REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

SONATRACH

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

Mise en ouvre d'un réseau SDN et évaluation de performance des contrôleurs OpenDaylight et ONOS

Hadjer Medjhoum

Présenté et soutenu publiquement le (06/07/2022)

Composition du Jury :

Président	**M. Zergui Rachid**	**MAA**	**ENP**
Promotrice	**Mme. Lani Fatiha**	**MAA**	**ENP**
Co-prometteur	**M. Neffah Mohamed**	**Cadre ingénieur**	**SONATRACH**
Examinatrice	**Mme. Bouadjenek Nesrine**	**MCB**	**ENP**

ENP 2022

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

SONATRACH

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

Mise en ouvre d'un réseau SDN et évaluation de performance des contrôleurs OpenDaylight et ONOS

Hadjer Medjhoum

Présenté et soutenu publiquement le (06/07/2022)

Composition du Jury :

Président	**M. Zergui Rachid**	**MAA**	**ENP**
Promotrice	**Mme. Lani Fatiha**	**MAA**	**ENP**
Co-prometteur	**M. Neffah Mohamed**	**Cadre ingénieur**	**SONATRACH**
Examinatrice	**Mme. Bouadjenek Nesrine**	**MCB**	**ENP**

ENP 2022

ملخص:

أدت الإنترنت إلى إنشاء مجتمع رقمي، حيث تقريبًا كل شيء متصل. ومع ذلك، على الرغم من اعتمادها على نطاق واسع، فإن شبكات ال IP التقليدية يصعب للغاية إدارتها. ما يجعل الأمور معقدة هو أن شبكات اليوم متداخلة عموديا: يتم تجميع خطط التحكم والبيانات معًا. تعد الشبكات المحددة برمجيا(SDN) نموذجًا ناشئًا يغير ذلك عن طريق فصل منطق التحكم عن أجهزة التوجيه في الشبكة، وتعزيز تحكمها المركزي، وإدخال قابلية برمجتها وتبسيط إدارتها وتطورها. في هذا المشروع، نقدم دراسة مفصلة عن الSDN . نقوم بمحاكاة وتكوين وبالتحكم بالبنية التحتية لشبكة مركز البيانات الخاص بنا من خلال محاكيMininet وواجهات برمجة التطبيقات (APIs) لوحدات تحكم الSDN. من أجل توقع التطور المستقبلي لهذا النموذج الجديد، نختار الدراسة المتعمقة لاثنين من وحدات التحكمSDN ، وهما OpenDaylight وONOS .على وجه الخصوص، نقوم بالتركيز على جانب أداء الشبكة الخاضعة لسيطرتهم، مثل الإنتاجية والكمون والرقصة.

الكلمات المفتاحية: الشبكات المحددة برمجيا، OpenVswitch، OpenDaylight، ONOS، Iperf، مركز البيانات.

Abstract :

The Internet has led to the creation of a digital society, where almost everything is connected. However, despite their widespread adoption, traditional IP networks are very difficult to manage because of their vertical integration : the control and data planes are bundled together. The Software Defined Network (SDN) is an emerging paradigm that changes this state of affairs, separating the control logic from the network equipment, favoring the centralization of its control and introducing the possibility to program it and simplify its management while facilitating its evolution. In this project, we present a detailed study on SDN. We emulate, configure and control our data center network infrastructure through the Mininet emulator and the SDN controller application programming interfaces (APIs). In order to anticipate the future evolution of this new paradigm, we choose to study in depth two of the SDN controllers namely OpenDaylight and ONOS. In particular, we address the network performance aspect in terms of throughput, latency, and jitter under their control.

Key words : SDN, OpenVswitch, OpenDaylight, ONOS, iperf, Data center.

Résumé :

L'internet a conduit à la création d'une société numérique, où presque tout est connecté. Cependant, malgré leur adoption généralisée, les réseaux IP traditionnels sont très difficiles à gérer à cause de leur intégration verticale : les plans de contrôle et de données sont regroupés. Le réseau défini par logiciel (SDN) est un paradigme émergeant qui change cet état de fait, en séparant la logique de contrôle, des équipement du réseau, en favorisant la centralisation de son contrôle et en introduisant la possibilité de le programmer et de simplifier leur gestion tout et facilitant son évolution. Dans ce projet, nous présentons une étude détaillée sur le SDN. Nous émulons, configurons et contrôlons notre infrastructure réseau data center à travers l'émulateur Mininet et les interfaces de programmation d'application (API) des contrôleurs SDN. Dans le but d'anticiper l'évolution future de ce nouveau paradigme, nous choisissons l'étude approfondit de deux des contrôleurs SDN à savoir OpenDaylight et ONOS. En particulier, nous abordons l'aspect de performances du réseau en terme de débit, de latence, et de la gigue sous leurs contrôle.

Mots clés : SDN, OpenVswitch, OpenDaylight, ONOS, iperf, Data center.

Dédicace

Ce mémoire est dédié principalement à mon défunt papa Hamid 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 Lynda, mon âme, qui a toujours cru en moi, et qui m'a permis de réaliser mes projets, même si aucune dédicace ne saurait exprimer ma profonde gratitude et ma vive reconnaissance envers tous les sacrifices qu'elle a faits pour moi.

À mon cher fiancé Aymen, le trésor le plus précieux que mon Seigneur m'a donné, ma source de joie, de courage et de l'espoir.

À ma superbe sœur Romaissa, Rofaida, à mon petit frère Mohammed. Vous êtes ma plus grande richesse, Je vous aime !

Aux frères d'âme de mon père : Madjid, Rachid et sa femme Fazia.

Je dédie aussi ce travail à toute ma famille, mes amies, Amina, Houria et à tous mes professeurs qui m'ont enseigné, et à tous ceux qui me sont chers.

Hadjer

Remerciement

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.

Avant tout, je tiens à exprimer ma profonde gratitude à ma promotrice Mm Fatiha LANI pour la confiance qu'elle ma a accordé en acceptant si généreusement de m'encadrer tout au long de mon travail, moyennant ses critiques constructives et ses suggestions pertinentes. Je la remercie pour son implication, ses conseils et l'intérêt qu'elle a porté à mon travail.

J'adresse mon vifs remerciements aux membres des jurys pour avoir accepté d'examiner et juger ce modeste travail.

Je remercie également M. NEFFAH Mohammed ainsi que toute l'équipe de SONATRACH pour leur disponibilité, leurs précieux conseils et pour m'avoir introduit au monde professionnel.

Je tient aussi à remercier ma mère, J 'espère qu'elle sera toujours fière de moi mes chères familles pour leurs soutient, encouragements et leurs bienveillance pour notre bien-être et notre succès. Je tient à remercier mes amies pour leur sincère amitié et confiance.

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

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ACL	A network access control list
AD-SAL	API Driven SAL
API	Application programing interface
BGP	Border Gateway Protocol
CLI	Command Line Interface
DAL	Data access layer
EGP	Exterior Gateway Protocol
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IS-IS	Intermediate system to intermediate system
IGRP	Interior Gateway Routing Protocol
MAC	Media Access Control
MD-SAL	Model Driven SAL
MIB	Management information base
NETCONF	NETwork CONFiguration protocol
NIB	Network Information Base
NFV	NetworkFunction Virtualisation
ODL	Opendaylight
OF	Openflow
OFM	OpenFlow Manager
ONF	Open Networking Foundation
ONOS	Open Networking Operating System
OSGI	Open Services Gateway Initiative
OSPF	Open Shortest Path First
OVS	Open Virtual Switch
POX	Pythonic Network Operating System
QOS	Quality of Service

RAM	Read Access memory
REST	REpresentational State Transfer
RIB	Routing Information Base
SAL	Service Abstraction Layer
SDN	Software Define Networking
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
UDP	User Datagram Protocol
VM	Virtual Machine
VMM	Virtual Machine Monitor
YANG	Yet Another Next Generation

Introduction

Contexe et motivation

De nos jours, les réseaux télécoms et par-delà, l'opérateur télécommunication pour les professionnels impliqués dans leur installation ont une grande importance et une place essentielle dans le fonctionnement d'une entreprise telle que l'entreprise SONATRACH. Du moment que les technologies de l'information et de la communication sont utilisables à tous les niveaux et ce par la plupart des professionnels. En deux décennies, l'architecture statique des réseaux traditionnels n'a connu aucun changement majeur, seuls les techniques hardware ont évolué pour proposer de meilleurs débits de transmission (Giga Ethernet, Fibre optique ....). Cette vision simpliste est dépassée et n'est pas adéquate avec la plupart des déploiements de centres de données.

Aujourd'hui, 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.

La transformation de l'infrastructure du réseau d'aujourd'hui en une solution plus robuste et plus rentable qui soutient l'innovation rapide, comprend deux éléments-clés : premièrement, l'abandon des plateformes matérielles personnalisées et exclusives au profit de plateformes standard de l'industrie, moins coûteuses et dotées de caractéristiques qui prennent en charge les fonctions de communication et, deuxièmement, la transition vers une architecture de réseau contrôlée par logiciel.

C'est dans ce contexte qu'a émergé l'idée du Software Defined Networking (SDN), ou réseau défini à base de logiciels. Un modèle qui a vu le jour ces dernières années, avec la promesse d'un réseau programmable et une préface très attirante à première vue. À cet effet, la majeure partie de la communauté réseau y compris des géants de l'industrie (Cisco, Citrix, Juniper, HP…), y ont concentré leur intérêt. Signe que la transition vers ce type d'architecture est inévitable.

Un réseau défini par logiciel (SDN) est un nouveau paradigme de mise en réseau qui ap-

porte un grand nombre de nouvelles capacités et permet de résoudre de nombreux problèmes difficiles des réseaux existants. Cette approche consiste à séparer l'intelligence du réseau du dispositif de commutation par paquets et à la centraliser logiquement dans un contrôleur. Le contrôleur est responsable des décisions de transfert qui sont prises dans les commutateurs via des protocoles standard, comme OpenFlow. La motivation du SDN est de réaliser un système d'exploitation de réseau, où les tâches du réseau peuvent être effectuées sans ajouter de logiciel supplémentaire pour chacun des éléments de commutation, il permet également de développer des applications qui contrôlent les commutateurs en les exécutant au-dessus d'un système d'exploitation réseau.

Présentation de l'organisme d'accueil SONATRACH

SONATRACH intervient dans l'exploitation, la production, le transport par canalisation, la transformation et la commercialisation des hydrocarbures et de leurs dérivés, elle se développe encore dans les activités de la pétrochimie, de raffinage, de génération électrique, d'énergies nouvelles et renouvelables, de dessalement d'eau de mer... . Nous présentant dans cette section une énumération de ses principales missions, ainsi que son organisation générale.

Missions de SONATRACH

Face à l'évolution de l'environnement actuel et à la situation économique mondiale SONATRACH a pour mission :

1 La prospection, la recherche, l'exploitation et le développement des gisements de pétrole et gaz naturel ainsi que la reconstitution et l'accroissement des réserves d'hydrocarbures ;

2 La construction, l'exploitation industrielle et commerciale de tous les moyens de transport d'hydrocarbures par voie terrestre ;

3 Le traitement, la transformation et la commercialisation des hydrocarbures et des produits dérives ainsi que leur approvisionnement énergétique a moyen terme ;

4 L'intensification des efforts d'exploitation et capitalisation des études réalisées dans ce domaine, pour une meilleure connaissance de sous-sol et la mise en évidence des réserves d'hydrocarbures potentielles ;

5 Le développement, l'exploitation et la gestion des réseaux de transport, de stockage et de chargement des hydrocarbures ;

6 Le développement des techniques modernes de gestion nationale par le biais de la formation continue.

Organisation Générale de SONATRACH

Le schéma d'organisation de la macrostructure de SONATRACH s'inscrit dans le cadre de l'évolution de son environnement interne ainsi qu'externe, et au service de ses objectifs :

— Conforter la Direction Générale dans son rôle de conception de la stratégie, d'orientation, de coordination, de pilotage et de management ;

— Concentrer les structures opérationnelles pour une meilleure synergie en veillant à leur assurer une meilleure efficacité ;

— Permettre une décentralisation accompagnée d'une maitrise des pouvoirs et d'une clarté en matière de responsabilités dans le cadre de procédures bien établies tout en renforçant le contrôle ;

— Assurer la réactivité, la transparence et la fluidité de l'information nécessaire à la conduite et au pilotage des activités dans le but d'assurer l'efficacité globale de l'entreprise.

Le nouveau schéma de la réorganisation de la macrostructure de SONATRACH s'articule autour des structures suivantes : la Direction Générale, les Structures Fonctionnelles Centrales et les Structures Opérationnelles.

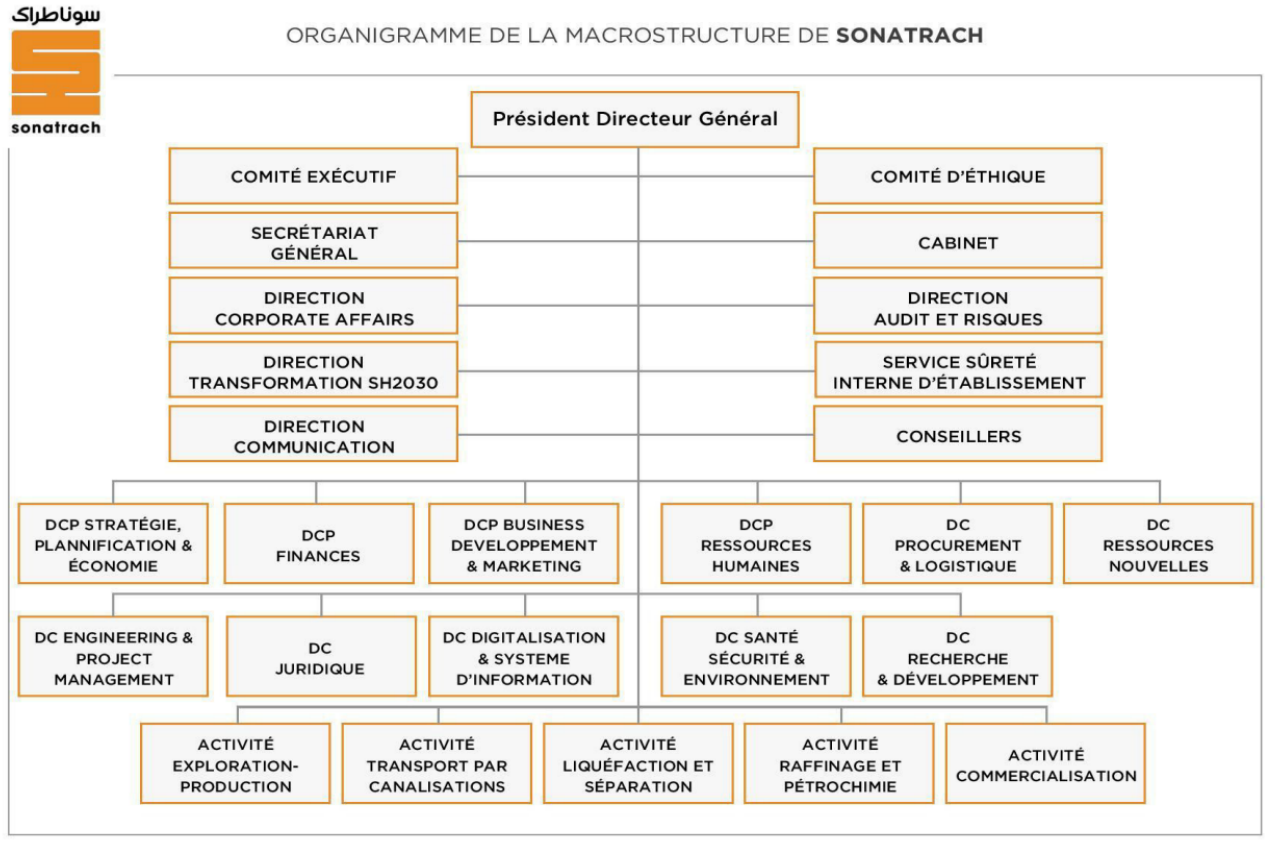

FIGURE 0.1 – Organigramme de la macrostructure de SONATRACH

Objectifs

Les principaux objectifs de ce mémoire sont les suivants :

1 Création et émulation d'un réseau SDN virtuel avec une topologie Data Center sous l'émulateur Mininet.

2 Contrôle et installation des flux du réseau avec le contrôleur OpenDayligt et ONOS.

3 Étude comparative et test de performance du réseau sous le contrôleur OpenDayligt dans un premier lieu, puis sous et contrôleur ONOS dans un deuxième lieu.

Organisation du mémoire

Le document est organisé comme suit :

— Le **Chapitre 1** résume en bref les limitation des réseaux traditionneles et le besoin de les faire évoluer. Il présente la virtualisation, le paradigme du SDN (son principe et son architecture), et le protocole dominant : OpenFlow. On montre aussi la relation du SDN avec le NFV ainsi que les avantages et défis face à son intégration dans les réseaux.

— **Le chapitre 2 :** présente les travaux connexes sur les outils (hyperviseurs, émulateurs et contrôleurs) afin de faire notre choix, et décrit entre autre les outils choisit.

— **Le chapitre 3 :** décrit la partie pratique. Nous mettons en ouvre un réseau avec une infrastructure SDN selon une topologie de type DATA CENTER que nous allons l'implémenter en utilisant l'outil MININET, nous montrons le contrôle du réseau à partir des expériences d'installation de flux sur les plateformes de contrôleurs retenue après étude comparative : OpenDayLight et ONOS. Nous testons ainsi le bon fonctionnement de notre réseau. Nous nous intéresserons dans une seconde phase à l'évaluation des performances de notre réseau à travers des expériences de test avec l'outil Iperf, afin de dresser notre table comparative finale.

— Finalement on **conclut**, et on présente les travaux futurs possibles.

— Et pour terminer, il y a la partie annexe qui contient les étapes d'installation de certains systèmes constituant l'environnement ainsi que des scripts.

Chapitre 1

Limitations des réseaux traditionnels et le paradigme du SDN

1.1 Introduction

Dans ce chapitre, nous allons introduire brièvement les réseaux traditionnels et les principaux défis des architectures des réseaux de télécommunications de nos jours, ainsi que les nouveaux concepts utilisés dans les réseaux à savoir la virtualisation. Nous présentons le paradigme SDN, ainsi que les concepts nécessaires à la conception et à la réalisation de notre projet. Enfin nous citons les principaux avantages et défis face à l'intégration du SDN, et nous terminons le chapitre par une conclusion.

1.2 Les réseaux traditionnels

Un réseau est un moyen de communication qui permet aux individus ou aux groupes de partager des informations et des services. Les réseaux de communication ont pour finalité de transmettre de l'information (ou bien des ressources informationnelles).

