlow pris en charge. Après cela, un message **Feature request** est envoyé par le contrôleur pour savoir quels ports sont disponibles dans le switch, qui à son tour répond avec un message **Feature reply** qui contient une liste des ports, les tables et les actions.

Un message **Set config** est envoyé ensuite par le contrôleur pour demander au switch d'envoyer des expirations de flux.

Finalement, des messages **echo request**, **echo reply** sont envoyés fréquemment entre le switch et le contrôleur pour échanger des informations relatives à la bande

passante, latence et temporisation de leur connexion.

La figure 3.4 illustre les messages échangés entre un switch et un contrôleur ainsi que la figure 3.5 présente une capture des paquets avec Wireshark [G.C16].

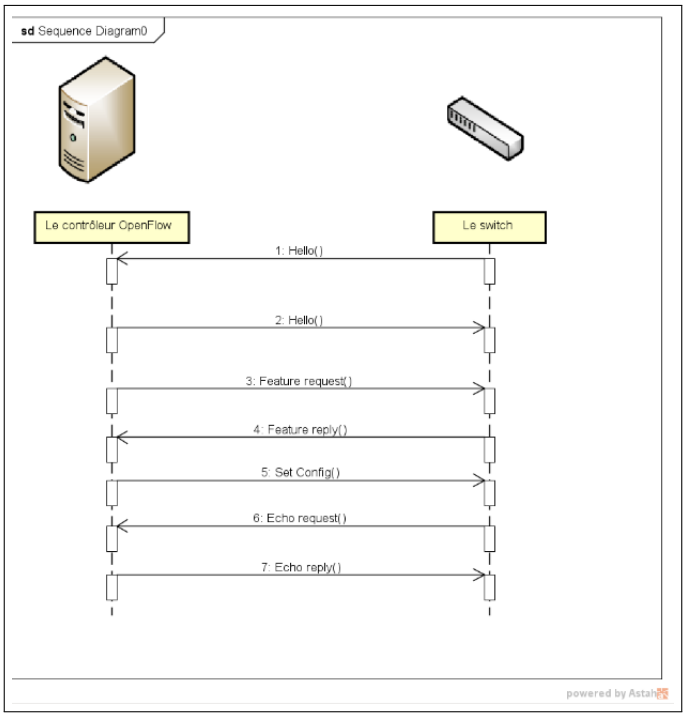


**Figure** 3.4 – *Les messages de communication entre le switch et le contrôleur OpenFlow*

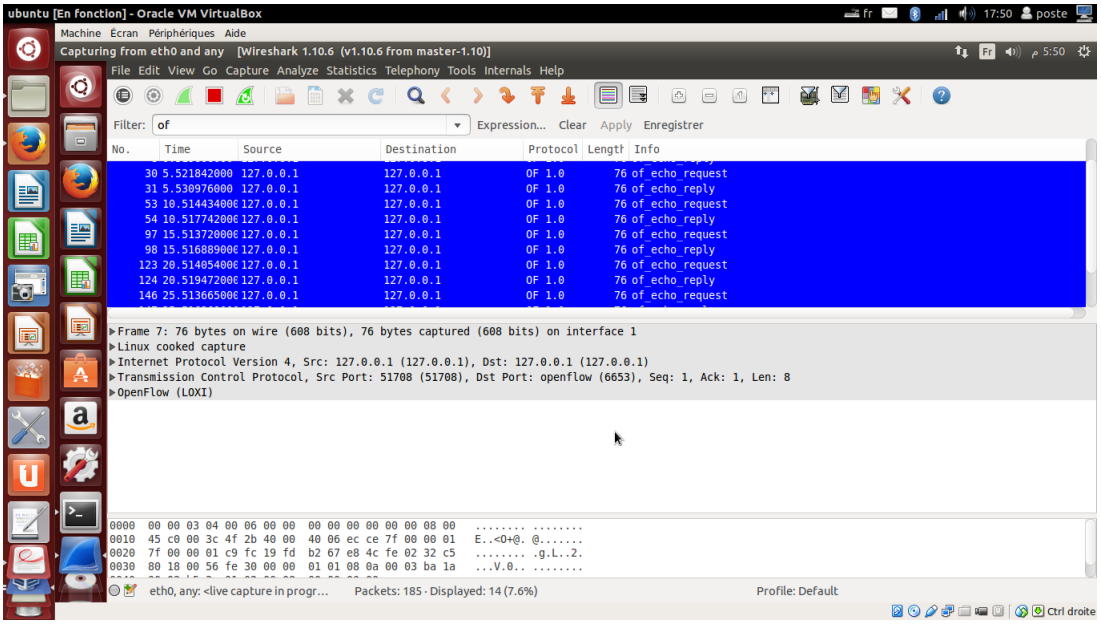


**Figure** 3.5 – *Capture de l'établissement de connexion entre le contrôleur et le switch avec Wireshark*

### 3.3.2 Messages échangés entre deux hosts

Pour démontrer comment la connexion host-host est effectuée dans un réseau OpenFlow, nous avons utilisé l'outil Ping pour envoyer des paquets ICMP de h1 à h2 et vice versa.

H1 envoie une requête ARP vers le switch, pour demander l'adresse MAC de h2, le switch ne sait pas comment traiter le paquet, alors il envoie le paquet comme un message de **PACKET-IN** au contrôleur.

Le contrôleur répond avec un message **PACKET-OUT** qui a une action à instruire le switch pour envoyer le paquet à tous les ports sauf le port entrant et attend une réponse pour cette demande. Lorsque h2 répond à cette demande, le switch envoie également cette réponse au contrôleur parce qu'il n'a aucune idée où il transfére le paquet.

Lorsque le contrôleur reçoit l'ARP reply, il envoie un message **FLOW-MOD** pour installer une nouvelle entrée de flux pour la prochaine réponse ARP de h2 pour être transmis directement par le switch sans aviser le contrôleur.

Le même processus se fait quand h1 envoie un ICMP request/reply , et quand h2 envoie une requête ARP pour la résolution de l'adresse MAC de h1.

À la fin, cinq nouvelles entrées de flux seront installées dans la table de flux de Switch par le contrôleur.

La figure 3.6 illustre l'échange de ces messages entre h1 et h2.

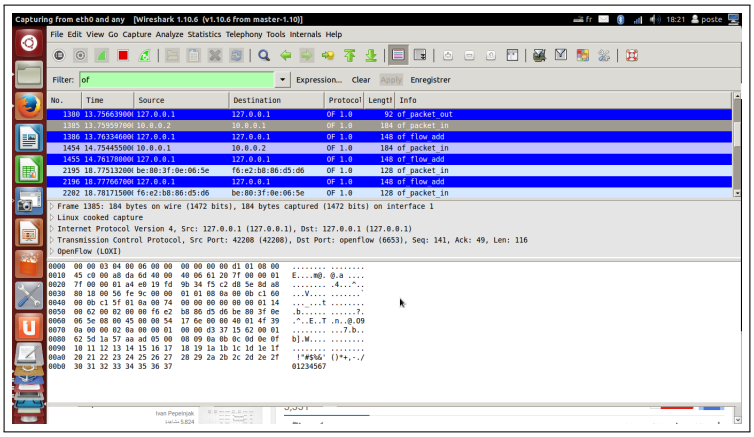

**Figure** 3.6 – *L'échange de messages entre h1 et h2*

## 3.4 L'évaluation des contrôleurs OpenFlow

### 3.4.1 Configuration de l'expérience

Pendant l'essai, nous avons lancer les deux outils d'analyse comparative (Cbench) et les contrôleurs, dans le test d'évaluation de contrôleur, nous avons

connecté des nombres différents de switchs virtuels et chaque switch a été connecté à 100k unique MAC (host).

Nous avons testé chaque contrôleur avec un nombre différent de switchs (1, 8, 16, 32, 64 et 128), chaque essai a été répété huit fois et la dernière a donné la valeur moyenne pour ces tests. La durée de chaque test est dix secondes.

### 3.4.2 Cbench (l'outil benchmark de contrôleur)

Selon l'ONF, Cbench [R.S16] peut être défini comme un "controller benchmarker" est un programme pour tester les contrôleurs OpenFlow. il y a deux modes possibles pour la configuration du test.

#### a. Configuration de test en mode throughput

En mode throughput, les switchs émulés envoient des messages PACKET-IN au contrôleur, en assurant que le contrôleur a toujours des messages à traiter [D.E13].

```
sara@sara-VirtualBox:~$ cbench -c localhost -p 6633 -m 10000 -l 4 -s 1 -M 100000
0 -t
cbench: controller benchmarking tool
   running in mode 'throughput'
   connecting to controller at localhost:6633
   faking 1 switches offset 1 :: 4 tests each; 10000 ms per test
   with 1000000 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
```

**Figure** 3.7 – *Le test en mode throughput*

#### b. Configuration de test en mode latency

Le test de latence utilise Cbench pour émuler des switchs qui envoie un seul paquet au contrôleur, attend une réponse, puis répéter ce processus . Le nombre total de réponses reçues par le contrôleur à la fin de la période de test peut être utilisé pour calculer le temps moyen [D.E13].

```
sara@sara-VirtualBox:~$ cbench -c localhost -p 6633 -m 10000 -l 4 -s 1 -M 1000000
cbench: controller benchmarking tool
   running in mode 'latency'
   connecting to controller at localhost:6633
   faking 1 switches offset 1 :: 4 tests each; 10000 ms per test
   with 1000000 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
```

**Figure** 3.8 – *Le test en mode latency*

### 3.4.3 Le contrôleur NOX

NOX [nox16] a été le premier contrôleur d'Open Flow, et il fut donné au milieu de recherche en 2008, il a été écrit en C++.

### 3.4.4 Le contrôleur POX

POX [E.C16] est un contrôleur open source en Python, le contrôleur pox permet un développement rapide et prototypage, devient plus communément utilisé que NOX.

### 3.4.5 Les paramètres de test

Les paramètres de l'émulation sont présentés dans la table 3.1 :

| | |
|---|---|
| Le contrôleur | localhost (par défaut) |
| Le port | 6633 (par défaut) |
| La durée de test | 10000 ms |
| Nombre de tests | 8 |
| Nombre de switchs | (1, 8, 16, 32, 64 et 128) |
| Nombre de hosts connectés | 1 M |

TABLE 3.1 – *Les paramètres de test*

### 3.4.6 Évaluation du NOX

La figure 3.9 représente les résultats de tests sous forme d'une courbe, qui représente la latence pour le contrôleur NOX en fonction de nombre de switchs.