Un réseau est constitué d'équipements appelés nœuds. En fonction de leur étendue et de leur domaine d'application, ces réseaux sont catégorisés. Pour communiquer entre eux, les nœuds utilisent des protocoles, ou langages, compréhensibles par tous [6] .

1.2.1 Les équipements de routage et de commutation dans un réseau IP traditionnel

Sur un réseau IP, la mise en relation entre un client et un serveur s'appuie sur des équipements de routage et des équipements de commutation.

Un routeur est un équipement d'interconnexion de réseaux informatiques permettant d'assurer le routage des paquets entre deux réseaux ou plus afin de déterminer le chemin qu'un paquet de données va emprunter. Avant d'être déployés sur le réseau, les routeurs doivent être configurés. La configuration permet au minima de définir les adresses IP des interfaces physiques et virtuelles du routeur et d'informer le routeur des protocoles de routage à appliquer pour acheminer les paquets IP. Les protocoles de routage permettent à chaque routeur de récupérer des informations des routeurs voisins afin de constituer localement des informations de routage (RIB : Routing Information Base). Ainsi, les informations de routage sont actualisées pour prendre en compte l'état de chaque nœud (saturé, hors ligne, ...) de manière dynamique. Les routeurs échangent entre eux des informations par l'intermédiaire du protocole de routage choisi, ensuite les informations de routage permettent de construire une table d'acheminement (Forwarding Information Base) qui est exploitée par le routeur pour définir l'interface sur laquelle envoyer le paquet (adresse de destination, classe de service).

Les informations de routages transmises entre routeurs dépendent du protocole de routage choisi :

(i) **Protocole de routage par état de liens :** Les routeurs de type link state routing écoutent le réseau en continu afin de recenser les différents éléments qui l'entourent.

A partir de ces informations chaque routeur calcule le plus court chemin (en temps) vers les routeurs voisins et diffuse cette information sous forme de paquets de mise à jour, chaque routeur construit enfin sa table de routage. Exemples de protocoles : OSPF, IS-IS ;

(ii) **Protocole de routage par vecteur de distance :** Les routeurs de type vecteur de distance (distance vector) établissent une table de routage recensant en calculant le « coût » (en terme de nombre de sauts) de chacune des routes puis transmettent cette table aux routeurs voisins. A chaque demande de connexion le routeur choisit la route la moins coûteuse. Exemples de protocoles : RIP, RIPv2, IGRP, etc ;

(iii) **Protocole BGP :** Les protocoles mentionnés précédemment sont des protocoles (Interior Gateway Protocol) permettant d'assurer le routage dans des zones limitées. Par contre, le routage sur Internet se fait par un protocole appelé BGP (Border Gateway Protocol) qui est dit EGP (Exterior Gateway Protocol). BGP est le seul protocole de routage à utiliser TCP comme protocole de transport.

1.2.2 Limitations des réseaux traditionnels et le besoin de les faire évoluer

Les architectures des réseaux traditionnels existantes n'ont pas été conçues de manière à répondre aux exigences actuelles des utilisateurs finaux, des fournisseurs de services et des entreprises. Certaines limites de l'architecture de réseau traditionnel sont les suivantes [7] [8] [9] :

(i) **La complexité de la gestion :**
Les technologies de réseau informatique précédentes ont toujours été construites sur un ensemble de protocoles de routage conçus pour connecter des hôtes de manière fiable sur de longues distances, à des vitesses élevées et selon différentes conceptions de réseau. Afin de répondre aux exigences de l'industrie telles que la haute disponibilité, la sécurité et la connectivité étendue, au cours des dernières décennies, les protocoles ont été conçus de nombreuses façons qui conduisent à la séparation, où chaque protocole doit résoudre un type spécifique de problèmes, sans garder à l'esprit de bénéficier des abstractions. Cette approche de la conception a conduit à l'un des principaux problèmes auxquels les administrateurs de réseau sont confrontés aujourd'hui, à savoir la complexité de la gestion du réseau. Par exemple, l'ajout ou la suppression d'un dispositif sur un réseau est devenu un fardeau pour les administrateurs de réseau, où plusieurs parties du réseau doivent être reconfigurées, telles que : Listes d'accès, VLANs politiques de qualité de service, protocoles de routage et topologies de réseau. En plus de ce qui précède, il faut tenir compte de la compatibilité des fournisseurs d'équipements et des versions de logiciels avant d'apporter toute modification au réseau. Par conséquent, les administrateurs de réseau gardent leur réseau plutôt statique, afin d'éviter ou de minimiser les interruptions de service qui peuvent être causées par tout changement. La nature statique de la conception du réseau limite la nature dynamique de la virtualisation des serveurs, qui à son tour augmente le nombre d'hôtes qui ont besoin de connectivité.

Avant l'introduction du service de virtualisation du temps, un seul serveur se connecte à des clients sélectionnés. Alors qu'aujourd'hui, grâce aux services que la virtualisation, il est devenu possible de répartir les applications sur plusieurs machines virtuelles qui se connectent les unes aux autres. Et dans de nombreux cas, les machines virtuelles (VM) (voir 1.3.2.2) doivent migrer pour obtenir des charges de travail équilibrées ; cette fonctionnalité des plates-formes virtualisées pose de nombreux défis au réseau traditionnel qui n'a pas été conçu pour de tels changements de flux dynamiques.

(ii) **Difficulté d'appliquer les politiques :**
Afin de maintenir une politique de réseau d'entreprise, les administrateurs de réseau devraient configurer des centaines de routeurs, de commutateurs et de mécanismes. Dans la virtualisation, chaque fois qu'on ajoute une VM au réseau, cela prend normalement jusqu'à des heures, voir des jours. l'administrateur réseau doit configurer et ajuster les listes d'accès (ACL) sur l'ensemble du réseau. Avec le type de réseau que nous gérons aujourd'hui, il est devenu très difficile pour les administrateurs réseau de maintenir un cadre cohérent pour les privilèges d'accès, la qualité des services et la sécurité.

(iii) **Problèmes d'évolutivité :**
Normalement, les centres de données ont une forte demande de croissance rapide, et en même temps le réseau doit croître à la même vitesse. Mais en réalité, les réseaux sont devenus extrêmement complexes en raison de l'ajout de centaines, voire de milliers de routeurs, de commutateurs et même de pare-feu qui doivent être gérables et configurables. Les administrateurs de réseaux se sont toujours appuyés sur la sursouscription de la bande passante pour faire évoluer les réseaux d'entreprise, mais avec la virtualisation des centres de données, les modèles de trafic sont devenus très dynamiques, ce qui a rendu difficile la prévision des modèles de trafic. Les grands opérateurs, comme Amazon, EBay et Facebook, sont confrontés à des problèmes d'extensibilité encore plus difficiles. De tels réseaux à grande échelle sont impossibles à configurer manuellement.

(iv) **Fiabilité de la fabrication des équipements :**
Les fournisseurs d'accès à Internet et les centres de données cherchent toujours à mettre en œuvre de nouvelles fonctionnalités et de nouveaux services pour satisfaire les exigences changeantes du secteur ou les demandes des utilisateurs finaux. Normalement, la capacité de réponse est limitée par les cycles de vie du fournisseur d'équipement pour le service et l'équipement produits, qui dans certains cas peuvent être d'environ trois ans ou même plus. De plus, l'absence d'interfaces ouvertes et normalisées a réduit la capacité des opérateurs de réseau à adapter le réseau à leurs environnements. Une telle inadéquation entre les capacités du réseau et les exigences des clients a fortement affecté l'industrie.

Actuellement, dans les réseaux traditionnels, tous les dispositifs de réseau sont constitués d'un plan de contrôle et d'un plan de transfert, ces programmes de contrôle sont étroitement couplés à chaque dispositif distribué sur le réseau. Cependant, le réseau définie par logiciel ou bien SDN (voir 1.4) émerge en créant un modèle de réseau où le plan de contrôle et le plan d'acheminement sont séparés, ce qui permet de gérer le plan de contrôle à l'aide d'un contrôleur centralisé programmable où les politiques sont configurées pour chaque dispositif,

qu'il s'agisse d'un routeur, d'un commutateur ou d'un pare-feu, c'est-à-dire que nous passons d'un réseau avec un plan de contrôle distribué à un réseau avec un plan de contrôle centralisé.

1.3 La virtualisation

1.3.1 Définition de la virtualisation

La virtualisation consiste à construire des réseaux logiciels pour remplacer les réseaux matériels. Elle permet de découpler des fonctions et d'utiliser des machines partagées pour héberger des algorithmes, ce qui permet de réaliser des économies substantielles en termes de ressources et de personnel qualifié [8].

1.3.2 Technologies de virtualisation

La virtualisation s'appuie sur les logiciels pour simuler une fonctionnalité matérielle et créer un système informatique virtuel. Ce modèle permet aux services informatiques d'exécuter plusieurs systèmes virtuels (et plusieurs systèmes d'exploitation et applications) sur un seul et même serveur. Les conteneurs et les machines virtuelles sont des technologies de virtualisation très similaires, les principales critère de différenciation entre eux sont cités par la suite :

1.3.2.1 Les conteneurs

Les conteneurs sont placés au-dessus d'un serveur physique et de son système d'exploitation hôte, par exemple Linux ou Windows. Chaque conteneur partage le noyau du système d'exploitation hôte et, généralement, les binaires et les bibliothèques également. Les composants partagés sont en lecture seule. Les conteneurs sont donc exceptionnellement "légers" : ils ne font que des mégaoctets et leur démarrage ne prend que quelques secondes, contre des gigaoctets et des minutes pour une VM.

Les conteneurs réduisent également les frais généraux de gestion. Parce qu'ils partagent un système d'exploitation commun, un seul système d'exploitation doit être entretenu et alimenté en corrections de bogues, en correctifs, etc. Un conteneur présente de nombreux risques de sécurité et des vulnérabilités [10].

1.3.2.2 Les Machines virtuelles

Dans chaque machine virtuelle s'exécute un système d'exploitation invité unique. Des machines virtuelles avec des systèmes d'exploitation différents peuvent fonctionner sur le même serveur physique - une machine virtuelle UNIX peut fonctionner à côté d'une machine virtuelle Linux, et ainsi de suite. Chaque VM possède ses propres binaires, bibliothèques et applications qu'elle dessert, et sa taille peut atteindre plusieurs gigaoctets.

Chaque VM comprend une image de système d'exploitation distincte, ce qui ajoute une surcharge en termes de mémoire et d'empreinte de stockage. Il s'avère que ce problème complexifie toutes les étapes du cycle de vie d'un logiciel, du développement et des tests à la production et à la reprise après sinistre. Cette approche limite également fortement la portabilité des applications entre les clouds publics, les clouds privés et les centres de données traditionnels. Il existe une forte isolation dans le noyau hôte. Par conséquent, ils sont plus sûrs que les conteneurs. La figure suivante illustre la différence entre les VMs et les conteneurs [10].

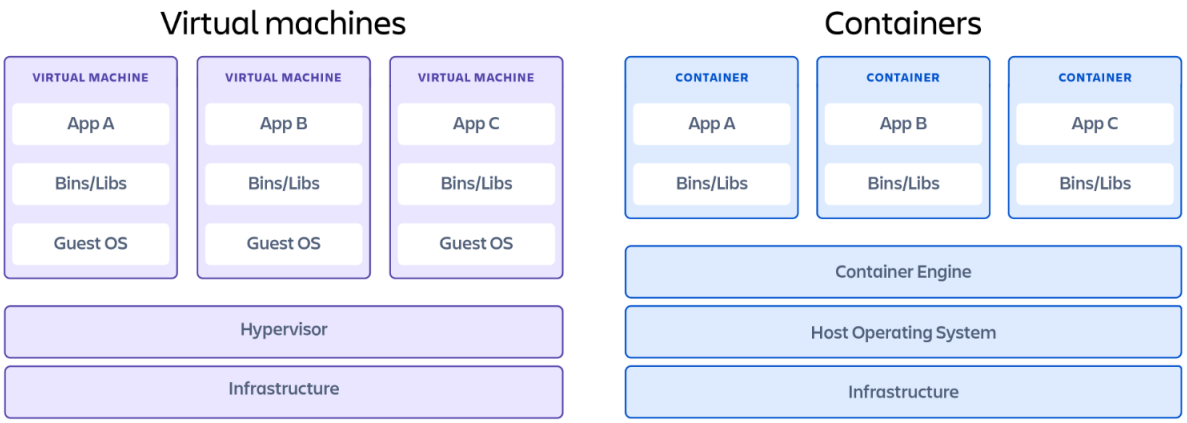

FIGURE 1.1 – Conteneurs vs Machines virtuelles [1]

Les conteneurs ont surpassé les machines virtuelles en termes de performances et d'évolutivité. En raison de leur meilleure évolutivité et utilisation des ressources, les conteneurs peuvent être utilisés pour les déploiements d'applications afin de réduire la surcharge en ressources. Cependant, il existe des cas d'utilisation où les machines virtuelles seraient plus adaptées que les conteneurs. L'un de ces cas d'utilisation est l'exécution d'applications contenant des données critiques pour l'entreprise [10], et les VMs sera notre choix, puisque la sécurité est primordiale dans un réseau d'entreprise.

1.3.3 Avantages de la virtualisation

La virtualisation de serveur débloque l'architecture traditionnelle des serveurs en abstrayant le système d'exploitation et les applications du matériel physique, ce qui permet d'obtenir un environnement de serveur plus rentable, plus agile et plus simple. Grâce à la virtualisation des serveurs, plusieurs systèmes d'exploitation peuvent fonctionner sur un seul serveur physique en tant que machines virtuelles, chacune ayant accès aux ressources informatiques du serveur sous-jacent [11].

Avec la virtualisation, tous les logiciels, les pilotes et le système d'exploitation sont stockés sur des serveurs, au lieu de devoir être installés sur la machine de chaque utilisateur. Cela signifie qu'un grand nombre d'utilisateurs peuvent être administrés de manière centralisée, toutes les informations et données relatives aux utilisateurs étant conservées dans le centre de données. Cela réduit le temps et les coûts nécessaires à l'administration d'une grande infrastructure. L'environnement des utilisateurs reste inchangé ; ils bénéficient de la

même expérience qu'avec un ordinateur normal. Le responsable informatique (technologies de l'information), quant à lui, bénéficie d'une plus grande efficacité.

Le modèle de virtualisation des postes de travail permet donc aux entreprises de réaliser d'importantes économies de temps et d'argent dans la gestion de leur infrastructure informatique [11].

1.4 Le SDN

Le SDN est l'acronyme du Software Defined Networking, ou bien réseau définie par logiciel. Selon [12], Le SDN est un ensemble de techniques visant à faciliter l'architecture, la livraison et l'opération de services réseaux de manière déterministe, dynamique et pouvant être déployé à grande échelle.

L'Open Networking Fondation [1] [7] définit quant à elle le SDN comme étant une architecture qui sépare le plan de contrôle du plan de données, et unifie les plans de contrôle de plusieurs périphériques dans un seul software de contrôle externe appelé « Contrôleur », qui voit le réseau dans sa totalité pour gérer l'infrastructure via des interfaces de communications appelées APIs. Le contrôleur en question fait abstraction de la couche physique pour les applications qui communiquent en langage développeur, permettant la programmation du réseau (voire figure ci-dessous).

L'objectif principale du réseau défini par logiciel (SDN) est de réduire les coûts par la virtualisation, l'automatisation et la simplification [13].

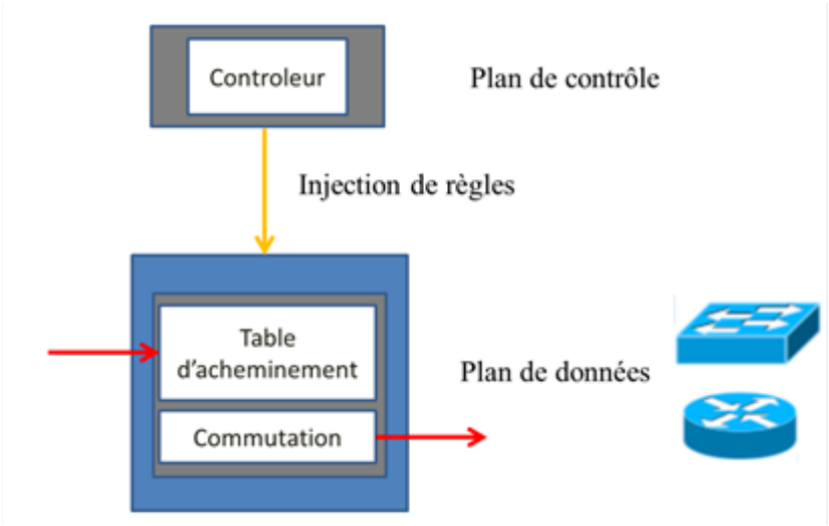

FIGURE 1.2 – Architecture bref du SDN [2]

1. L'ONF est un consortium à but non lucratif dirigé par des opérateurs. Il utilise un modèle d'entreprise open source visant à promouvoir la mise en réseau via un réseau défini par logiciel et à standardiser le protocole OpenFlow et les technologies associées

1.5 Architecture du SDN

Le SDN est composée principalement de trois couches [14], et est composé de deux interfaces de communications ou API qui permettent au contrôleur d'interagir avec les autres couches du réseau SDN [15] [?]. La notation Nord/Sud/Est/Ouest sont leurs attribuées en fonction de la position de la couche avec laquelle communique le contrôleur dans la hiérarchie de l'architecture [4]. La figure suivante illustre l'architecture du SDN, et les détaille seront donnés dans les sections qui suivent.