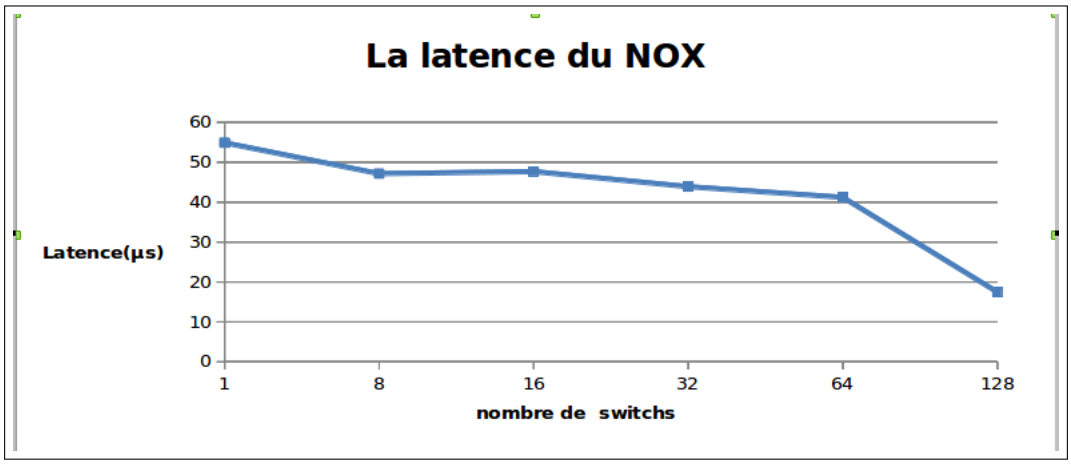


**Figure** 3.9 – *Le résultat de latence avec NOX*

Les résultats montrent que pour un seul switch connecté la valeur de latence est plus élevée (54 µ s) tel qu'illustré à la figure 3.9. Après ça pour le nombre de switch (8,16, 32 et 64) on remarque une stabilité de temps de latence ensuite pour 128 switchs une diminution du temps de latence qui atteint aussi bas que (17,46 µ s).

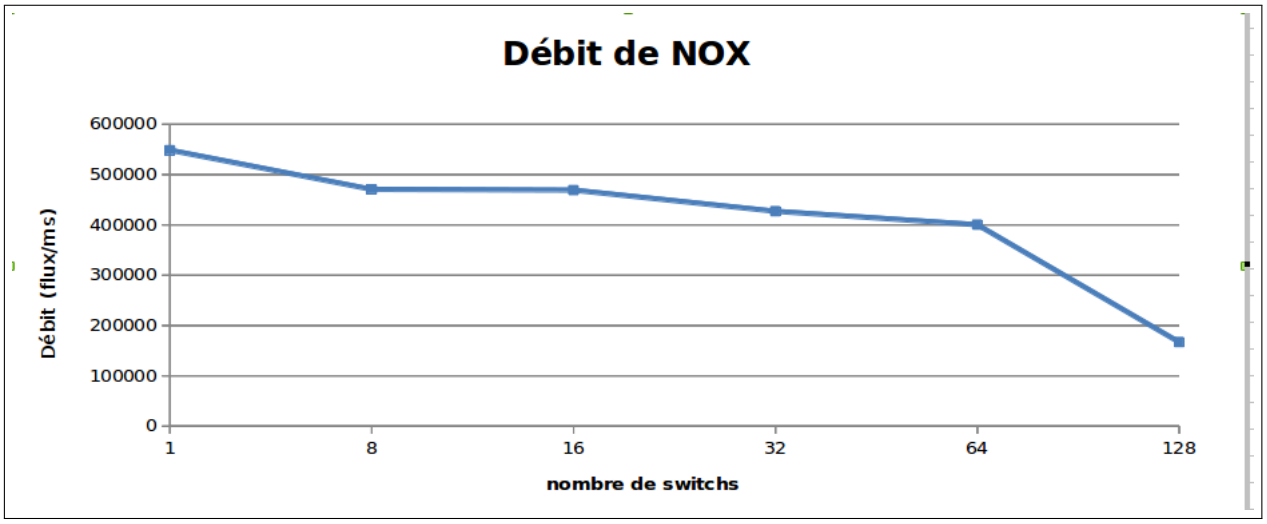


**Figure** 3.10 – *Le résultat de débit avec NOX*

La figure 3.10 représente les résultats du débit pour le contrôleur NOX en fonction de nombre de switchs. On remarque que le débit a atteint une valeur élevée quand un switch connecté avec une valeur de (550000 flux /ms), ensuite une diminution de débit est remarqué quand on connecte (8, 16, 32 et 64) switchs pour atteindre une valeur plus bas (150000 flux /ms) pour 128 switchs.

### 3.4.7 Évaluation du POX

La figure 3.11 représente les résultats de tests sous forme d'une courbe, qui représente la latence pour le contrôleur POX en fonction de nombre de switchs.

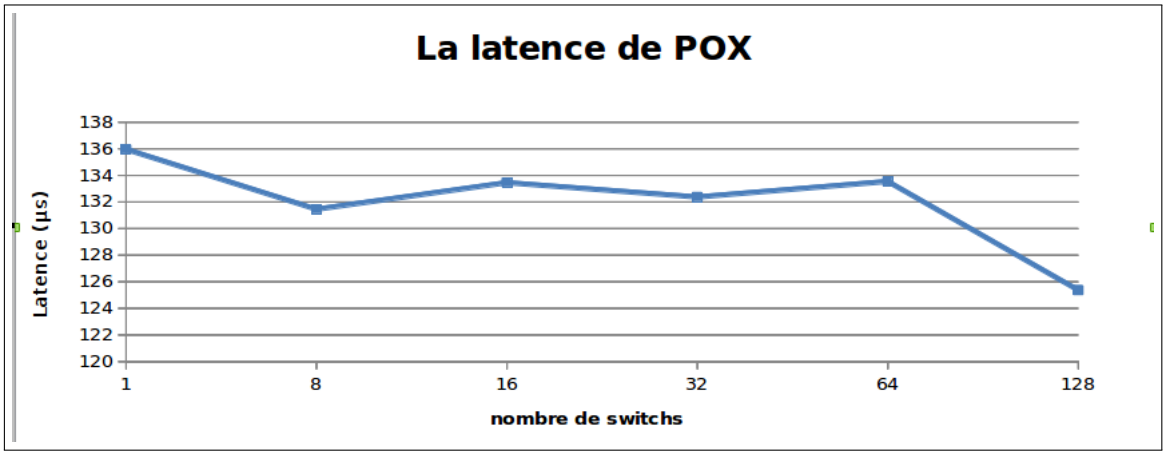


**Figure** 3.11 – *Le résultat de latence avec POX*

On peut dire que pour un seul switch connecté la latence a atteint la plus grande valeur avec (135,9 µ s) ensuite cette valeur diminue à (131 µ s) et reste stable pour le nombre de switch égale à (8, 16, 32 et 64), pour 128 switchs la valeur de latence est diminué à (125 µ s).