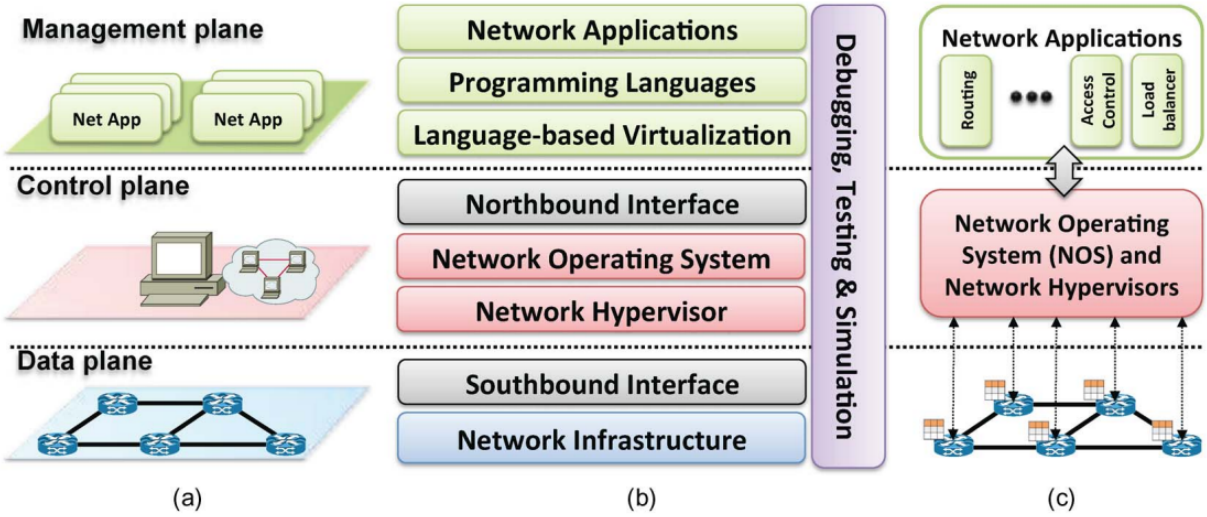

FIGURE 1.3 – Architecture du SDN [3]

1.5.1 Le plan de transmission (Data plane)

Appelée aussi "plan de données", elle est composée des équipements d'acheminement (du matériel) tels que les commutateurs ou les routeurs, d'autres exemples incluent des éléments de réseau qui peuvent fonctionner à une couche supérieure à IP (comme les pare-feu) ou inférieure à IP (comme les commutateurs de couche 2 et les éléments de réseau optique ou à micro-ondes). Son rôle principal est de transmettre les données selon les politiques configurées au niveau du plan de contrôle. Ce périphérique réseau est une entité qui reçoit des paquets sur ses ports et exécute une ou plusieurs fonctions réseau sur ces perchloriques. Par exemple, le périphérique réseau peut transmettre un paquet reçu, l'abandonner, modifier l'entête du paquet (ou la charge utile), transmettre le paquet, etc. C'est une agrégation de plusieurs ressources telles que des ports, une unité centrale, de la mémoire et des files d'attente. Les périphériques réseaux sont en soi une ressource complexe, ils peuvent être mis en œuvre sous forme matérielle ou logicielle et peuvent être physiques ou virtuels.

Les plans d'acheminement et d'exploitation sont exposés via la couche d'abstraction des dispositifs et des ressources (DAL), qui peut être exprimée par un ou plusieurs modèles. Des

exemples de modèles d'abstraction du plan d'acheminement sont le modèle YANG [2] et les MIB [3] SNMP [4].

1.5.2 Le plan de contrôle (Control plane)

Le plan de contrôle est généralement distribué et est principalement responsable de la configuration du plan d'acheminement à l'aide d'une interface Control Plane Southbound Interface (CPSI) avec DAL comme point de référence. Le plan de contrôle est chargé d'indiquer au plan de transmission comment traiter les paquets du réseau.

Les fonctionnalités du plan de contrôle comprennent généralement :

o la découverte et la maintenance de la topologie.

o La sélection et l'instanciation des routes par paquets.

o Mécanismes de basculement de chemin.

1.5.3 Le plan d'application (Management plane)

Le plan d'application ou de gestion est généralement centralisé et vise à garantir que l'ensemble du réseau fonctionne de manière optimale en communiquant avec le plan de contrôle.

Les fonctionnalités du plan de gestion sont généralement initiées, sur la base d'une vue globale du réseau, et sont traditionnellement centrées sur l'homme. Cependant, ces derniers temps, les algorithmes remplacent la plupart des interventions humaines. Les fonctionnalités du plan de gestion comprennent généralement les éléments suivants :

o la gestion de la configuration.

o la gestion des pannes et de la surveillance.

En outre, les fonctionnalités du plan de gestion peuvent également inclure des entités telles que des orchestrateurs, des gestionnaires de fonctions de réseau virtuel (VNF Managers) et des gestionnaires d'infrastructure virtualisée. Ces entités peuvent utiliser des interfaces de gestion pour des ressources du plan opérationnel pour demander et fournir des ressources pour les fonctions virtuelles, ainsi que d'ordonner l'instanciation de fonctions d'acheminement virtuelles au-dessus des fonctions d'acheminement physiques.

1.5.4 Interface Sud

Les API orientées vers le sud permettent de contrôler le réseau. Ces API sont utilisées par le contrôleur pour modifier dynamiquement les règles de transfert installées dans les disposi-

2. YANG (Yet Another Next Generation) est un langage de modélisation de données pour la configuration du réseau. Il est utilisé pour exprimer la structure des données et les instances de données peuvent être exprimées en XML, JSON, etc

3. Une MIB (management information base) est une base d'informations pour la gestion du réseau

4. Le protocole SNMP (Simple Network Management Protocol) est un protocole de couche d'application basé sur le protocole IP qui permet d'échanger des informations entre une solution de gestion de réseau et tout périphérique compatible SNMP.

tifs du plan de données, à savoir les commutateurs, les routeurs, etc. Bien qu'OpenFlow soit le plus connu des protocoles SDN pour les API sud, il n'est pas le seul disponible ou en cours de développement. NETCONF [5] (normalisé par l'IETF), OF-Config (soutenu par l'ONF), Opflex (soutenu par Cisco) et d'autres sont des exemples d'interfaces sud utilisées pour la gestion des périphériques réseau. En outre, certains protocoles de routage tels que IS-IS [6], OSPF(Open Shortest Path First), BGP sont également développés en tant qu'interfaces sud dans certains contrôleurs dans le but de prendre en charge des réseaux hybrides ou d'appliquer la mise en réseau traditionnelle d'une manière définie par logiciel.

1.5.5 Interface Nord

Les API nord sont utilisées par la couche applicative pour communiquer avec le contrôleur. Elles constituent la partie la plus critique de l'architecture du contrôleur SDN. L'avantage le plus précieux du SDN provient de sa capacité à prendre en charge et à permettre des applications innovantes. Parce qu'elles sont si critiques, les API nord doivent prendre en charge une grande variété d'applications. Ces API permettent également la connexion avec des piles automatisées telles qu'OpenStack ou CloudStack utilisées pour la gestion du Cloud. Récemment, l'ONF [7] s'est concentré sur l'API SDN nord après avoir travaillé à la standardisation de l'interface sud (OpenFlow). L'ONF a créé un groupe de travail sur l'interface nord qui écrira du code, développera des prototypes et s'occupera de la création de normes. Actuellement, le protocole REST [7] semble être l'interface nord la plus utilisée et la plupart des contrôleurs l'implémentent.

1.5.6 Interface Est/Ouest

Ce sont des interfaces inter-contrôleurs, on les trouve dans les architectures distribués (Multi-contrôleurs). Ils permettent la communication entre contrôleurs pour synchroniser les états du réseau. La communication entre ces entités est généralement mise en œuvre par le biais de des protocoles de passerelle tels que BGP ou d'autres protocoles tels que le Path Computation Element (PCE). Ces messages de protocole correspondants sont généralement échangés en bande et ensuite redirigés par le plan d'acheminement vers le plan de contrôle pour un traitement ultérieur.

Ainsi en résumant toute cette section, l'architecture du SDN se concentre sur la fourniture de fonctionnalités de mise en réseau en employant un contrôleur SDN logiquement centralisé (Plan de contrôle) qui communique avec les dispositifs de réseau programmables (Plan de transmission) via l'interface "southbound" et avec les applications SDN (Plan d'application)

5. NETCONF est un protocole basé sur du XML permettant la gestion et la configuration d'équipements réseaux

6. IS-IS (Intermediate system to intermediate system) est un protocole de routage interne multi-protocoles à états de lien.

7. Representational state transfer : est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web.

via l'interface "northbound". La communication entre la couche de contrôle et la couche de transmission se fait par le protocole OpenFlow ; plus de détails sur ce protocole seront donnée dans une section ultérieure 1.8.

1.6 Composants d'une architecture SDN

1.6.1 Le commutateur SDN

Les commutateurs Ethernet et les routeurs les plus modernes contiennent des tables de flux qui sont utilisées pour effectuer des fonctions de transfert selon les couches 2,3 et 4, indiquées dans les en-têtes de paquets. Bien que chaque fournisseur ait des tables de flux différentes, il existe un ensemble commun de fonctions pour une large gamme de commutateurs et de routeurs. Cet ensemble commun de fonctions est mis à profit par OpenFlow peut être utilisé pour programmer la logique de transfert ou d'acheminement du commutateur.

Donc, par abus de langage, tout les équipements de transmission SDN sont appelés switch. Dans ce jargon le terme switch est attribué à tout équipement de transmission qui compare l'entête des paquets aux tables de flux, que la comparaison se fasse à base d'adresses MAC (Couche 2), d'adresses IP (Couche 3) ou une combinaison de plusieurs champs. La terminologie switch a été adoptée en référence à l'unique tache de ces équipements qui est la transmission [3].

Les principales fonctions des commutateurs sont les suivantes :

1 **Fonction de support du contrôle (Controle support function) :** Interagit avec la couche contrôle SDN afin de supporter la programmabilité via les interfaces ressource. Le commutateur communique avec le contrôleur et le contrôleur gère le commutateur avec le protocole OpenFlow.

2 **Fonction d'acheminement des données (Data forwarding function) :** Accepte les flux de données entrants provenant d'autres équipements de réseau et des systèmes de terminaison et les relaie sur un chemin de commutation qui a été calculé et établi à partir des règles définies par les applications SDN, passées au contrôleur et redescendues au commutateur.

Ces règles d'acheminement des données (data forwarding rules) sont présentes dans les tables d'acheminement (forwarding tables). Ces règles indiquent pour des catégories de paquet données quel doit être le prochain saut sur la route. Le commutateur peut par ailleurs modifier l'en-tête du paquet avant son acheminement, ou rejeter le paquet.

Le commutateur illustré dans la figure suivante dispose de trois ports d'entrée/sortie : Un port fournissant la communication de contrôle avec un contrôleur SDN, et deux autres ports pour les entrées et sorties de paquets de données.

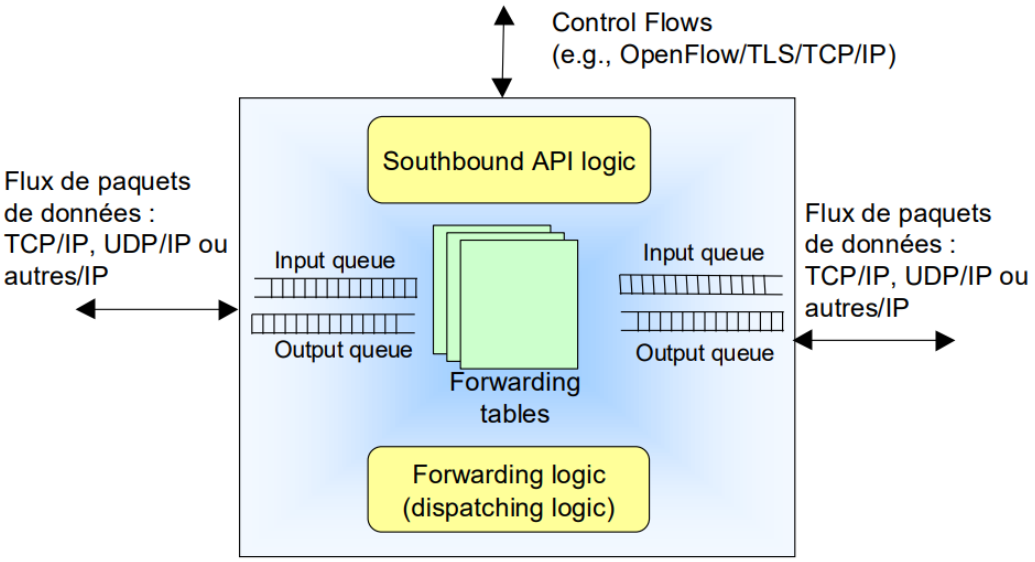

FIGURE 1.4 – Commutateur OpenFlow [3]

Le commutateur le plus utilise est le Open vSwitch [16]. C'est un commutateur virtuel multicouche de qualité de production, sous licence open source Apache 2.0. Il est conçu pour permettre une automatisation massive du réseau grâce à une extension programmatique, tout en continuant à prendre en charge les interfaces et protocoles de gestion standard (par exemple OpenFlow, NetFlow, sFlow, IPFIX, RSPAN, ...). En outre, il est conçu pour prendre en charge la distribution sur plusieurs serveurs physiques.

1.6.2 Le contrôleur SDN

Le Software Defined Networking (SDN) est une nouvelle architecture de réseau. L'un de ses composants est le contrôleur, qui est la partie intelligente du SDN. Le plan de contrôle est une partie essentielle de l'architecture SDN, il est donc très important d'accorder une attention appropriée à toute proposition ou conception d'un contrôleur SDN.

Les contrôleurs SDN communiquent avec les commutateurs qui leur sont attribués via un canal de commande et mettent en œuvre des actions réseau telles que la commutation et le routage de paquets à l'aide d'applications SDN. Cela se fait principalement en fournissant une vue globale du réseau (y compris les règles de flux au sein des dispositifs du réseau et les statistiques correspondantes) aux applications SDN, qui sont ainsi en mesure de prendre des décisions relatives au réseau.

Modes de fonctionnement des contrôleurs

Le contrôleur Openflow présente deux approches : réactive et proactive.

Reactive : Dans l'approche réactive, le premier paquet de flux reçu par le commutateur déclenche le contrôleur pour insérer des entrées de flux dans chaque commutateur OpenFlow du réseau. Cette approche présente l'utilisation la plus efficace de la mémoire de la table de flux existante, mais chaque nouveau flux entraîne un petit temps de configuration supplémentaire.

Enfin, avec la dépendance rigide du contrôleur, si un commutateur perd la connexion, son utilité est limitée.

Proactive : Dans l'approche proactive, le contrôleur pré-remplit la table de flux dans chaque commutateur. Cette approche n'a pas de temps de configuration de flux supplémentaire car la règle de transfert est définie. Maintenant, si le commutateur perd la connexion avec le contrôleur, cela ne perturbe pas le trafic. Cependant, le fonctionnement du réseau nécessite une gestion difficile.

Dans [17], ces deux modes de fonctionnement sont comparés. Le mode proactif a de meilleures performances que le mode réactif en raison du fait que les règles sont chargées dans le commutateur dès le début et non pas comme dans le mode réactif où les règles sont chargées dans le commutateur chaque fois que le commutateur reçoit un paquet sans règle correspondante dans sa table de flux. Cette comparaison mette en lumière un facteur important dans la comparaison des performances. Pour notre cas, on vas traiter les deux cas.

1.7 Modèles SDN

Il existe quatre modèles du SDN [8] :

(i) **SDN ouvert :** Les administrateurs réseau utilisent un protocole de type OpenFlow pour contrôler le comportement des commutateurs virtuels et physiques au niveau du plan de données.

(ii) **SDN piloté par API :** Au lieu d'utiliser un protocole ouvert, les interfaces de programmation d'applications contrôlent la façon dont les données circulent sur chaque terminal à travers le réseau.

(iii) **SDN de superposition :** Cet autre type de réseau software-defined exécute un réseau virtuel par dessus une infrastructure matérielle existante, créant des tunnels dynamiques vers différents Data Centers on premise et distants. Le réseau virtuel alloue la bande passante à divers canaux et affecte des terminaux à chaque canal, laissant le réseau physique intact.

(iv) **SDN hybride :** Ce modèle combine un réseau software-defined à des protocoles réseau traditionnels dans un environnement unique pour la prise en charge des différentes fonctions du réseau. Les protocoles de réseau standard continuent à orienter une partie du trafic, tandis que le SDN prend en charge une autre partie du trafic, ce qui permet aux administrateurs réseau d'introduire le SDN par étapes au sein d'un environnement legacy.

1.8 OpenFlow

Il s'agit d'un protocole standard utilisé par le contrôleur pour transmettre au commutateur des instructions qui permettent de programmer leur plan de données et d'obtenir des

informations de ces commutateurs. Il a été initialement proposé et implémenté par l'université de Stanford, et standardisé par la suite par l'ONF.

Grâce à ce protocole, les contrôleurs SDN peuvent fournir une vue globale d'un réseau, y compris les statistiques du réseau, aux applications SDN, tandis que ces applications peuvent reprogrammer le réseau sur la base des informations fournies. Par conséquent, la reprogrammation du réseau implique l'ajout, la modification ou la suppression de règles de flux [?].

1.8.1 Architecture Openflow :

L'architecture openflow est l'implémentation réelle des réseaux SDN, elle est basée principalement sur trois composantes : le plan de données, qui est composé des switches openflow ; le plan de contrôle, constitué par des contrôleurs OpenFlow ; une chaîne sécurisée qui permettent aux commutateurs de se connecter au plan de contrôle. Un commutateur openflow doit contenir un ou plusieures tables de flux, ces tables de flux contiennent plusieurs champs d'entrées qui correspondent à des règles a appliquer pour les commutateurs. La figure suivante illustre le protocole OpenFlow.

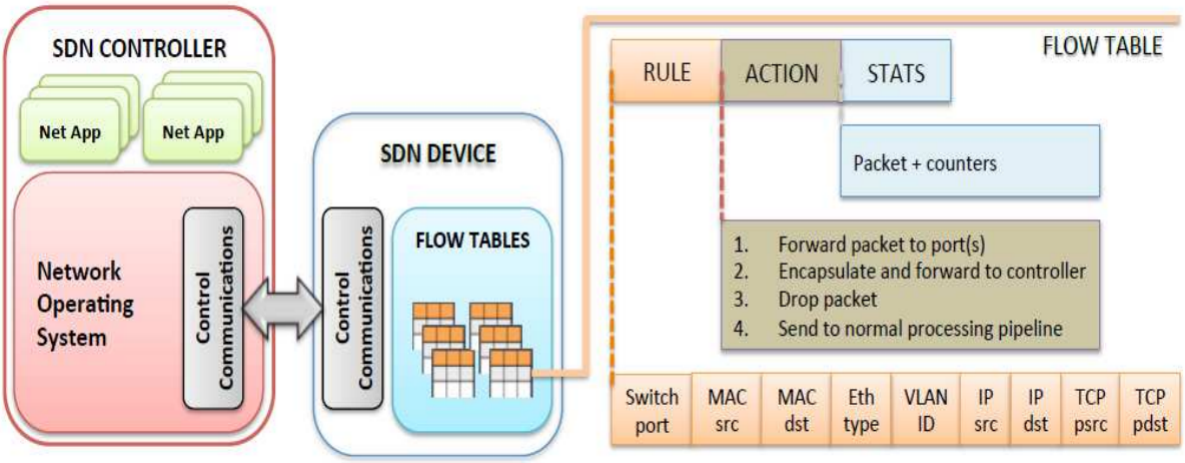

FIGURE 1.5 – Architecture OpenFlow

— **Tables de flux :** les commutateurs Openflow sont composés d'un pipeline de tables de flux. Celui-ci permet la gestion de l'acheminement des paquets, le mécanisme consiste à cumuler les actions de chaque table de flux pour les exécuter à la sortie du pipeline.