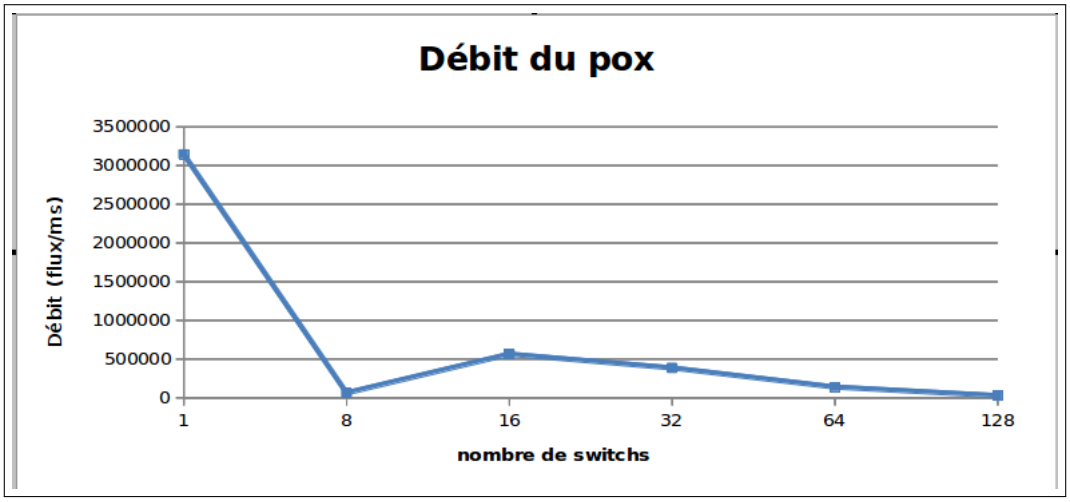


**Figure** 3.12 – *Le résultat de débit avec POX*

La figure 3.12 représente les résultats de tests de débit (flux /ms) pour le contrôleur POX.

Les résultats montrent que lorsqu'on connecte un seul Switch le débit est égale à (3 millions flux /ms) et cette valeur c'est la plus grande valeur estimée pendant ce test, pour 8 Switchs connectés le débit diminue brusquement à (61000 flux /ms) ensuite il se montrait pour atteindre (560000 flux /ms) pour (16, 32 et 64 Switchs) et diminue à (27000 flux /ms) pour 128 Switchs.

### 3.4.8 Comparaison et discussion

La figure 3.13 représente les résultats de tests de latence sous forme d'histogramme, qui représente la latence pour les deux contrôleurs NOX et POX en fonction du nombre de Switchs, ainsi que la figure 3.14 représente la comparaison des valeurs de débit pour les deux contrôleurs.