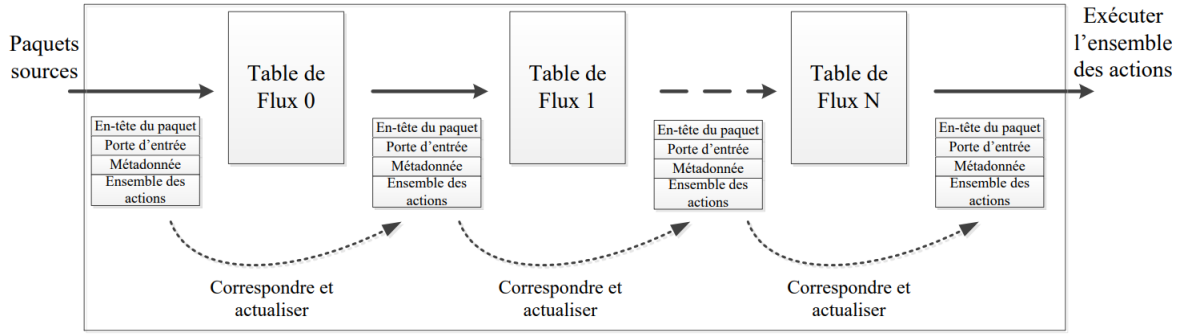

FIGURE 1.6 – Tables de flux OpenFlow

— **Champs d'entrées :** où chaque table de flux est constituée principalement des trois champs suivants :

— **L'En-tête de paquet :** il définit le flux de données. La figure suivante montre quelques uns des champs proposés par le protocole.

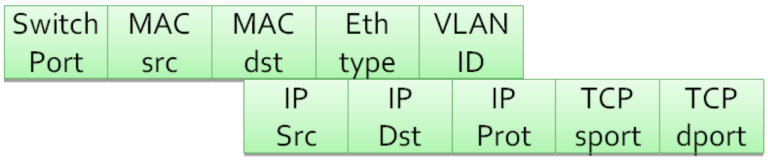

FIGURE 1.7 – En-tête du packet OpenFlow

— **Les Compteurs :** sont réservés à la collecte des statistiques de flux. Les compteurs peuvent être maintenus pour chaque table de flux, entrée de flux, port et file d'attente.

— **L'Action :** spécifie comment les paquets d'un flux seront traités. Il existe des actions obligatoires que doit supporter tout commutateur OpenFlow et des actions OpenFlow que peut supporter un commutateur OpenFlow.

Voici les actions **obligatoires** :

Forward : Les commutateurs OpenFlow doivent supporter l'acheminement de packet aux ports physiques ainsi qu'aux ports virtuels.

Encapsule and forward to controller Encapsuler et envoyer le paquet vers un contrôleur à travers le canal sécurisé. Cette action est principalement utilisée lorsqu'un paquet d'un flux est reçu pour la première fois. Le contrôleur peut décider si une nouvelle entrée doit être ajoutée dans la table, ou si le paquet doit être supprimé.

Drop : Supprimer tous les paquets appartenant à un flux est aussi une action qui peut être utilisée pour une raison de sécurité pour mettre fin à une attaque par exemple.

Les actions **optionnelles** portent sur la modification de certains champs dans le paquet (Modification du VLAN ID, Réécriture de l'adresse MAC, recalcule du TTL ... etc.).

1.8.2 Messages OpenFlow

Le protocole OpenFlow utilise le protocole TCP et le port 6633. En option, la communication peut utiliser un canal sécurisé basé sur TLS. Le protocole OpenFlow supporte trois types de messages [18] :

1 **Les messages de contrôleur à commutateur**
Ces messages sont envoyés uniquement par le contrôleur aux commutateurs, et ils remplissent les fonctions de configuration des commutateurs, échangent des informations sur les capacités des commutateurs et gèrent également la table de flux. Ils peuvent être

représentés en cinq sous-catégories : switch configuration, command from controller, statistics, queue configuration, et barrier.

2 **Messages asynchrones**
Ces messages sont envoyés par le commutateur au contrôleur pour annoncer des changements dans le réseau et l'état du commutateur. Ces messages sont : Packet-in, Flow-removed, Port-satus et Role-status.

3 **Messages symétriques**
Ces messages sont envoyés dans les deux sens pour signaler les problèmes de connexion entre le contrôleur et le commutateur. Ces messages sont : Hello, Echo, et Error.

1.8.3 Fonctionnement Openflow :

L'architecture OpenFlow est illustrée par la figure ci-dessous qui décrit le processus de transmission d'un paquet avec openflow. Lorsqu'un paquet arrive à un commutateur, le commutateur vérifie s'il y a une entrée dans la table de flux qui correspond à l'en-tête de paquet. Si c'est le cas, le commutateur exécute l'action correspondante dans la table de flux. Dans le cas contraire, c'est-à-dire il n'y a pas une entrée correspondante (1), le commutateur génère un message asynchrone vers le contrôleur (2) sous la forme d'un 'Packet in', puis le contrôleur décide selon sa configuration une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme d'un 'Packet out' et 'Flow mod' au commutateur (3), et enfin, la table de flux du commutateur est actualisée, pour prendre en compte la nouvelle règle installée par le contrôleur (4).

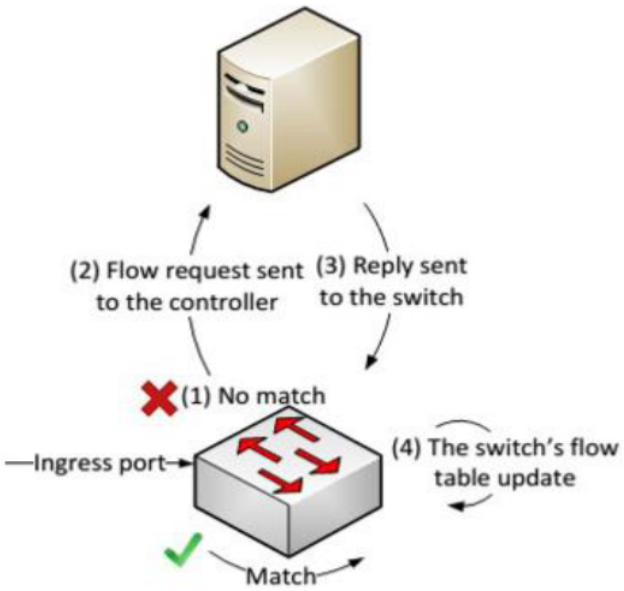

FIGURE 1.8 – Processus de transmission d'un paquet avec OpenFlow

1.9 SDN et NFV

Le réseau mobile moderne est construit sur une grande variété d'équipements propriétaires et de systèmes de contrôle des équipements correspondants. La diversité de ces équipements de réseau augmente les dépenses d'investissement et d'exploitation des fournisseurs de services, tout en causant des problèmes d'ossification du réseau. La virtualisation des fonctions de réseau (**NFV**)[8] est proposée pour résoudre ces problèmes en mettant en œuvre des fonctions de réseau sous forme de logiciels purs et généraux (fonctionnant sur des équipements commerciaux prêts à l'emploi, c'est-à-dire des machines virtuelles), sur du matériel de base et général [19]. Ainsi, elle peut réduire les coûts grâce à l'application de serveurs prêts à l'emploi pour la mise en œuvre de nouvelles fonctionnalités de réseau, qui sont capables de répondre aux besoins changeants du marché mobile [11]. NFV permet un approvisionnement, un déploiement et une gestion centralisée flexibles des fonctions de réseau virtuelles. Ainsi, son objectif est de découpler les fonctions réseau des équipements réseau, il vise donc à virtualiser toutes les ressources du réseau physique sous un hyperviseur, ce qui permet au réseau de se développer sans avoir à ajouter de nouveaux appareils.

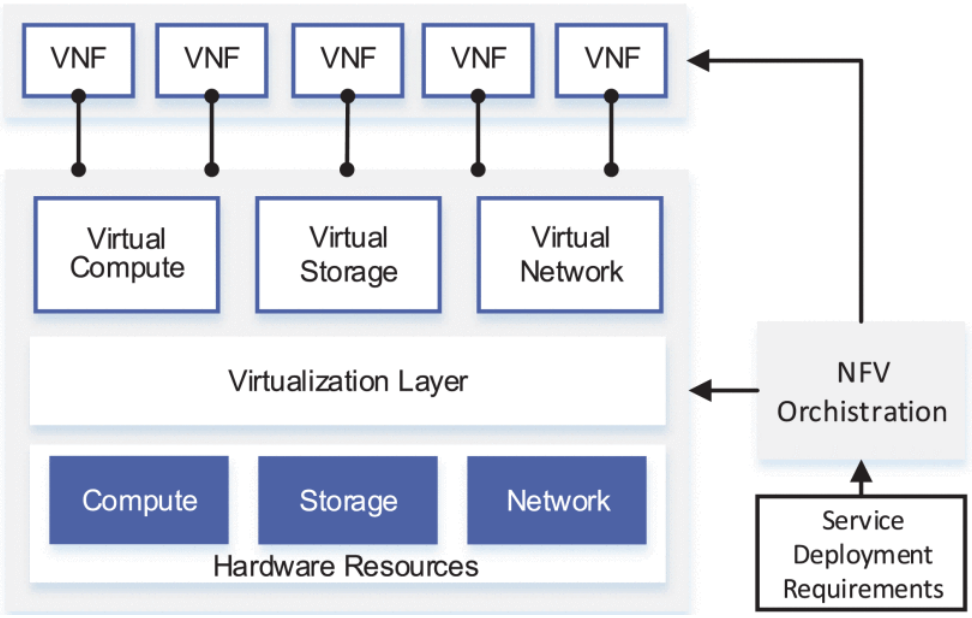

FIGURE 1.9 – Infrastructure du NFV

La principale similitude entre le réseau défini par logiciel (SDN) et la virtualisation des fonctions réseau (NFV) est qu'ils utilisent tous deux l'abstraction du réseau.

NFV et SDN sont complètement différents, mais sont complémentaires. Des concepts à l'architecture et aux fonctions du système, qui sont résumés par les aspects suivants [19] :

— NFV est un concept de mise en œuvre des fonctions de réseau de manière logicielle, tandis que SDN est un concept de réalisation d'une architecture de réseau programmable et contrôlée de manière centralisée pour fournir une meilleure connectivité.

8. NFV (Network function virtualization) ou bien en français VNF (virtualisation des fonctions réseau)

— NFV vise à réduire les coûts d'investissement et d'exploitation, ainsi que la consommation d'espace et d'énergie, tandis que SDN vise à fournir des abstractions de réseau pour permettre un contrôle et une configuration flexibles du réseau et une innovation rapide.

— NFV dissocie les fonctions de réseau du matériel propriétaire afin de réaliser un approvisionnement et un déploiement agiles, tandis que SDN dissocie le plan de contrôle du réseau du plan de transfert des données afin de fournir un contrôleur centralisé en permettant la programmabilité.

le SDN et le NFV rendent les architectures de réseau plus flexibles et dynamiques, en fournissant des fonctions de réseau de base pour le NFV, et on les contrôlant et les orchestrant pour des utilisations spécifiques, cas du SDN.

1.10 Cas d'utilisation du SDN

L'ONF est guidé par des entreprises et des fournisseurs de services de premier plan, des développeurs de systèmes et d'applications, des sociétés de logiciels et d'informatique, ainsi que des fournisseurs de semi-conducteurs et de réseaux. Cet échantillon diversifié des industries des communications et de l'informatique contribue à garantir que le SDN et les normes associées répondent efficacement aux besoins des opérateurs de réseaux dans chaque segment du marché, notamment :

— **Campus** - Le modèle de contrôle et d'approvisionnement centralisé et automatisé du SDN prend en charge la convergence des données, de la voix et de la vidéo, ainsi que l'accès à tout moment et en tout lieu, en permettant au service informatique d'appliquer des politiques de manière cohérente sur les infrastructures câblées et sans fil. De même, le SDN prend en charge le provisionnement et la gestion automatisés des ressources réseau, déterminés par les profils individuels des utilisateurs et les exigences des applications, afin de garantir une expérience utilisateur optimale dans le respect des contraintes de l'entreprise.

— **Centre de données** - Les architectures SDN facilitent la virtualisation des réseaux, ce qui permet l'hyperscalabilité du centre de données, la migration automatisée des machines virtuelles, une intégration plus étroite avec le stockage, une meilleure utilisation des serveurs, une moindre consommation d'énergie et l'optimisation de la bande passante.

— **Cloud** - Qu'il soit utilisé pour prendre en charge un environnement de cloud privé ou hybride, le SDN permet d'allouer les ressources réseau de manière très élastique, ce qui permet un approvisionnement rapide des services de cloud et un transfert plus souple vers le fournisseur de cloud externe. Grâce aux outils permettant de gérer en toute sécurité leurs réseaux virtuels, les entreprises et les unités commerciales feront de plus en plus confiance aux services en nuage.

1.11 Avantages et apport du SDN pour les réseaux

Les réseaux définis par logiciel vont changer la façon dont les ingénieurs et les concepteurs de réseaux construisent et exploitent leurs réseaux pour répondre aux besoins des entreprises. Avec l'introduction du SDN, les réseaux sont devenus des normes ouvertes, non propriétaires, et faciles à programmer et à gérer. Le SDN permettra aux entreprises et aux opérateurs de mieux contrôler leurs réseaux, de les adapter et de les optimiser afin de réduire le coût global d'exploitation du réseau. Certains des principaux avantages du SDN peuvent être résumés comme suit [13] :

- **Simplicité de la gestion du réseau :** Avec le SDN, le réseau peut être considéré et géré comme un nœud unique, ce qui permet de transférer les tâches complexes de gestion du réseau par défaut vers des interfaces plus faciles à gérer avec une visibilité à l'échelle du réseau, qui permet une surveillance simplifiée de son ensemble.
- **Déploiement rapide des services :** Les nouvelles fonctionnalités et applications peuvent être déployées rapidement, en quelques heures au lieu de plusieurs jours.
- **Configuration automatisée :** Les tâches de configuration manuelles telle que l'attribution de VLAN et la configuration de la QoS peuvent être effectuées automatiquement.
- **Virtualisation du réseau :** Depuis que la virtualisation des serveurs et du stockage a été déployée plus qu'auparavant, les réseaux peuvent bénéficier du SDN pour être également virtualisés.
- **Réduction des coûts d'exploitation :** En bénéficiant de l'automatisation du déploiement du réseau, un changement sur le réseau n'a jamais été aussi facile, ce qui réduit les coûts d'exploitation du réseau.

Le SDN a connu aussi plusieurs défis que ce soit sur le plan de données que sur le plan de contrôle, les principaux défis de SDN au niveau de son plan de contrôle, qui comprend la performance, la scalabilité, la sécurité, et la fiabilité. Des solutions sont proposées et résumées dans l'article [20].

1.12 Conclusion

Dans ce chapitre, nous avons introduit les concepts clés de départ, utiles à la compréhension du projet dans sa globalité. Les applications SDN montrent l'intérêt qu'a suscité SDN auprès des chercheurs et des industriels dans le domaine des réseaux. Cet intérêt est dû à la simplification de l'architecture des équipements réseaux et à l'abstraction du réseau. À travers les interfaces de communication fournies par le contrôleur, les développeurs peuvent communiquer avec le réseau et proposer de nouveaux services. Alors que le concept de SDN offre plusieurs avantages, il soulève cependant de nouveaux défis de performances surtout au niveau du plan de contrôle puisqu'il concentre toute la complexité et l'intelligence du réseau. Dans le chapitre suivant, on va présenter les outils à utilisé pour mettre en œuvre une architecture SDN, pour ensuite en faire un choix pour cas notre étude.

Chapitre 2

Outils SDN et Travaux connexes

2.1 Introduction

Souhaitons mettre en œuvre une architecture SDN, et vue la découverte de plusieurs environnements de virtualisation (hyperviseurs), d'émulation, ainsi que de contrôle. Dans ce chapitre nous allons présenter les outils les plus utilisés pour réaliser une architecture SDN, pour ensuite détailler ceux choisit pour notre projet.

2.2 Hyperviseurs

La virtualisation a permis de réduire les investissements en matériel et en infrastructure tout en facilitant la migration et la dématérialisation. Les méthodes modernes de virtualisation apportent une grande flexibilité et une grande fiabilité.

L'utilisation d'un hyperviseur qui héberge plusieurs machines virtuelles présente plusieurs avantages tel que la rapidité, l'efficacité et la flexibilité [8]. Un hyperviseur, également connu sous le nom de moniteur de machine virtuelle ou VMM, est un logiciel qui crée et exécute des machines virtuelles (VM). Un hyperviseur permet à un ordinateur hôte de prendre en charge plusieurs machines virtuelles invitées en partageant virtuellement ses ressources, telles que la mémoire et le traitement. Les hyperviseurs les plus connue sont : Vmware, Virtual-Box, Hyper-V et Proxmox.

Selon l'étude de [21], dont le but était d'évaluer l'impact des charges d'E/S sur les performances de la RAM, et de déterminer la configuration de la RAM qui offre des performances optimales pour les deux hyperviseurs Vmware 5.1 et Proxmox 5.3. Les résultats était que Proxmox est performant lorsque la taille du bloc est petite, pour les systèmes et les services qui nécessitent des blocs de données de grande taille, VMware est le plus adapté à la gestion des charges de travail volumineuses. Une autre étude [22] pour comparer les performances du VMware Workstation et Hyper- V à montré que le premier est plus performant à tous les égards, tandis que son avance se réduit en termes de vitesse du disque dur et de vitesse d'E/S de la RAM sous l'environnement Windows 10. Hyper-V ne montre pas une performance de calcul de CPU virtuel significativement plus efficace que VMware Workstation, que ce soit dans l'environnement. Pour Virtuel Box [23], Virtual Box convient aussi bien à un usage personnel qu'à des environnements professionnels moins exigeants, et VMWare fournit une solution complète de virtualisation.

Finalement, pour installer les machines virtuelles, nous avons opté pour l'hyperviseur VMware® Workstation 16 Pro version 16.2.2 build-19200509.

2.3 Outils émulateur du réseau

Mininet [24] est un émulateur de réseau open source disponible gratuitement. Il s'agit d'une plateforme SDN populaire que les chercheurs utilisent en raison de sa flexibilité, de sa disponibilité et de sa simplicité. En outre, elle est entièrement consacrée à l'architecture

OpenFlow. Dans Mininet, l'utilisateur peut créer, personnaliser et partager diverses topologies qui se composent de contrôleurs, de commutateurs, de routeurs, de liens et d'hôtes finaux, et effectuer des tests sur ceux-ci très facilement. Mininet contient des topologies communes prédéfinies telles que les topologies simples, linéaires et arborescentes. De plus, des topologies personnalisées peuvent être créées. Mininet peut être connecté à un contrôleur distant, et il existe également des contrôleurs locaux. La documentation de Mininet est disponible sur le site officiel `mininet.org`.