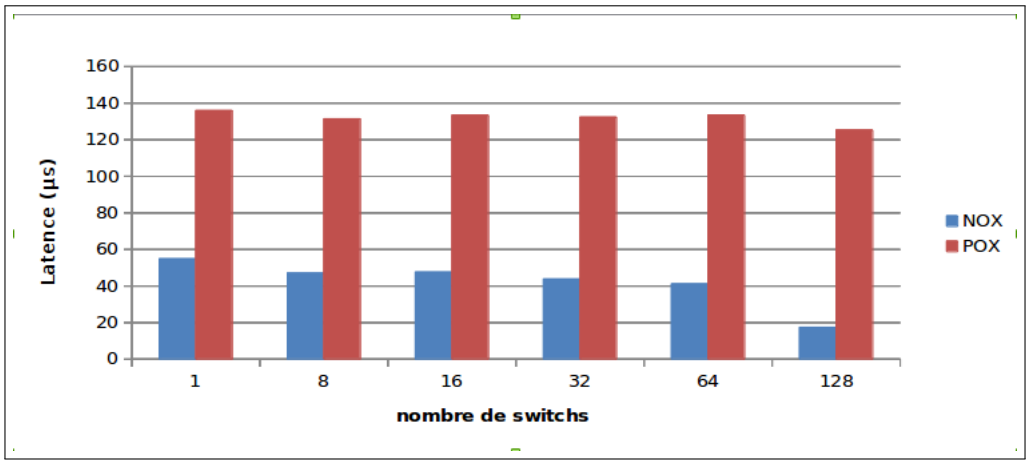


**Figure** 3.13 – *Comparaison de latence pour NOX et POX*

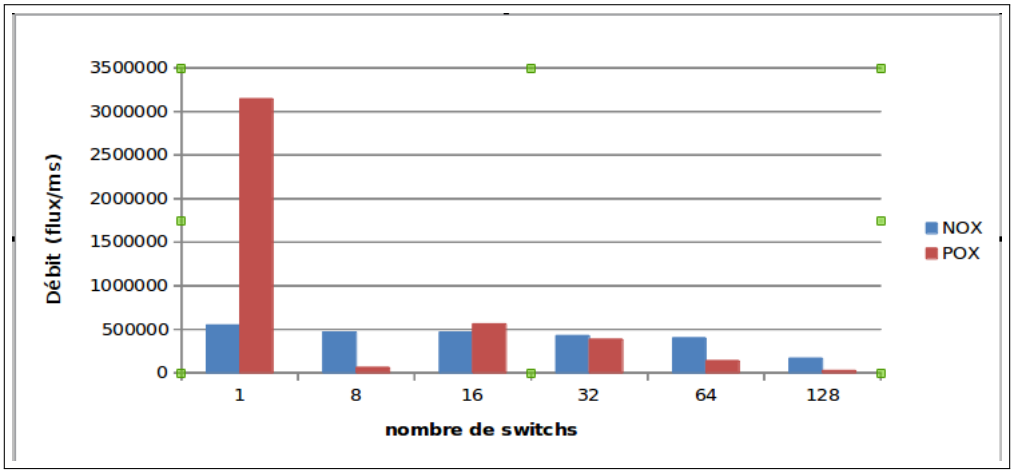


**Figure** 3.14 – *Comparaison de débit pour NOX et POX*

### Discussion

D'après les résultats des tests de débit et les tests de latence on peut dire que dans le contrôleur NOX semble avoir le problème de scalabilité pour le débit a cause de l'instabilité quand on augmente le nombre de Switch, d'autre part le débit avec le contrôleur POX pour qu'un seul Switch connecté est plus élevé mais pour 128 Switchs est plus bas ce qui indique que le contrôleur POX n'est pas adapté pour les grands réseaux, mais pour les réseaux locaux le contrôleur POX est préférable paraport le contrôleur NOX.

Le test de latence est réalisée avec de nombres différents de Switchs connectés pour voir comment le temps de réponse de ces contrôleurs peut être affecté quand la charge de travail a été augmenté.

Les tests ont montré que le contrôleur NOX a la valeur la plus faible de latence à environ (17 µ s) par flux. Tandis que le contrôleur POX a montré la plus forte latence environ(135 µ s).

Les résultats de latence sont considérés comme plus élevés dans la mise en œuvre réelle, ou dans notre test les communications entre les Switchs et le contrôleur sont effectuées par l'intermédiaire de l'interface, où il n'y a pas de retard tandis que les paquets traversant les ports virtuels par rapport aux ports matériels.

# Conclusion générale

DANS ce mémoire, l'évaluation de la performance pour les deux contrôleurs POX et NOX d'OpenFlow a été présentée ainsi qu'une analyse pour SDN et le protocole OpenFlow. Au début de ce mémoire, une étude approfondie sur le SDN a été présentée à la découverte de cette nouvelle technique en particulier dans les data centers où il peut être utilisé pour réaliser le concept de virtualisation du réseau qui est similaire à la virtualisation des serveurs dans les déploiements actuels de Data center. Puis une étude de protocole OpenFlow a été présenté pour explorer l'architecture d'OpenFlow et les contrôleurs OpenFlow.Les résultats ont été bénéfiques pour analyser le comportement de ce protocole.

Ce mémoire nous a permis de :

– Comprendre le fonctionnement d'un nouveau type de réseaux.

– Comprendre le fonctionnement de plusieurs contrôleurs.

– Améliorer nos connaissances sur les technologies de communication.

– Améliorer nos connaissances sur les réseaux informatiques.

– Se familiariser avec l'outil de l'émulation Mininet.

# Bibliographie


[A.F16] A.Frikha. *Disponible à :http ://montrealopenstack.org/fr/introduction-aux-software-defined-network-openstack/*, consulté le 1 mai 2016.

[BMX+14] B.N.Astuto, M.Mendonça, X.N.Nguyen, K.Obraczka, and T.Turletti. A survey of software-defined networking : Past,present, and future of programmable networks. *IEEE COMMUNICATIONS SURVEYS and TUTORIALS, VOL :16,NO :3*, 19 Jan 2014.

[D.E13] D.Erickson. The beacon openflow controller. *In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, Aout,2013.

[E.C16] E.Chou. Openflow tutorial with pox part 1[online]. *disponible à :http ://blog.pythonicneteng.com/2013/02/openflow-tutorial-with-pox*, consulté le 20 mai 2016.

[FC16] T. FERREIRA and V. CLOAREC. le projet openflow[online]. *Disponible à :https ://wapiti.telecom-lille.fr/commun/ens/peda/options/st/rio/pub/exposes/exposesrio2009-ttnfa2010/cloarec-ferreira/controller.html*, consulté le 1 mai 2016.

[Fou09] Open Networking Foundation. Openflow switch specification. *Version 1.0.0*, 2009.

[Fou12] Open Networking Foundation. Software-defined networking :the new norm for networks. *ONF White Paper*, Apr. 2012.

[Fou13] Open Networking Foundation. Openflow switch specification. *Version 1.4.0*, October 14, 2013.

[Fou16] Open Networking Foundation. Software-defined networking(sdn) definition[online]. *disponible à :http ://www.opennetworking.org/sdn-resources/sdn-definition*, consulté le 1 mai 2016.

[G.C16] G.Combs. Wireshark[online]. *disponible à :https ://www.wireshark.org/*, consulté le 30 avril 2016.

[HP16] HP. Openflow une technologie habilitante pour le software defined networking[online]. *Disponible à :http ://h17007.www1.hp.com/fr/fr/solutions/technology/openflow/index.aspx*, consulté le 21 mai 2016.

[J.D15] J.Durand. Le sdn pour les nuls. *Cisco*, 2015.

[M.T] M.Tury. Les risques d'openflow et du sdn. *ANSSI*.

[nox16] Github noxrepo nox : The nox controller[online]. *disponible à :http ://www.noxrepo.org*, consulté le 20 mai 2016.

[O.I15] O.IFAKREN. Software difined network. *Projet de semestre à Haute École du paysage, d'ingénierie et d'architecture de Genève*, 2015.

[R.S16] R.Sherwood. Oflops[online]. *disponible à :http ://archive.openflow.org/wk/index.php/Oflops*, consulté le 21 mai 2016.

[SA14] A. Sonba and H. Abdalkreim. Performance comparison of the state of the art openflow controllers. *mémoire de master en réseau informatique,Halmstad University*, Decembre 2014.

[SVAS15] S.T.Ali, V.Sivaraman, A.Radford, and S.Jha. A survey of securing networks using software defined networking. *IEEE TRANSACTIONS ON RELIABILITY, VOL :64,NO :03*, septembre 2015.

[Tea16] Mininet Team. Mininet,an instant virtual network on your laptop (or other pc)[online]. *disponible à :http ://mininet.org/*, consulté le 21 mai 2016.

[TMX+15] T.Chen, M.Matinmikko, X.Chen, X.Zhou, and P.Ahokangas. Software defined mobile networks : Concept, survey, and research direction. *IEEE Communications Magazine*, pages 126–133, Novembre 2015.

# Annexe

## A Mininet

### A.1 Installation de Mininet

Plusieurs versions de Mininet existent sur le net, Il est préférable de télécharger et d'utiliser la dernière version celle-ci est la plus performante. La version que nous avons utilisée est Mininet 2.2.1 téléchargée depuis le site officiel de Mininet ou bien comme le montre la figure A1 :


**Figure** A1 – *Installation de Mininet*

### A.2 Création des topologies réseaux avec Mininet

#### a. Une topologie de réseau minimal

Premièrement, retenez bien que vous devez exécuter Mininet avec les droits d'administrateur (c'est à dire avec la commande sudo).

Si tout s'est bien passé, à la fin vous devez voir l'invite de commande Mininet (le mode CLI de Mininet), et une topologie réseau devrait être créée. La topologie par défaut est appelé minimal.

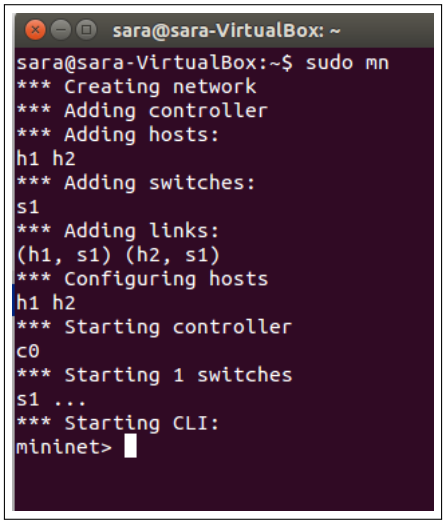


**Figure** A2 – *création d'une simple topologie avec Mininet*

### b. Autres topologies réseaux

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

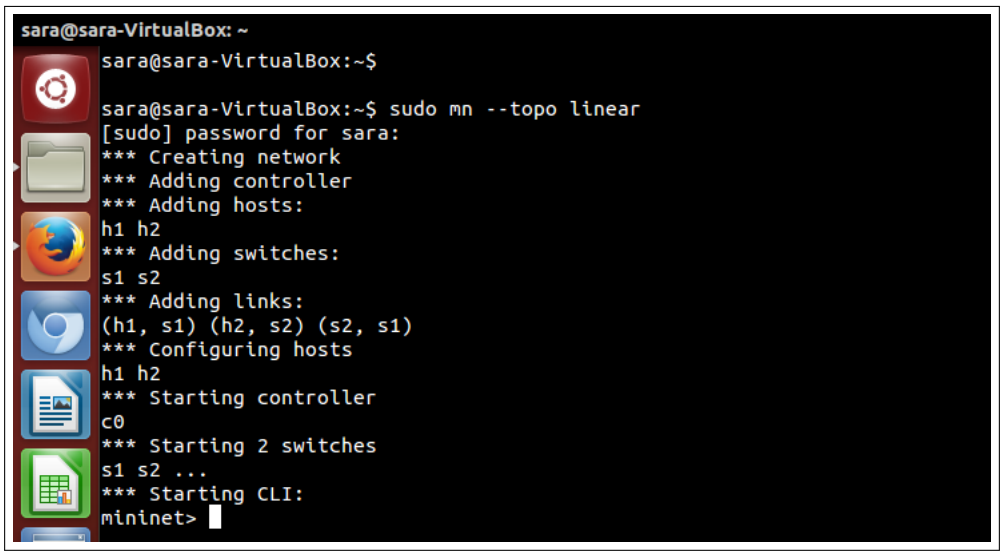


**Figure** A3 – *La topologie de réseau linear*

```
sara@sara-VirtualBox:~$ sudo mn --topo single,3 --mac --switch ovsk --controller
 remote
[sudo] password for sara:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