EstiNet est un simulateur/émulateur de réseau issu de NCTUns. L'environnement de simulation de réseau d'EstiNet comprend la couche physique, la couche de contrôle d'accès au support, la couche réseau, la couche transport et la couche application. En outre, l'interface graphique conviviale d'EstiNet offre aux utilisateurs un moyen pratique de construire un réseau simulé et un affichage visuel pour l'observation des résultats de simulation et le débogage. La documentation d'Estinet est disponible sur ce site `estinet.com`.

Selon l'étude de l'article [25], qui fait une comparaison, une évaluation, l'exactitude, la performance et l'évolutivité de l'émulateur EstiNet OpenFlow et de l'émulateur Mininet OpenFlow sur un ensemble de réseaux de grille, les commentaires de l'étude était qu'EstiNet génère des résultats de performance corrects, alors que les résultats de performance de Mininet sont indignes de confiance ; de plus, EstiNet est beaucoup plus évolutif que Mininet lors de l'étude de grands réseaux OpenFlow.

Pour notre cas, notre choix sera sur Mininet, car c'est un émulateur gratuit (open source), contrairement à Estinet qui est un logiciel payant, et il n'offre qu'une semaine d'essai gratuit en émulant au maximum 15 nœuds.

2.4 Contrôleur

Depuis que le Software Defined Networking (SDN) a été introduit, il a conduit à la découverte de plusieurs contrôleurs, qui ont été constamment améliorés par les chercheurs pour surmonter de nombreuses contraintes.

2.4.1 Quelques contrôleurs SDN

De nombreux contrôleurs tels que cité auprès et d'autres ont été lancés.

1 **NOX [26]** a été l'un des premiers contrôleurs à mettre en œuvre le protocole OpenFlow 1.0. Ce contrôleur permet le développement en langage C++, ce qui permet un environnement asynchrone et basé sur les événements. C'est un contrôleur largement utilisé dans le milieu universitaire pour le développement d'applications, mais son activité a diminué en raison de son successeur, POX.

2 **POX [27]** Il s'agit d'une variante de NOX qui a commencé à être plus largement utilisée parce qu'elle prend en charge une vue topologique graphique et offre un support pour la virtualisation. Son langage de programmation est Python, offrant également une plateforme de développement très rapide.

3 **Floodlight [28]** Ce contrôleur est né dans un environnement plus dédié au domaine commercial et il est donc l'un des plus utilisés dans ce contexte. En outre, l'une des principales raisons de son utilisation est qu'il explore la question de l'évolutivité, étant signalé comme l'un des plus évolutifs du marché. Développé en Java, il est compatible avec les plates-formes multiples et dispose d'un système de gestion de la qualité. licence open source fournie par Apache.

4 **Beacon [29]** Ce contrôleur a été proposé par l'Université de Stanford en 2010 et est basé sur Floodlight. Comme il est basé sur Java, il prend en charge les plates-formes multiples et offre également des fonctions de multithreading et des opérations basées sur des événements.

5 **IRIS [30]** Ce contrôleur est basé sur les contrôleurs Floodlight et Beacon. Il entend non seulement améliorer ces deux contrôleurs, mais il propose l'horizontalité dans les réseaux, la haute disponibilité et la transparence en cas de défaillance, ainsi que le support multi-domaine abstrait basé sur OpenFlow.

6 **OpenDaylight [31]** OpenDaylight, ou bien en abréviation ODL. Ce contrôleur a été créé pour diffuser et promouvoir les réseaux SDN et les fonctions de réseau virtuel. Sous l'égide de la Fondation Linux, ce contrôleur permet non seulement de travailler dans un environnement professionnel, mais aussi dans un environnement académique à des usages multiples. Outre la mise en œuvre d'OpenFlow, il permet d'autres types de protocoles, ce qui en fait l'un des plus utilisés dans ce domaine. Il a été développé en Java afin de couvrir de multiples plateformes suite à sa philosophie d'ouverture à tous les utilisateurs.

7 **Ryu [32]** Tout comme OpenDayLight, Ryu permet également le multiprotocole, à la différence que sa langue est le Python. Ce contrôleur permet le contrôle d'événements, le contrôle d'applications, le traitement de messages et offre une série de fonctionnalités réutilisables pour d'autres protocoles.

8 **OpenMul [33]** Solution qui permet de contrôler les équipements en utilisant OpenFlow, OVSDB et NETCONF comme protocole. Développé à partir de zéro en C, il est très évolutif par rapport aux autres contrôleurs et se caractérise par ses hautes performances, grâce à de faibles latences et à des vitesses de transfert de données élevées entre le contrôleur et l'équipement de réseau.

9 **Maestro [34]** Contrairement aux contrôleurs mentionnés précédemment, il s'agit d'un système d'exploitation et non d'un simple contrôleur. Cette solution est très axée sur la question du parallélisme dans le contrôle du réseau pour améliorer les performances du système. En outre, ce contrôleur permet l'utilisation d'autres types de protocoles qu'OpenFlow.

10 **ONOS [35]** C'est l'acronyme de Open Networking Operating System. C'est un système d'exploitation aussi qui vise à offrir une extensibilité, une haute disponibilité et de hautes

performances aux fournisseurs de services de communication. Bien qu'il soit spécifique à ces situations, il peut également servir de contrôleur SDN. Ce contrôleur permet de contrôler le cloud afin de pouvoir introduire de nouvelles applications plus rapidement, car il n'est pas nécessaire de modifier le plan de données. En outre, sa configuration et son contrôle en temps réel rendent inutiles les protocoles de routage et/ou de commutation. Un autre détail qui fait qu'il est largement utilisé est le fait qu'il appartient à la Fondation Linux qui dispose de moyens de diffusion bien développés.

En ce qui suit, un résumé sur les caractéristiques les plus importantes des contrôleurs SDN [4].

	Prog language	Gui	Doc	Modularity	Distributed/ Centralized	Platform support	prod	Southbound APIs	Northbound APIs	Partener	Multithreading support	Open-stack support
ONOS	Java	Web Based	Good	High	D	Linux, MAC OS, And Windows	fair	OF1.0,1.3, NETCONF	REST API	ON.LAB, AT&T, Ciena, Cisco,Ericsson, Fujitsu,Huawei , Intel, NecNsf. Ntt Communication ,Sk Telecom	Y	N
Open-Day- Light	Java	Web Based	Very good	High	D	Linux, MAC OS, And Windows	fair	OF1.0,1.3, 1.4, NETCONF/ YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP	REST API	Linux Foundation With Memberships Covering Over 40 Companies, Such As Cisco, IBM, NEC	Y	Y
NOX	C++	Python + QT4	Poor	Low	C	Most Supported On Linux	fair	OF 1.0	REST API	Nicira	NOX_MT	N
POX	Python	Python + QT4	Poor	Low	C	Linux, MAC OS, And Windows	high	OF 1.0	REST API	Nicira	N	N
RYU	Python	Yes	Fair	Fair	C	Most Supported On Linux	high	OF 1.0,1.2, 1.3, 1.4, NETCONF, OFCONFIG	REST For South bound	Nippo Telegraph And Telephone Corporation	Y	Y

FIGURE 2.1 – Comparaison des contrôleurs sélectionnés en fonction de leurs caractéristiques [4]

Beacon	Java	Web Based	Fair	Fair	C	Linux, MAC OS, And Windows	fair	OF 1.0	REST API	Standford University	Y	N
Maestro	Java	-	Poor	Fair	C	Linux, MAC OS, And Windows	fair	OF 1.0	REST API	RICE, NSF	Y	N
Flood-Light	Java	Web/ Java Based	Good	Fair	C	Linux, MAC OS, And Windows	fair	OF 1.0, 1.3	REST API	Big Switch Networks	Y	N
Iris	Java	Web Based	Fair	Fair	C	Linux, MAC OS, And Windows	fair	OF 1.0, 1.3, OVSDB	REST API	ETRI	Y	N
MUL	C	Web Based	Fair	Fair	C	Most Supported On Linux	fair	OF 1.4,1.3, 1.0, OVSDB, OFCONFIG	REST API	Kulcloud	Y	Y
Runos	C++	Web Based	Fair	Fair	D	Most Supported On Linux	fair	OF 1.3	REST API	ARCCN	Y	N
Lib-Fluid	C++	-	Fair	Fair	-	Most Supported On Linux	fair	OF 1.0,1.3	-	ONF	Y	N

FIGURE 2.2 – Comparaison des contrôleurs sélectionnés en fonction de leurs caractéristiques [4]

2.4.2 Critère de notre choix des contrôleurs à étudiés

Parmi les contrôleurs cités juste avant, notre choix pour l'étude sera sur les contrôleurs OpenDaylight (ODL) et ONOS. Les critères de ce choix sont les suivants :

1 **Le langage de programmation :** ODL et ONOS sont écrits avec le langage JAVA. Le langage Java est généralement plus rapide et plus efficace que Python car c'est un langage compilé, de plus il ne dépend pas de la plateforme utilisée contrairement à C et C++. Ce que confirme l'étude. [36], qui dit que les contrôleurs basés sur Python ne peuvent pas répondre aux hautes performances et à la faible latence des grands réseaux, alors que les contrôleurs basés sur Java et C peuvent offrir une meilleure évolutivité et de meilleures performances.

2 **La modularité :** De manière générale, la modularité est le degré auquel les composants d'un système peuvent être séparés et recombinés, souvent au bénéfice de la flexibilité et de la variété d'utilisation. Le concept de modularité est utilisé principalement pour réduire la complexité en décomposant un système en divers degrés d'interdépendance et d'indépendance à travers, et "cacher la complexité de chaque partie derrière une abstraction et une interface". ODL et ONOS sont basés sur l'initiative Open Services Gateway (OSGi) et sont très modulaires.

3 **Fonctionnalités :** ONOS et ODL sont les contrôleurs les plus riches en fonctionnalités parmi les projets open-source actuels.

4 **L'interface API :** Les deux projets présentent l'avantage d'une interface graphique assez conviviale pour les développeurs d'applications, avec une grande communauté de contributeurs et une documentation étendue.

.

2.4.3 OpenDaylight

Selon le site officiel d ' ODL, OpenDaylight est un projet open source soutenu par IBM, Cisco, Juniper, VMWare et plusieurs autres grands fournisseurs de réseaux. OpenDaylight est une plateforme de contrôleur SDN implémentée en Java. C'est une plat-forme modulaire permettant de personnaliser et d'automatiser des réseaux de toutes tailles et de toutes échelles. Il fournit l'abstraction, la programmabilite et l'ouverture qui ouvrent la voie à une infrastructure intelligente, définie par logiciel. ODL fournit une plate-forme commune flexible qui sous-tend une grande variété d'application.

La structure d'OpenDaylight (figure suivante) s'inspire de la nature de l'environnement SDN, la communication vers l'extérieur est basée sur des API et le fonctionnement du contrôleur en interne est géré par la plateforme OSGi (Open Service Gateway initiative) montée sur un conteneur Karaf, qui implémente un modèle de composants complètement dynamique. Autrement dit les l'OSGi gère les applications internes alors que l'API gère les applications externes.

OpenDaylight intègre une couche d'abstraction de services (SAL), qui relie les plugins de protocole et les modules de fonction de réseau de services. Initialement, OpenDaylight

utilisait une approche APIDriven SAL (AD-SAL), mais utilise maintenant Model-Driven SAL (MD-SAL), car AD-SAL était inefficace. MD-SAL fournit une approche standard pour le développement de plugins et permet l'unification entre les API nord et sud. MD-SAL utilise YANG comme langage de modélisation pour décrire tous les services du réseau. YANG facilite le développement des modules de base, des composants de plate-forme et des applications Northbound en fournissant des structures et des types intégrés, tels que des listes et des conteneurs. C'est un mécanisme d'échange de données et d'adaptation entre les modèles YANG représentant les dispositifs et les applications du réseau. Ce sera l'approche adoptée pour le projet et nous allons nous y intéresser par la suite à travers les différentes méthodes d'exploitation de l'interface OFM.

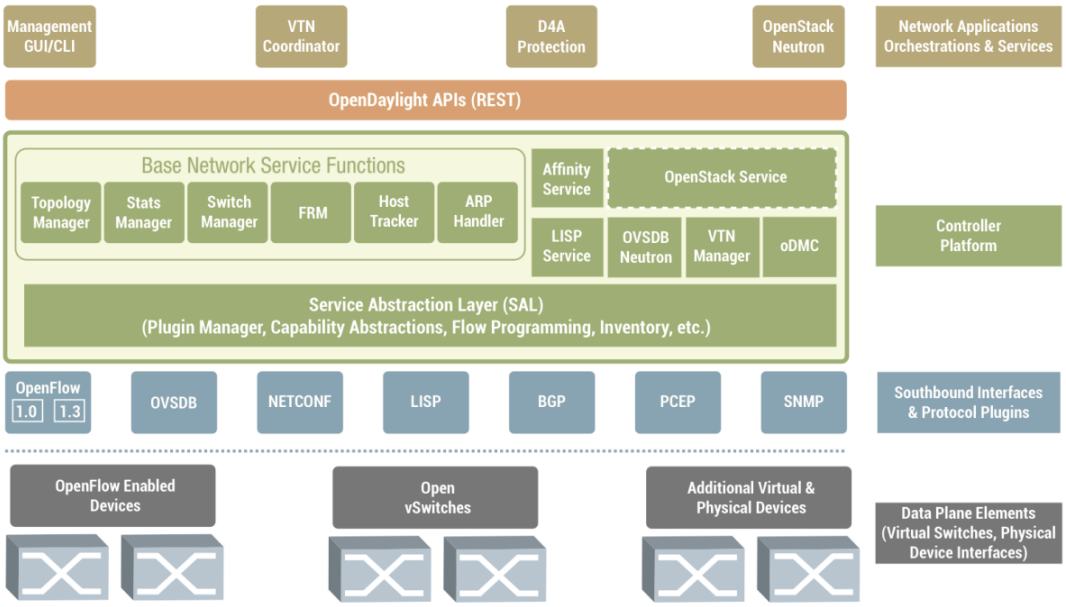

FIGURE 2.3 – Architecture d'ODL [5]

OpenFlow Manager : (OFM) est une application développée pour fonctionner au-dessus d'ODL afin de visualiser les topologies OpenFlow (OF), programmer les chemins OF et rassembler les statistiques OF. La figure suivante décrit l'architecture des composants de l'OFM.

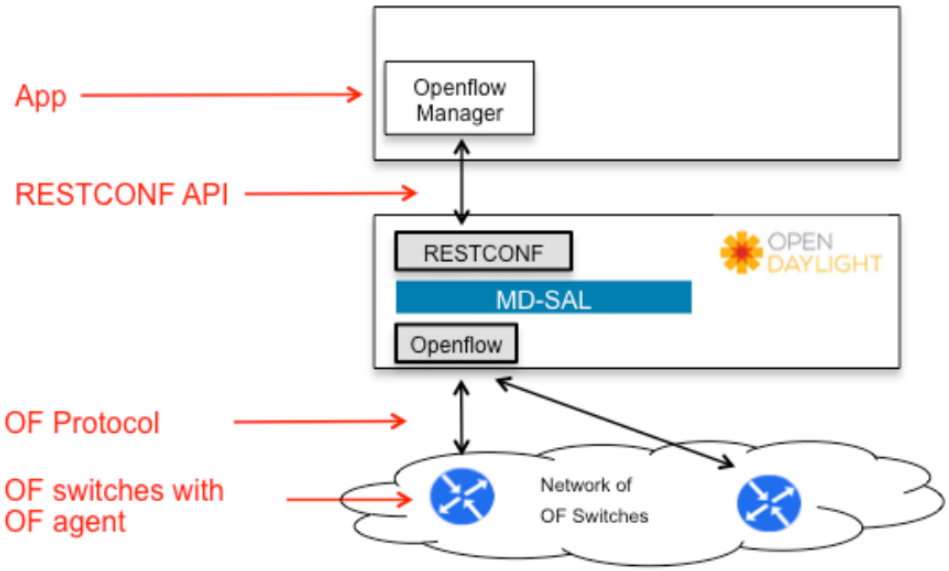

FIGURE 2.4 – architecture des composants de l'OFM

Elle est faite pour une architecture MD-SAL, offrant la possibilité de créer des flux aisément grâce à une interface graphique accueillante. L'application communique avec les modèles YANG se trouvant dans le contrôleur à travers l'interface RESTCONF. L'approche est simple, en cliquant sur une case pour choisir l'action ou les champs de correspondance par exemple, on génère une ligne de code en Json, une fois l'opération validé l'application envoie la requête via RESTCONF au contrôleur où le SAL traduira le modèle au langage de l'équipement en question.

2.4.4 ONOS

Selon le site officiel d'ONOS®, est l'acronyme de Open Network Operating System (système d'exploitation de réseau ouvert) est le principal contrôleur SDN open source permettant de créer des solutions SDN/NFV de nouvelle génération. ONOS fournit le plan de contrôle pour un réseau défini par logiciel (SDN), en gérant les composants du réseau, tels que les commutateurs et les liaisons, et en exécutant des programmes ou des modules logiciels pour fournir des services de communication aux hôtes finaux et aux réseaux voisins. C'est une plateforme de contrôleur SDN implémentée en Java.

ONOS a été conçu pour répondre aux besoins des opérateurs souhaitant construire des solutions de qualité opérateur qui tirent parti de l'économie du matériel silicium marchand en boîte blanche tout en offrant la flexibilité nécessaire pour créer et déployer de nouveaux services réseau dynamiques avec des interfaces programmatiques simplifiées. ONOS prend en charge à la fois la configuration et le contrôle en temps réel du réseau, éliminant ainsi la nécessité d'exécuter des protocoles de contrôle de routage et de commutation à l'intérieur de la structure du réseau. En transférant l'intelligence dans le contrôleur en cloud ONOS, l'innovation est permise et les utilisateurs finaux peuvent facilement créer de nouvelles applications réseau sans avoir à modifier les systèmes de voies de données.

La plateforme ONOS comprend :

— Une plateforme et un ensemble d'applications qui agissent comme un contrôleur SDN extensible, modulaire et distribué.

— Une gestion, une configuration et un déploiement simplifiés de nouveaux logiciels, matériels et services.

— Une architecture scale-out pour fournir la résilience et l'évolutivité nécessaires pour répondre aux rigueurs des environnements de production des opérateurs.

Il s'appuie sur quatre objectifs majeurs : la modularité, la souplesse de configuration, l'isolation des sous-systèmes et l'agnosticisme protocolaire. ONOS utilise une API basée sur l'intention qui capture les directives de politique pour contrôler la fonction réseau [37]. Son architecture est illustrée dans la figure qui suit :

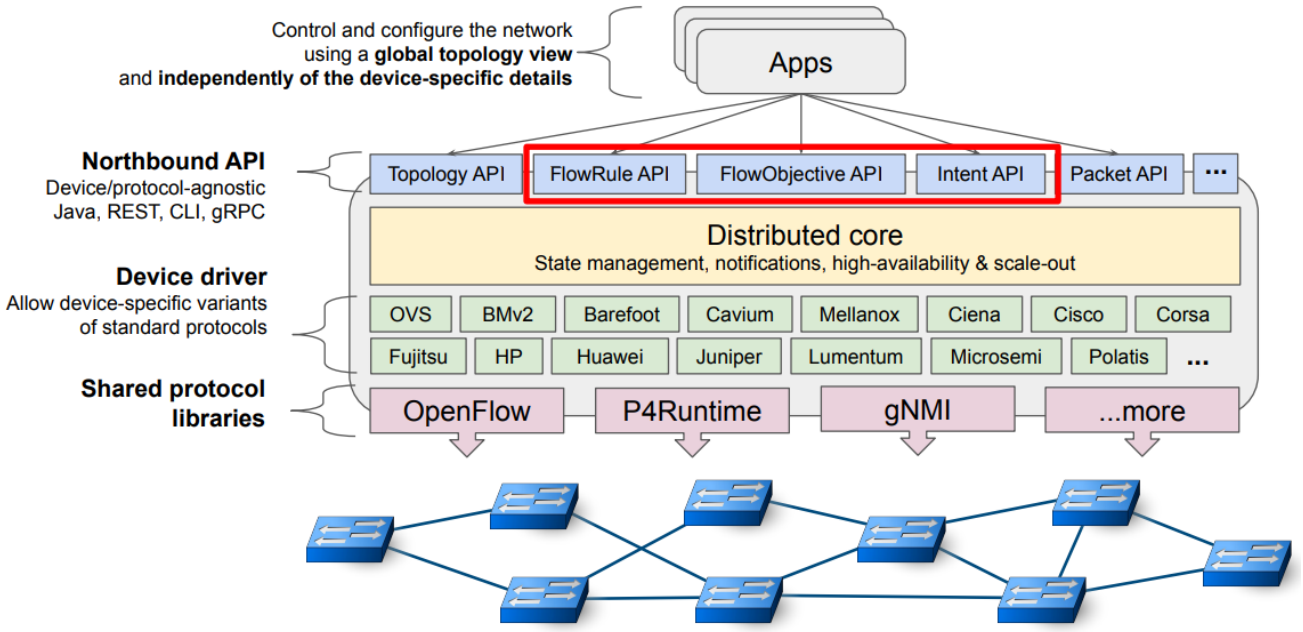

FIGURE 2.5 – Archirecture d'ONOS

ONOS est écrit de manière à utiliser Apache Karaf comme cadre OSGi et utilise Swagger pour générer automatiquement la documentation de l'API REST.

2.5 Conclusion

Après avoir présenté les outils nécessaires à la mise en œuvre d'une architecture réseau SDN, nous avons opté à l'utilisation de Mininet comme émulateur pour implémenter notre réseau avec les différents composants (nœuds) à travers l'exploitation de l'hyperviseur Vmware. L'étude et l'évaluation du réseau implémenté sont basées sur les deux contrôleurs choisis. Dans ce qui suit nous allons procéder à l'implémentation du réseau du type Data Center avec les deux contrôleurs sélectionnés : OpenDaylight et ONOS.

Chapitre 3

Implémentation, tests et résultats

3.1 Introduction

Dans ce chapitre nous allons présenter la topologie à implémenter sous Mininet, ensuite la connecter dans un premier lieu au contrôleur ODL, puis au contrôleur ONOS, et sur chacun des contrôleurs nous allons configurer des règles de flux afin d'avoir le comportement désiré de notre topologie à travers les API correspondante. Puis à la fin de ce chapitre nous allons faire nos tests de performance.

3.2 Configuration du banc d'essai

Pour pouvoir réaliser un réseau SDN virtuel (pour notre cas), nous émulons dans un serveur physique notre topologie en utilisant l'émulateur Mininet dans sa version 2.3.0. Les caractéristiques de notre serveur physique sont, Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz 2.70 GHz avec 08 GB de RAM. Chaque domaine du réseau émulé utilise deux machines virtuelles, une pour le contrôleur SDN et une autre pour le réseau. La VM du réseau local (Mininet) est connecté au contrôleur (une première fois avec ODL, et une autre avec ONOS). Le protocole sud de communication entre les OVS et les contrôleurs est OpenFlow 1.3.

La configuration de nos machines virtuelles (VMs) est la suivante :

— **Mininet :** Nous avons utilisé une machine virtuelle Mininet pré-construite, disponible sur le site `mininet.org`. Toutes les versions de l'émulateur sont disponibles sur (Github, 2017), sous formes de VM préparées, il suffit d'importer le fichier .OVF sur l'hyperviseur. La version installée dans notre cas est la 2.2.2, sur un OS ubuntu serveur 14.04.4 amd 64. Par défaut la machine est configurée avec 1Gb de RAM, 1 Processeur et 8Gb de stockage max. Nous n'avons pas jugé utile de changer cette configuration.

— **OpenDayLight : Carbon version 0.6.4**

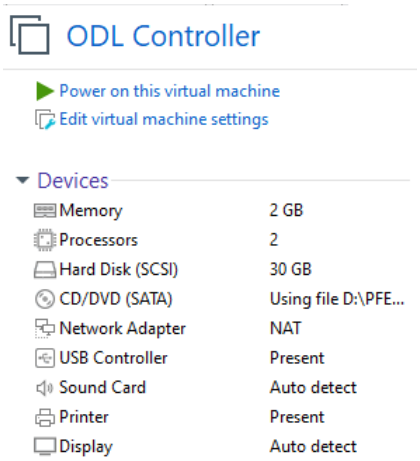

FIGURE 3.1 – Configuration de la VM du controlleur ODL

— **ONOS : X-Wing (LTS) v2.0.0 :**

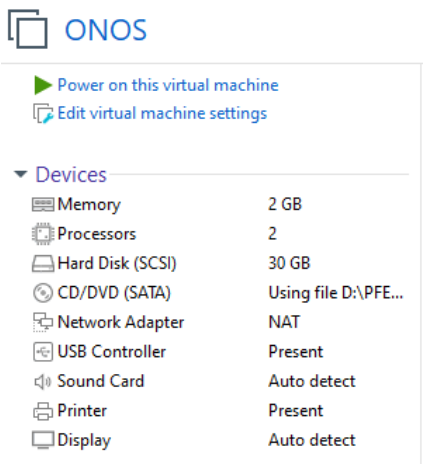

FIGURE 3.2 – Configuration de la VM du controlleur ONOS

3.3 Topologie du réseau

3.3.1 Topologie DATA CENTER proposée

La topologie à configurer est adaptée à la structure de l'entreprise SONATRACH qui est la topologie Data Center.

Pour notre exemple, nous avons pris le cas d'une entreprise composée de 4 départements. Cette architecture réseau nécessite l'utilisation de 4 switchs d'accès sur lesquelles seront branchés les différents hôtes.

Le passage vers le cœur du réseau pour avoir accès à la ressource commune (application, Cloud, internet ...) du réseau devra se faire via l'un des 2 commutateurs de distribution. Ceci fournit aux paquets une route de secours en cas de panne et offre une redondance considérable. Les 2 cores commutateurs C1 et C2 devraient être reliés aux ressources externes par des liens de très large bande passante, vu que tout le trafic nord et sud du réseau (hôte vers ressources) y sera acheminé. L'architecture ciblée est illustrée dans la figure suivante :

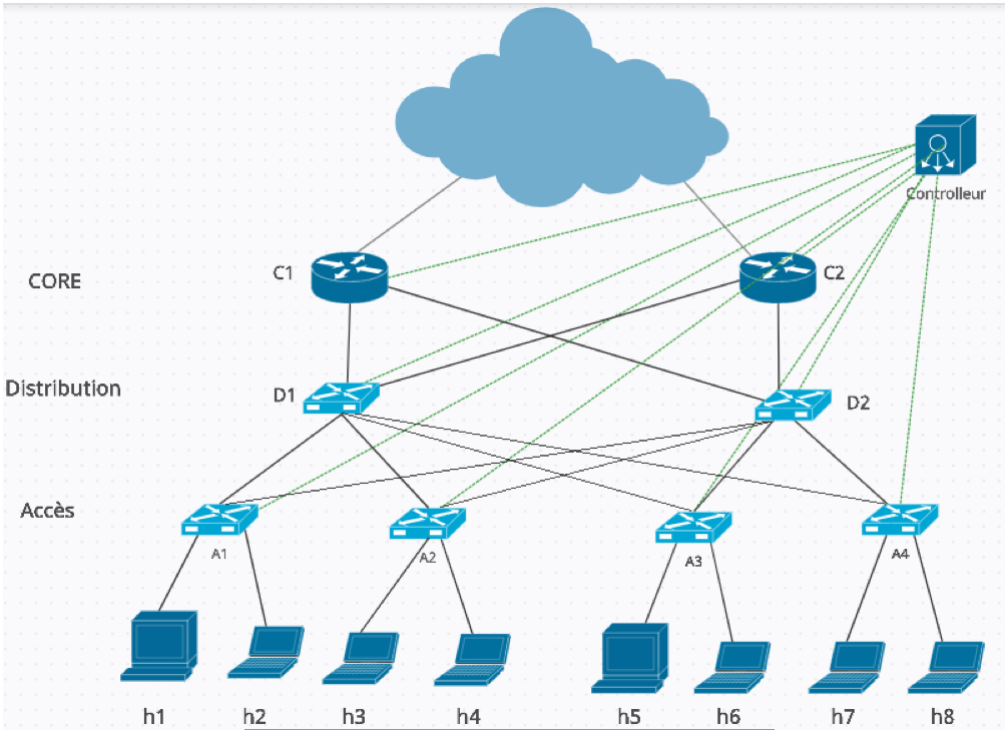

FIGURE 3.3 – Topologie proposée du réseaux

La configuration de base des différents hôtes est donnée par la table suivante :

TABLE 3.1 – Informations des hôtes

Hotes	Adresse MAC	Adresse IP
h1	00 :00 :00 :00 :00 :01	192.168.10.1
h2	00 :00 :00 :00 :00 :02	192.168.10.2
h3	00 :00 :00 :00 :00 :03	192.168.10.3
h4	00 :00 :00 :00 :00 :04	192.168.10.4
h5	00 :00 :00 :00 :00 :05	192.168.10.5
h6	00 :00 :00 :00 :00 :06	192.168.10.6
h7	00 :00 :00 :00 :00 :07	192.168.10.7
h8	00 :00 :00 :00 :00 :08	192.168.10.8

3.3.2 Comportement de la topologie et règles des flux

Ici nous allons résumer le comportement du réseau en précisant, les trafics bloqués par les paquets qui seront transmis :

1 Le commutateur A4 est isolé du reste du réseau, les hôtes qui y sont connectés ne communiquent qu'entre eux. Ceci peut être achevé par la simple assignation d'un VLAN, mais dans notre cas nous avons opté pour des règles pare-feu de couche 2, bloquant tous les paquets allant et venant du sous réseau auxquelles appartient les hôtes h8 et h7, sauf pour les paquets transmis entre eux, qui serons de priorité élevée, afin de laisser le trafic qu'entre h7 et h8.

2 h2 n'a pas les privilèges nécessaires pour communiquer avec h5. Les entrées pour bloquer ces paquets seront installés sur les deux commutateurs A1 et A3 concernés sous formes de régles pare-feu de couche 2 ciblant les deux adresses MAC spécifiques aux hôtes. Il est important de bloquer le trafic dans les deux sens, pour ne pas consommer de la bande passante inutilement.

3 h3 et h4 ne communiquent pas, un pare-feu de couche 2 leur sera appliqué.

Afin de suivre le comportement proposé de la topologie, nous allons installer des flux pour les Vswitch concerné. En plus des regèles introduites par défaut par le contrôleur ODL ou bien ONOS (exploitées en activant les fonctionnalitées correspondantes pour chacun des contrôleurs (voir Annexe)), les différents flux additionnel des Vswitchs sont résumés sous forme de tables de flux suivantes (Concernant les adresses MAC seul le dernier octet sera mentionné) :

Table 3.2 – Table de flux A1

ID_Flux	in_port	Srs_MAC	Dst_MAC	Action	Priorité
205	-	02	05	Drop	4000
01	-	-	-	Contrôleur	100

Table 3.3 – Table de flux A2

ID_Flux	in_port	Srs_MAC	Dst_MAC	Action	Priorité
304	A2 :1	-	04	Drop	4000
403	A2 :2	-	03	Drop	4000
02	-	-	-	Contrôleur	1000

Table 3.4 – Table de flux A3

ID_Flux	in_port	Srs_MAC	Dst_MAC	Action	Priorité
502	-	05	02	Drop	4000
03	-	-	-	Contrôleur	1000

Table 3.5 – Table de flux A4

ID_Flux	in_port	Srs_MAC	Dst_MAC	Action	Priorité
700	A4 :1	-	-	Drop	4000
718	A4 :1	-	-	Output :2	5000
800	A4 :2	-	-	Drop	4000
817	A4 :2	-	-	Output :1	5000
04	-	-	-	Contrôleur	1000

Voici quelques détails :

- **ID_Flux :** C'est un identifiant du flux approprié.
- **in_port :** C'est le port d'entrée du paquet de données.
- **Src_MAC :** Adresse MAC de l'hôte de source.
- **Dst_MAC :** Adresse MAC de l'hôte de destination.
- **Action :** Correspond à l'action a appliqué par le Vswitch, selon les exigences des entrées sorties, si la priorité est la plus élevée.
- **Priorité :** La règle qui sera appliquée par le Vswitch en cas de présence de plusieurs règles est celle qui a la priorité la plus élevée.

3.4 Création et exécution du scénario de la topologie

Nous présentons dans cette section les efforts expérimentaux pour évaluer l'expérience de mise en réseau SDN proposé sous Mininet, et son contrôle avec le contrôleur ODL une première fois, et une deuxième fois avec le contrôleur ONOS. Pour l'installation des flux sous ODL nous allons nous baser sur l'application OFM, et pour ONOS sur le REST API. La raison principale de ce choix, est la visibilité qu'offrent les deux interfaces graphiques des applications, ce qui simplifie l'édition et la suppression de flux installés. On peut installer différentes règles que ce soit de la 2ème ou la 3ème couche du modèle OSI pour chaque Vswitch.

3.5 Création de topologies sous Mininet

Pour exécuter Mininet, il faut utiliser la commande suivante : `sudo mn`. De nombreuses options sont à disposition, elles peuvent être affichées en ajoutant le flag `-h`. Mininet fournit une interface en ligne de commande qui peut être utilisée pour voir l'état du réseau et faire des tests dessus. Voici les commandes principales pour réaliser différents types de topologies à travers :

- **La ligne de commande de Mininet :** Grâce à la commande `mn` et la multitude d'options offertes par l'émulateur, il est possible de réaliser différentes formes de topologies et de réseaux virtuels. Les commandes de base de Mininet sont les suivantes :

Commande	principaux arguments	Explication
–topo	linear | single | tree	L'option permet de créer plusieurs types de topologies de base, principalement : tree (Arbre), single (topologie centralisée d'un seul switch), et linear (linéaire)
–switch	default | lxbr | ovs | ovsbr | ovsk	offre la possibilité de choisir le modèle de switch parmi ceux supportés par l'émulateur,tel que ovs=OVSSwitch default=OVSSwitch ovsk=OVSSwitch lxbr=LinuxBridge.
–controller	default | none | nox | remote | ryu	Pour connecter notre topologie à un contrôleur none=NullController, remote=RemoteController (controlleur distant en ajoutant , ip=<CONTROLLER_ IP_ ADRESS> default=DefaultController, nox=NOX ryu=Ryu.
–custom	CUSTOM	lire les classes personnalisées ou les paramètres du ou des fichiers .py.
–mac	default | lxbr | ovs | ovsbr | ovsk	définir automatiquement les MAC des hôtes.

TABLE 3.6 – Principales commandes Mininet

— **Via l'API Mininet**

L'émulateur Mininet fournit une interface de programmation Python, l'une de ces utilisations consiste en la création de topologies customisées en y définissant toutes les caractéristiques voulues des éléments du réseau, une approche pareille nous offre l'avantage d'exploiter le temps et les ressources en main de la meilleure façon possible. Le fichier .py doit être sauvegardé dans le répertoire `/mininet/custom`, de plus pour pouvoir appeler la topologie par l'option `-custom` la dernière ligne du script est primordiale. Les commandes principales du code python pour créer la topologie réseau souhaitée sont les suivantes :

build() : La méthode à surcharger pour créer des topologies personnalisées.
addSwitch() : Ajoute un switch et retourne son nom.
addHost() : Ajoute un hôte et retourne son nom.
addLink() : Ajoute un lien bidirectionnel entre les composants passés en argument.
start() : Démarre le réseau.
stop() : Arrête le réseau.

— **Via Miniedit**

Mininet fournit un outil qui permet de créer sa propre topologie de manière graphique du nom de miniedit. Cette interface s'exécute avec la commande : `sudo mininet/examples/miniedit.py`. Voici l'interface graphique de miniedit, avec un exemple de topologie :

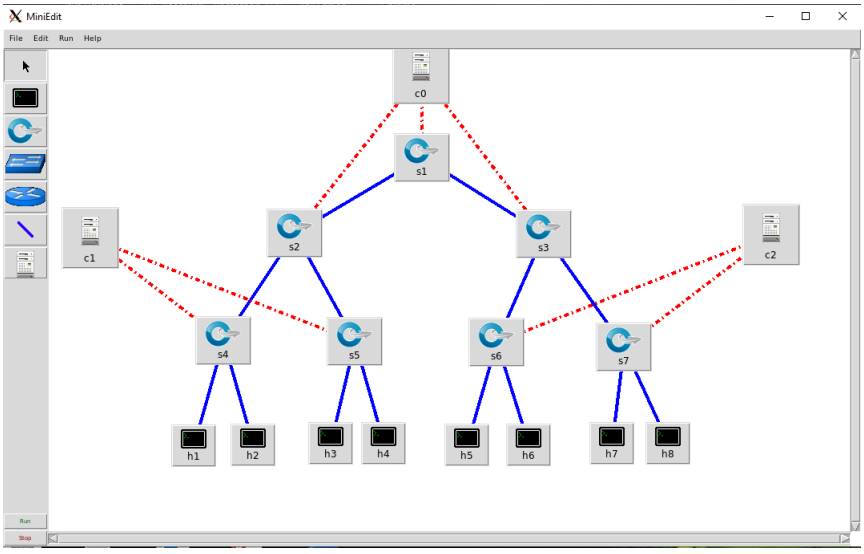

FIGURE 3.4 – Exemple de topologie du réseaux via Miniedit

Lorsque la topologie a été créée avec miniedit, il est possible de l'exporter dans un fichier Python en allant dans `File / Export Level 2 Script`. Le fichier Python pourra être utilisé comme un fichier Python normal. Le code python offre toutes les fonctionnalités que l'on peut trouver dans mininet.