**Figure** A4 – *La topologie de réseau single*

## A.3 MiniEdit

Le simulateur de réseau Mininet comprend MiniEdit, un éditeur graphique simple pour Mininet. MiniEdit est un outil expérimental créé pour démontrer comment Mininet peut être déployé.

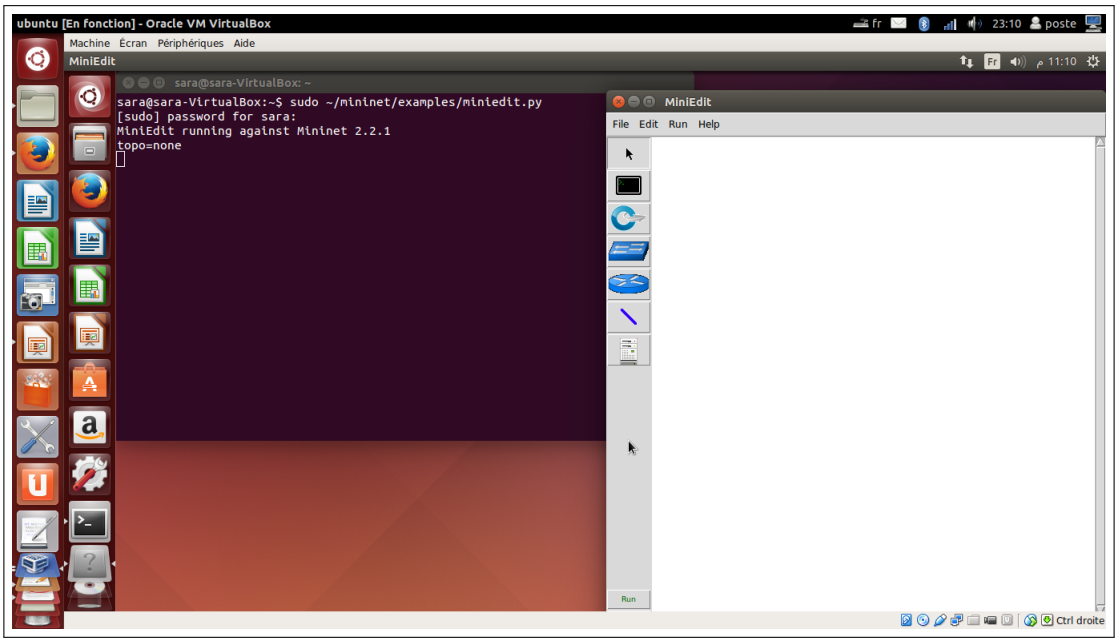


**Figure** A5 – *L'interface de MiniEdit*

### a. Création d'une topologie de réseau personnalisé à l'aide de MiniEdit

D'abord nous allons ajouter certains hosts pour le scénario de réseau. Cliquez sur l'icône d'host, puis déplacez le pointeur vers l'emplacement sur la toile MiniEdit où vous voulez l'host à apparaître, puis cliquez à nouveau. Une icône host apparaîtra sur la toile.

Tant que l'outil host est actif, vous pouvez ajouter plusieurs hosts. Continuez à cliquer à chaque endroit sur la toile où vous voulez un host apparaisse. Dans cet exemple, nous allons ajouter 4 hosts.

Ajouter 3 Switchs et un contrôleur utilisant la même méthode : cliquez sur l'outil Switch et ajouter des Switchs, puis cliquez sur l'outil de contrôleur et ajouter des contrôleurs.

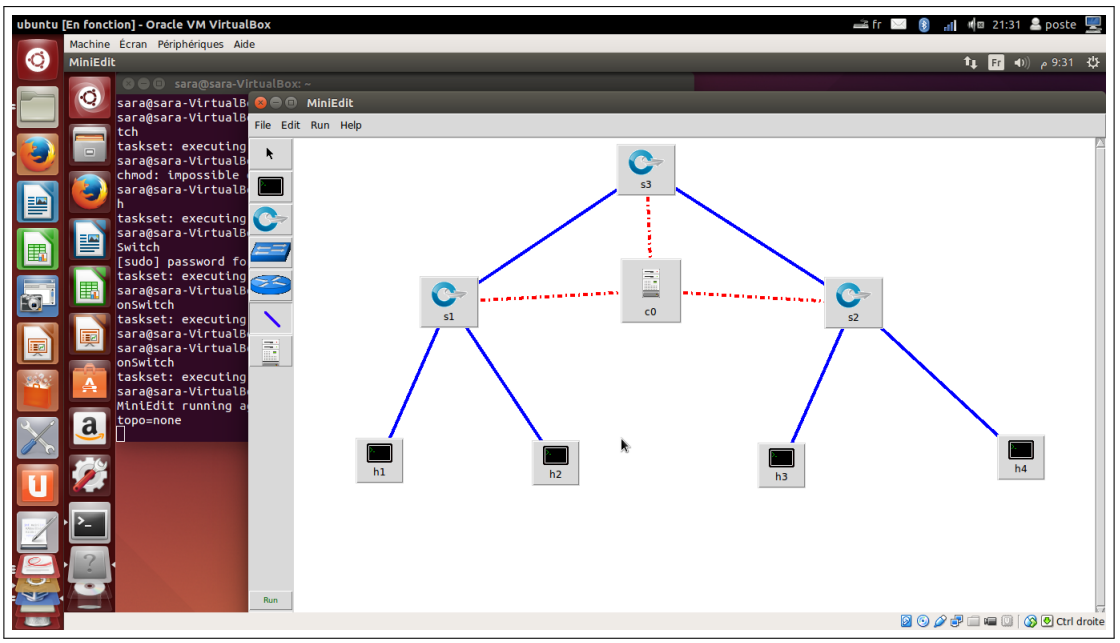


**Figure** A6 – *Une topologie de réseau avec MiniEdit*

# B Wireshark

Wireshark est un analyseur de protocole de réseau. Il nous permet de capturer et de parcourir le trafic en cours d'exécution sur un réseau informatique. Il dispose d'un ensemble de fonctionnalités riches et puissantes et il est l'outil le plus populaire en son genre.
La figure A7 montre comment on filtre pour capturer seulement le trafic OpenFlow :

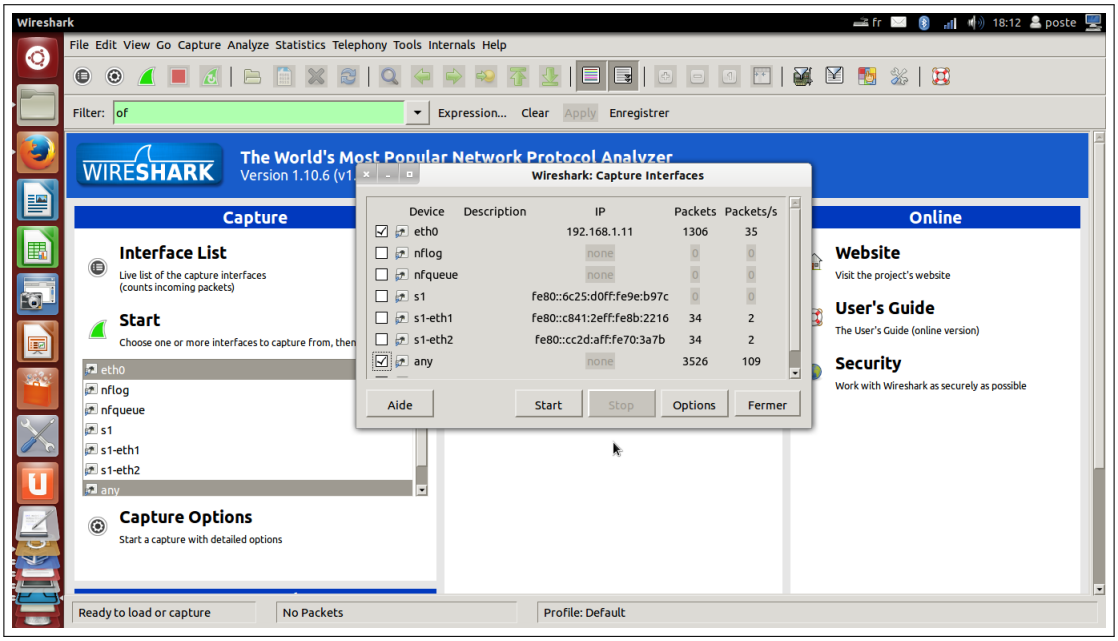


**Figure** A7 – *L'interface de Wireshark*

# C POX

Aprés l'installation du POX il faut executer l'application (pox.py) pour que le le contrôleur démarre. La figure A8 montre le démarrage du POX :