Pour notre cas, la partie infrastructure de notre architecture sera customisée et générée sur la plateforme Mininet. La topologie a été créée par le biais d'un script Python (Voir script dans la partie annexe) sauvegardé dans le répertorie `mininet/custom`.

La commande pour créer notre réseau personnalisé cité ci-dessus est la suivante :

```
sudo mn - -controller=remote,ip=<CTRL_IP_ADR> -custom
mininet/custom/NetworkTopo.py -topo mytopo -mac
```

Celle-ci générera les différents éléments, tel que hôtes, commutateurs de la couche physique qui sera connectée au contrôleur OpenDaylight ou bien ONOS à distance.

```
mininet@mininet-vm:~$ sudo mn --controller=remote,ip=192.168.40.137 --switch=ovs
k,protocols=OpenFlow13 --custom mininet/custom/NetworkTopo_networktopo.py --topo
 networktopo --mac --ipbase=192.168.40.0/24
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.40.137:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
A1 A2 A3 A4 C1 C2 D1 D2
*** Adding links:
(C1, D1) (C1, D2) (C2, D1) (C2, D2) (D1, A1) (D1, A2) (D1, A3) (D1, A4) (D1, D2)
 (D2, A1) (D2, A2) (D2, A3) (D2, A4) (h1, A1) (h2, A1) (h3, A2) (h4, A2) (h5, A3
) (h6, A3) (h7, A4) (h8, A4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 8 switches
A1 A2 A3 A4 C1 C2 D1 D2 ...
*** Starting CLI:
mininet> pingall
```

FIGURE 3.5 – Création de la topologie

Après un `pingall` global, le contrôleur détecte les hôtes et chaque hôte arrive a pinguer tous les autres. À noter que le ping suffit pour assurer la connexion ou non, c'est un outil très courant utilisé pour vérifier la connectivité entre deux hôtes d'un réseau et pour déterminer l'accessibilité des hôtes, la congestion du réseau et la durée du trajet, puis n'importe quel protocole application peut être utilisé par la suite. La figure suivante illustre le résultat premier du ping.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
```

FIGURE 3.6 – `pingall` de la topologie

Les contrôleurs après ceci parvient à afficher la structure du réseau sous-jacent. Ces structures vont être illustrées dans des sections suivantes.

3.6 Réseau sous le contrôle d'OpenDaylight

3.6.1 Connexion avec Mininet

Après connexion de mininet avec le contrôleur OpenDaylight en introduisant son adresse IP dans la ligne de commande de Mininet, et après effectuer la commande `pingall`, ceci générera les différents éléments mentionnés auparavant, et parvient à afficher la structure du réseau sous-jacent dans l'interface graphique d'ODL "Dlux" comme l'illustre la figure suivante :

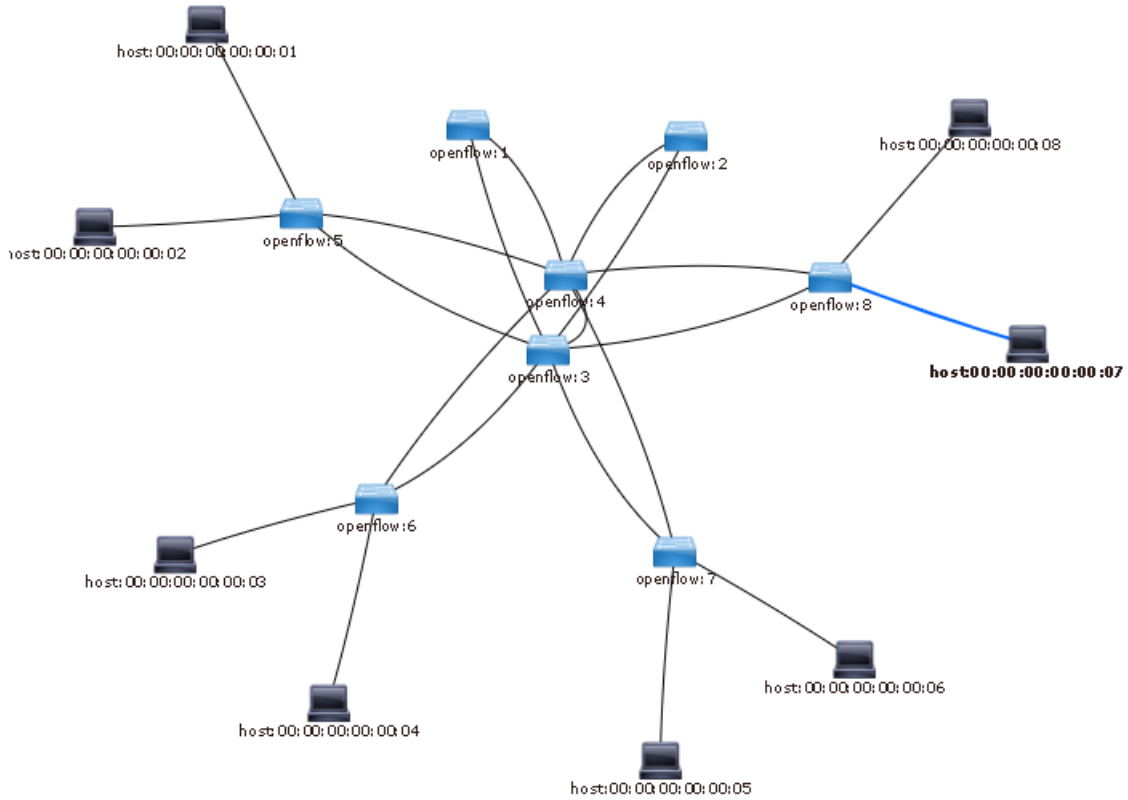

FIGURE 3.7 – Topologie du réseau sous ODL

3.6.2 Installation des flux sous OFM

Une fois l'application OFM est installée et configurée (voir en Annexe), l'url qui nous renvoie l'interface graphique est le suivant : http ://<ADRESSE_IP> :9000. Voici l'interface utilisateur d'OFM qui s'affiche avec notre topologie :

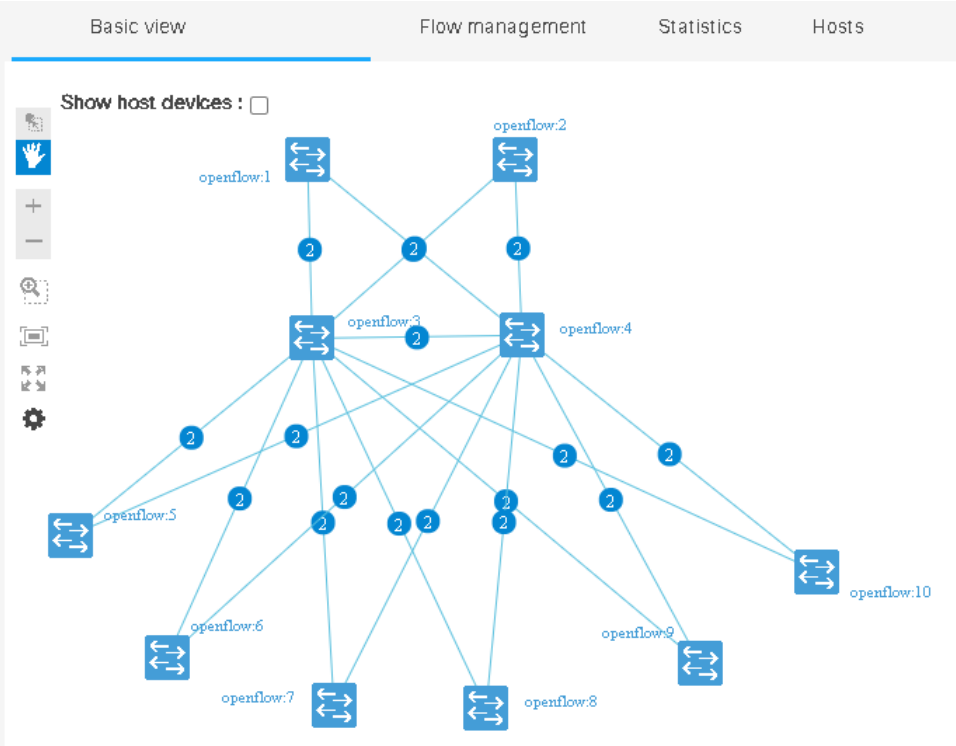

FIGURE 3.8 – Topologie sous OFM

Il existe quatre onglets des fonctions de base suivante :

— **Basic view :** Vue globale de la topologie sous-jacente en affichant les switchs OpenFlow et les hôtes qui leurs sont connectés

— **Flow management :** Ou gestion de flux. Permet de visualiser, ajouter, modifier ou supprimer les différents flux. Voici l'interface Flow Management donnant la possibilité de voir un résumé, filtrer ou modifier les flux.

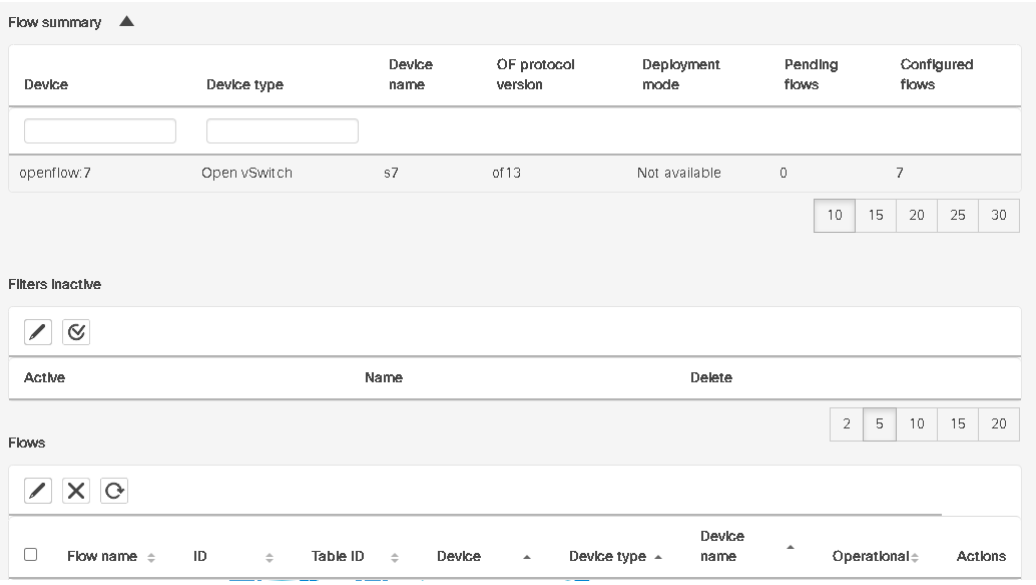

FIGURE 3.9 – Interface Flow managmenet

— **Statistics :** Fournit les statistiques liées aux flux et aux ports des switchs.

— **Hosts :** Résume les informations concernant les hôtes gérés par OFM

Pour l'installation des flux, nous allons seulement afficher ci-dessous l'un des flux introduits, qui est le flux 205 de la table de flux de A1 sous OFM :

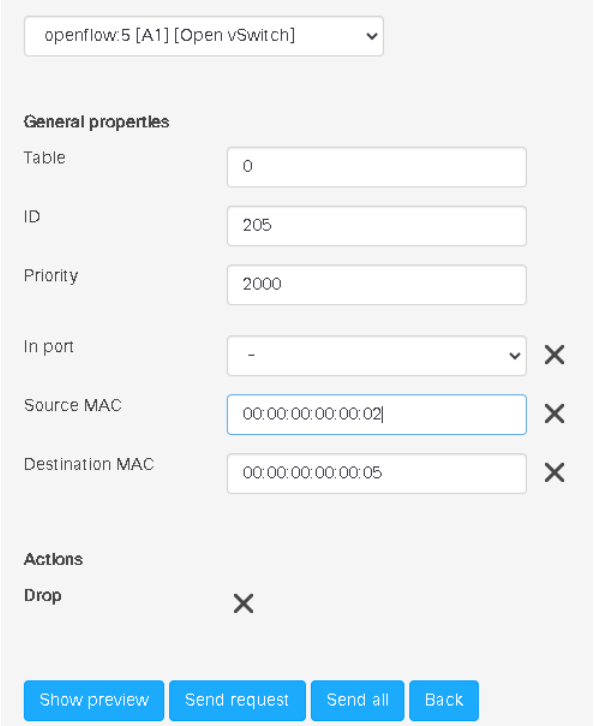

FIGURE 3.10 – Installation du flux 205 avec OFM -ODL-

La suit des règles sera appliquées de la même manière, et pour confirmer le comportement du réseau en termes d'accessibilité, voici un test de `pingall` comme l'atteste la figure suivante :

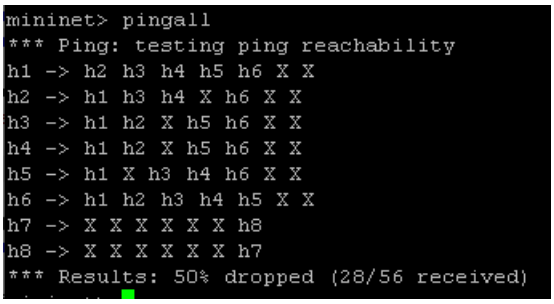

FIGURE 3.11 – test d'accessibilité des hôtes

Le réseau se comporte comme prévu, ce qui confirme la fiabilité des restrictions appliquées.

3.7 Réseau sous le contrôle d'ONOS

3.7.1 Connexion avec Mininet

De la même manière que la connexion qui a été établi avec le contrôleur ODL, on fait de même pour ONOS, mais cette fois-ci en changeant juste l'adresse IP pour mettre celle d'ONOS.

La figure suivante illustre la topologie sous le contrôleur ONOS :

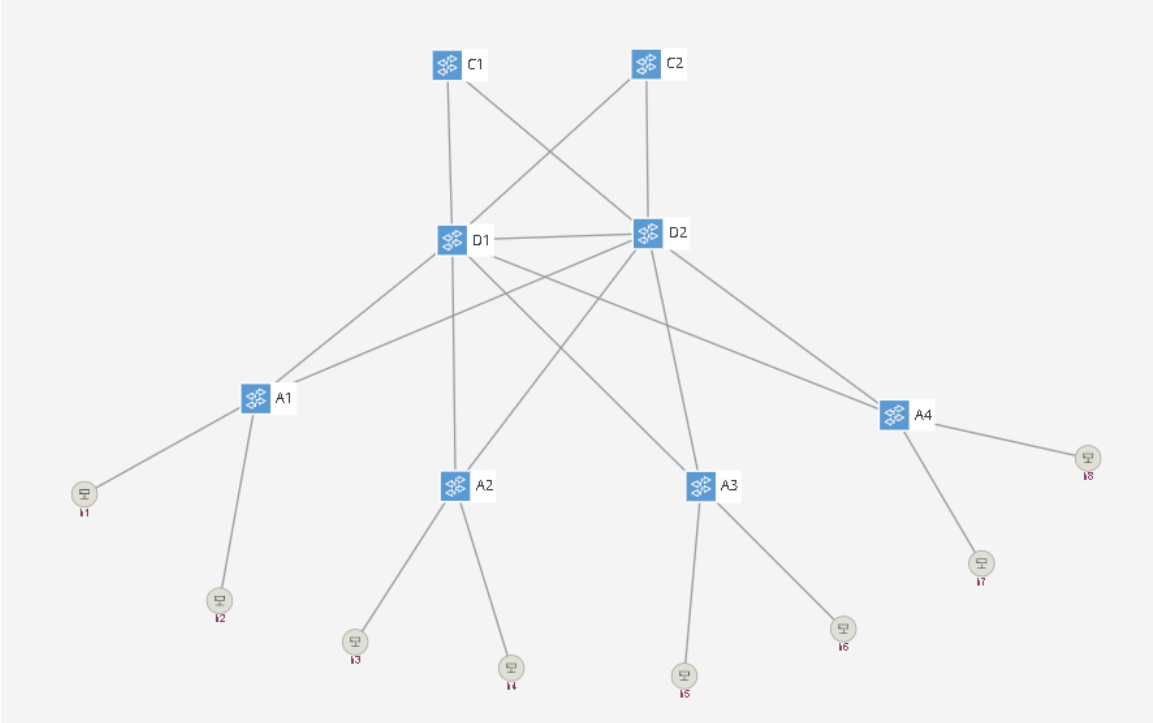

FIGURE 3.12 – Topologie du réseau sous Onos

3.7.2 Installation des flux avec le REST API

l'API REST peut-être consultée depuis n'importe quelle instance ONOS en cours d'exécution à l'URL suivante : `http://<IP_Adresse>:8181/onos/v1/docs/#/`. En ce qui suit l'interface du REST API d'ONOS :

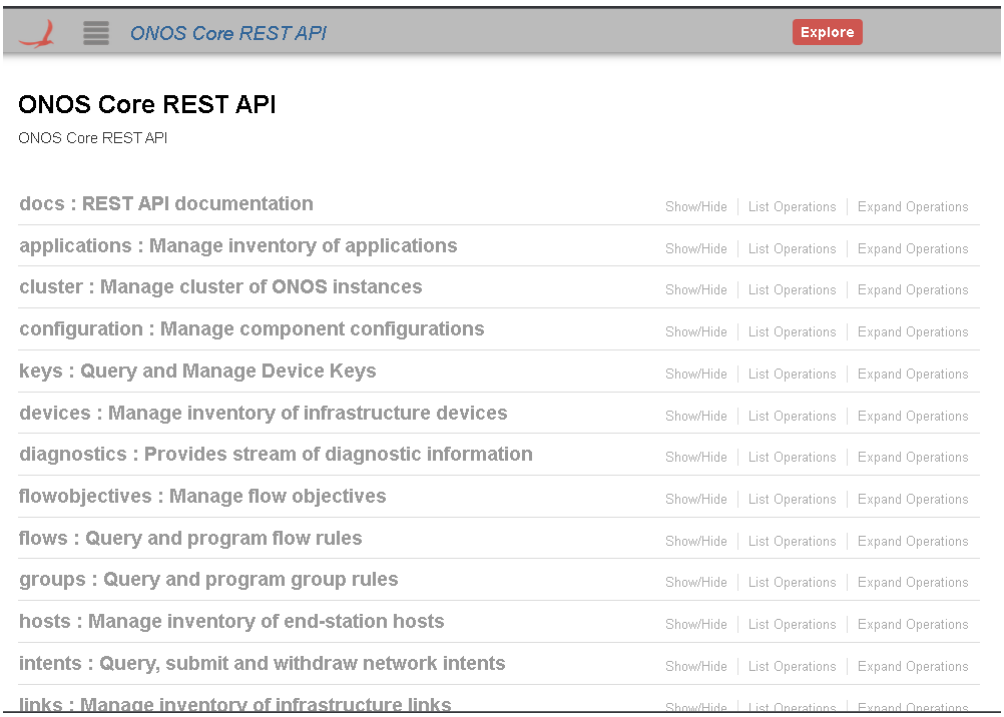

FIGURE 3.13 – Interface du REST API d'ONOS

Il existe plusieurs actions pour chacune de ces fonctionnalités, pour le flux par exemple, on peut voir (GET), supprimer (DELETE), ou bien rajouter (POST) un flux, en introduisant l'id_flux, ou bien l'id_flux et l'id_device, ou bien une action sans argument pour tous les flux existants. Ceci peut être illustrée par la figure suivante :

FIGURE 3.14 – Fonctionnalités du REST API d'ONOS pour les flux

Les mêmes règles qu'ODL sont introduites sous ONOS, mais cette fois-ci sous forme de script Json. Nous ne présentons que le script des flux 205 et 817 :

— **Flux 205 du switch A1 :**

{ **"id" : "205",**
"tableId" : "0",
"appId" : "205",
"groupId" : 0,
"priority" : 45000,
"timeout" : 0,
"isPermanent" : true,
"deviceId" : "of :0000000000000005",
"state" : "ADDED",
"life" : 0,
"packets" : 0,
"bytes" : 0,
"liveType" : "UNKNOWN",
"lastSeen" : 0,
"selector" : {
"criteria" : [
{
"type" : "ETH_DST",
"mac" : "00 :00 :00 :00 :00 :05"
},
{
"type" : "ETH_SRC",
"mac" : "00 :00 :00 :00 :00 :02"
}
]
},
"treatment" : {
"instruction" : [
{
"order" : 0,
"apply-actions" : {
"action" : [
{
"order" : 0,
"drop-action" : {}
}
] } }] } }

— **Flux 817 du switch A4 :**

{ **"id" : "817",**
"tableId" : "0",
"appId" : "817",
"groupId" : 0,
"priority" : 50000,
"timeout" : 0,
"isPermanent" : true,
"deviceId" : "of :0000000000000008",
"state" : "ADDED",
"life" : 0,
"packets" : 0,
"bytes" : 0,
"liveType" : "UNKNOWN",
"lastSeen" : 0,
"treatment" : {
"instructions" : [
{
"type" : "OUTPUT",
"port" : "1"
}
]
},
"selector" : {
"criteria" : [
{
"type" : "IN_PORT",
"port" : 2
}
] } }

Voici un exemple d'installation du flux 718 du Vswitch A4 sous REST API d'ONOS :

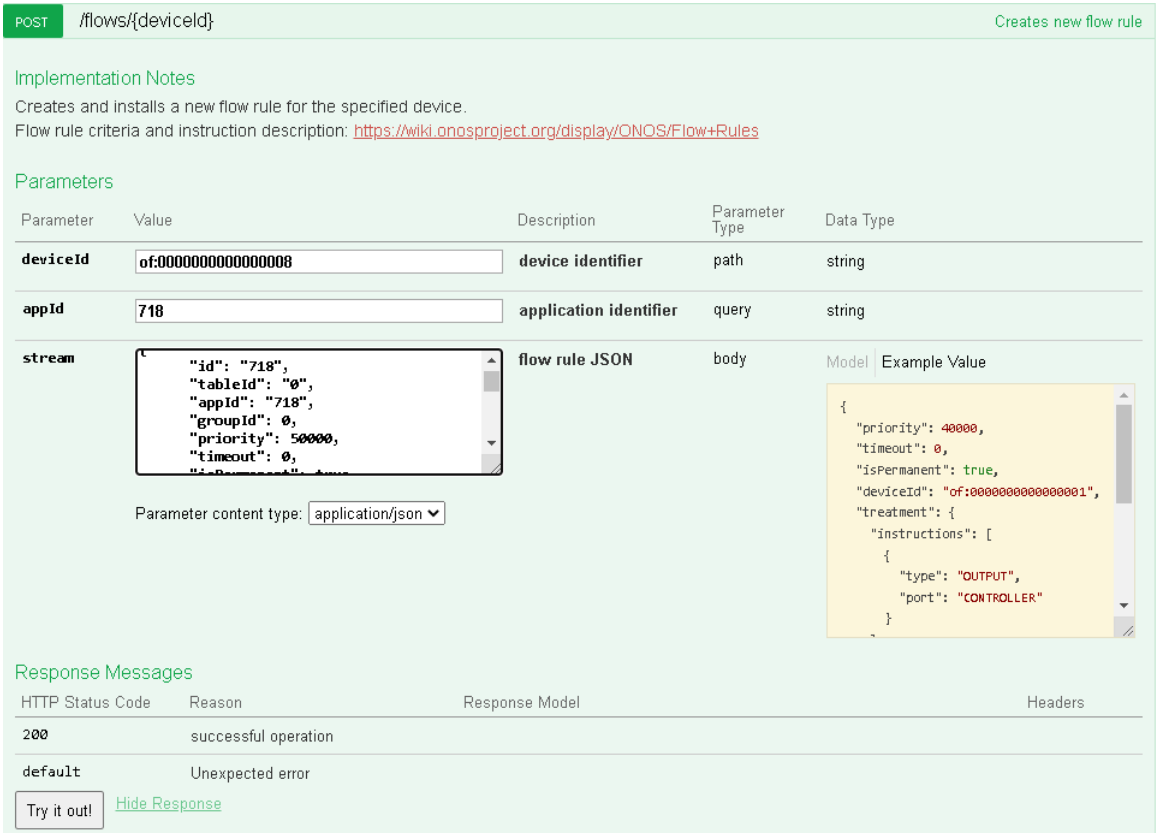

FIGURE 3.15 – Installation du flux 718 avec ONOS API

La suit des règles sera appliquées de la même manière, et pour confirmer le comportement du réseau en termes d'accessibilité, voici un test de `pingall` comme l'atteste la figure suivante :

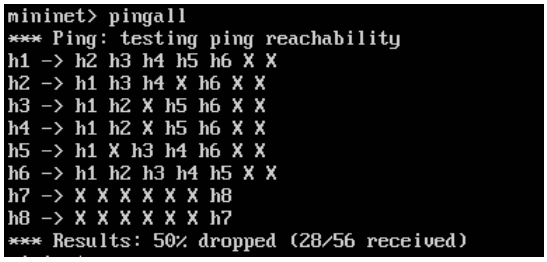

FIGURE 3.16 – Test d'acceptabilité sous ONOS

3.8 Méthodologie de test des performances du réseau

L'objectif de cette partie est d'évaluer ; en fonction de trois paramètres : la latence moyenne de configuration, le débit, et la gigue, dans le programme de la plateforme Mininet en utilisant l'outil iperf et ping. Nous insistons sur l'équité de notre comparaison entre les contrôleurs en adoptant le même banc d'essai et les mêmes paramètres que ceux décrits précédemment. Dans le cas de la mesure avec la commande iperf l'un des hôtes d'extrémité (dans notre démonstration c'est l'hôte h1) exécute la commande iperf en mode serveur et l'autre hôte (h5 pour la démonstration) exécute la commande iperf en mode client.

Les résultats de la mesure du RTT, du débit et de la gigue lors de la connexion de deux contrôleurs ODL et ONOS à notre plan de données seront présentés pour comparer et évaluer les performances de ces contrôleurs pour voir l'effet de l'utilisation de deux contrôleurs sur ces paramètres.

3.8.1 L'outil de test Iperf et la commande ping

Iperf (internet performance), outil de benchmarking qui a la fonctionnalité de client et de serveur et qui peut créer du trafic TCP (Transmission Control Protocol) ou UDP (User Datagram Protocol) pour calculer le débit dans une ou deux directions entre les deux extrémités du réseau [38]. En exécute la commande iperf dans la console de l'interface de ligne de commande (CLI) Mininet.

Ping envoie un ou plusieurs paquets de requête d'écho ICMP (Internet Control Message Protocol) à l'adresse IP de destination du réseau et attend une réponse. Arrivé à destination, il envoie une réponse d'écho ICMP. La latence est mesurée avec la commande ping.

Après que Mininet se soit conformé avec succès à l'établissement de la session, nous démarrons 2 terminaux (h1 et h5) pour exécuter iPerf ou bien ping afin de générer du trafic entre les hôtes, la commande suivante exécutée dans l'invité de commande de Mininet pour démarrer les terminaux des hôtes : `mininet> xterm h1 h5`. Nous enregistrons les résultats des mesures d'IPerf dans un fichier que nous analyserons avec l'outil Gnuplot après.

3.8.2 Paramètres d'analyse comparative

Les tests évalués seront sur la topologie Data Center du chapitre précédent. Pour chacun des tests mentionnés, les paramètres de base des performances du réseau (RTT, débit et gigue) seront mesurés pour évaluer les performances des contrôleurs ONOS et ODL. Ces paramètres sont mesurés entre les deux hôtes d'extrémité suivants : entre **h1 et h5**, **h3 et h2** et enfin **h4 et h6**. Entre chaque couple d'hôtes semblables (comme le couple h1 et h5 avec le couple h2 et h6) nous choisissons qu'un seul d'entre eux afin de ne pas avoir de tests redondants.

Les paramètres à tester sont les suivants (Pour les commandes nous donnant que l'exemple de test entre le hôte h1 et h5) :

3.8.2.1 Mesure de la latence (RTT)

La latence moyenne d'établissement de flux est le temps nécessaire à un paquet pour aller de la source à une destination et retour. Pour chacun des tests, le RTT moyen en millisecondes (ms) est mesuré à l'aide de la commande Ping. Ping fonctionne en envoyant un paquet ICMP echo=request à une adresse, puis en attendant une réponse ICMP echo-reply. Pour mesurer la latence nous exécutons dans le terminal du hôte (h1 dans notre cas) la commande suivante :