```
sara@sara-VirtualBox:~/pox/pox/forwarding$ udo ~/pox/pox.py forwardin
\
> info.packet_dump samples.pretty_log log.level --DEBUG
Le programme « udo » n'est pas encore installé. Vous pouvez l'install
t :
sudo apt-get install udo
sara@sara-VirtualBox:~/pox/pox/forwarding$ sudo ~/pox/pox.py forwardi
 info.packet_dump samples.pretty_log log.level --DEBUG
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:forwarding.l2_pairs:Pair-Learning switch running.
INFO:info.packet_dump:Packet dumper running
[core                  ] POX 0.2.0 (carp) going up...
[core                  ] Running on CPython (2.7.6/Jun 22 2015 18:00
[core                  ] Platform is Linux-4.2.0-27-generic-i686-wit
.04-trusty
[core                  ] POX 0.2.0 (carp) is up.
[openflow.of_01        ] Listening on 0.0.0.0:6633
```

**Figure** A8 – *Démarrer le contrôleur POX*

# D NOX

NOX doit être invoqué par la ligne de commande dans le répertoire de build/src. Généralement, la commande qui lance le contrôleur ressemble à ceci :

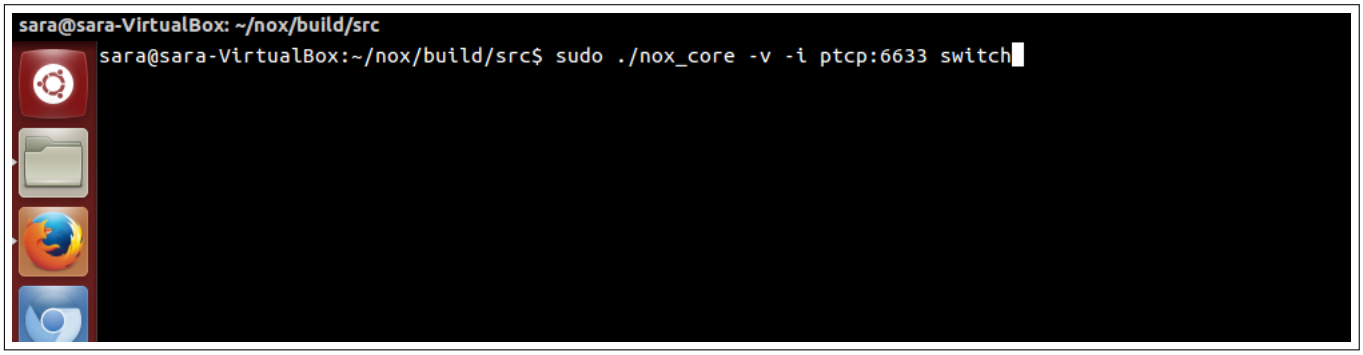


**Figure** A9 – *L'exécution du contrôleur NOX*

Aprés l'exécution de la commande, le contrôleur est démarré comme la figure A10 montre :

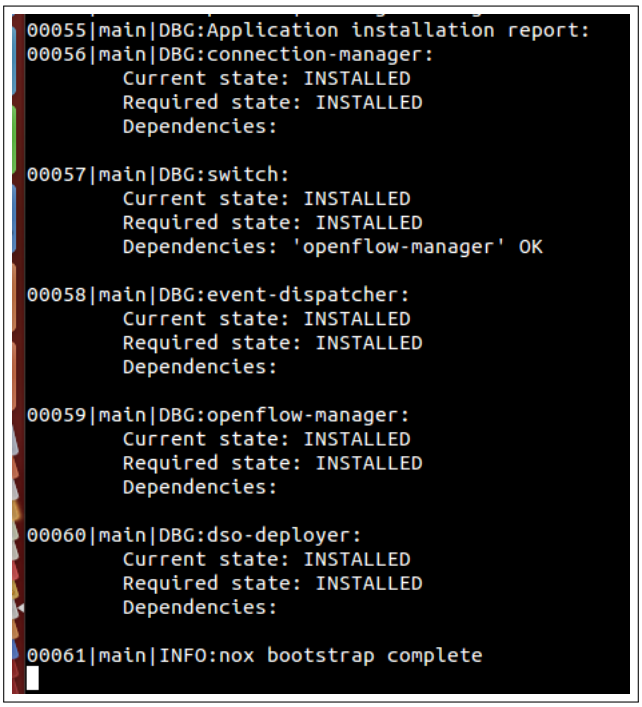


**Figure** A10 – *Le contrôleur NOX est démarré*