```
ping 192.168.40.5 -c 15 -i 2
```

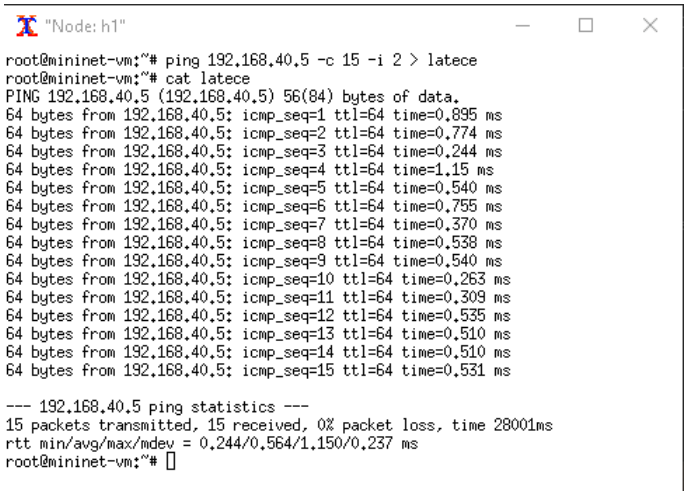

FIGURE 3.17 – Test de latence entre h1 et h5

3.8.2.2 Mesure du débit

Le deuxième paramètre de performance est le débit. Le débit définit la quantité de données utiles pouvant être transmises par unité de temps ; il est égal à la bande passante s'il n'y a pas de protocole, cependant, dans la plupart des cas pratiques, le débit est inférieur à la bande passante. Pour chaque test, le débit en mégaoctets par seconde (Mbytes/sec) est mesuré à l'aide de la commande Iperf en établissant une connexion TCP entre le client iperf et le serveur iperf tout en spécifiant les paramètres du serveur. Pour notre démonstration avec le couple h1 et h5, elle est utilisée pour faire fonctionner la machine h1 comme serveur (192.168.40.1) et l'autre comme client (192.168.40.5). L'iPerf est configuré

pour exécuter ce test pendant 30 secondes. Afin de mesurer le débit nous exécutons ce qui suit :

— Coté serveur h1 : `iperf -s -i 2`

— Coté client h5 : `iperf -c 192.168.40.1 -t 30`

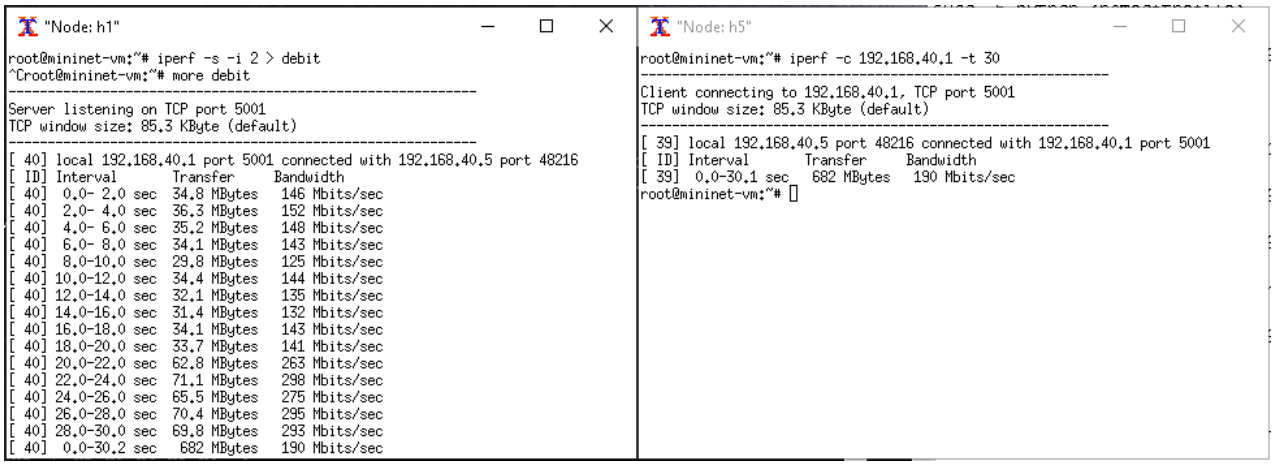

FIGURE 3.18 – Test du débit entre h1 et h5

3.8.2.3 Mesure de la gigue

Enfin, la gigue qui est la variabilité du délai du paquet en ms. L'effet de la gigue n'est pas visible dans les services qui n'ont pas besoin de temps réel pour la transmission, comme le transfert de données, alors que son effet est significatif dans les services en temps réel comme les services vidéo et audio. Elle est mesurée à l'aide d'Iperf, en établissant une connexion UDP entre le client et le serveur d'Iperf. En effet, UDP n'utilise aucun algorithme pour assurer l'arrivée du paquet à destination et envoie les datagrammes les uns après les autres sans retransmission. Cela se produit souvent à cause de l'encombrement du réseau ainsi que des changements de chemin. Pour chaque test, la gigue est mesurée en 30 s, et en fixant le débit à envoyer à 10 Mbits/seconde. Afin de mesurer la gigue nous exécutons ce qui suit :

— Coté serveur h1 : `iperf -s -u -i 2`

— Coté client h5 : `iperf -c 192.168.40.1 -b 10m -t 30`

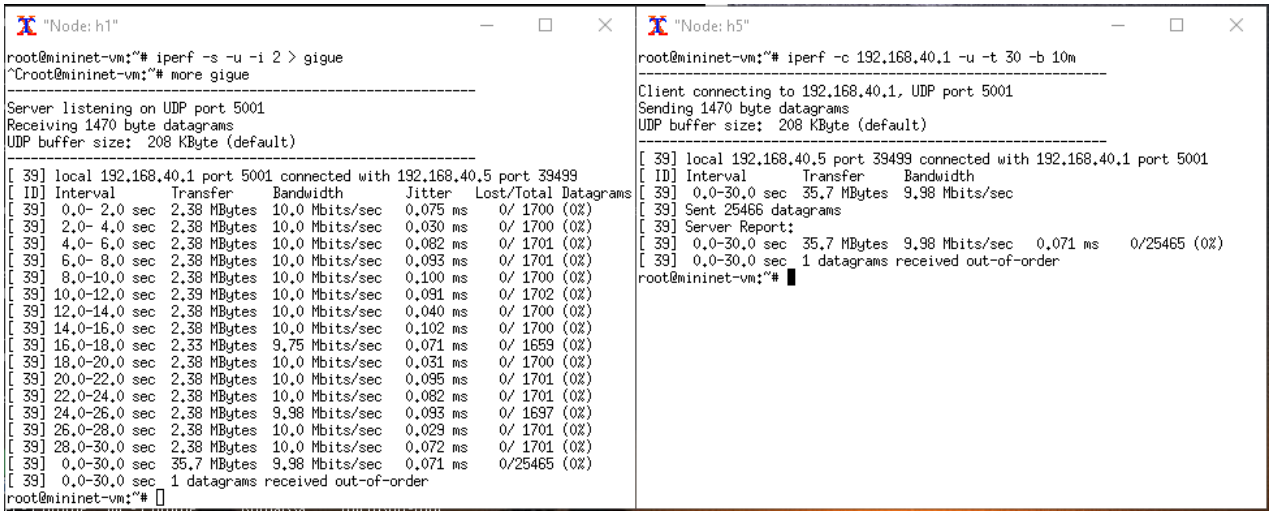

FIGURE 3.19 – Test de gigue entre h1 et h5

Toutefois, il convient de mentionner que les performances en termes de RTT, de débit et de gigue ne doivent pas être les seuls facteurs utilisés pour choisir parmi les différents contrôleurs ; d'autres facteurs tels que la fiabilité et la convivialité sont également importantes.

3.9 Résultats, analyse et discussion

Dans cette section, nous présentons les résultats d'évaluation de l'expérience de mise en réseau SDN, en tant que mise en œuvre et pratique dans un cadre de test et d'évaluation. Toutes les expériences se sont déroulées en deux phases en utilisant la même topologie Data Center présenté précédemment. Dans la première phase, le contrôleur ODL était connecté au plan de données et dans la deuxième phase, le contrôleur ONOS était connecté au plan de données. A chaque exécution, les mesures de performance latence, débit et gigue en fonction du temps ont été mesurées.

3.9.1 Latence

La figure ci-après où l'axe des X représente le temps, et l'axe des Y la latence en ms présente la variation de latence entre les quatre couples des hôtes pendant une durée de 30 secondes du test de ping. Le contrôleur OpenDaylight a fait une latence de 0.048 ms comme minimale valeur, et de 1.661 ms comme valeur maximale ; alors que dans le cas du contrôleur ONOS, elle était de 3.422 ms comme valeur minimale et de 53.444 ms comme valeur maximale !

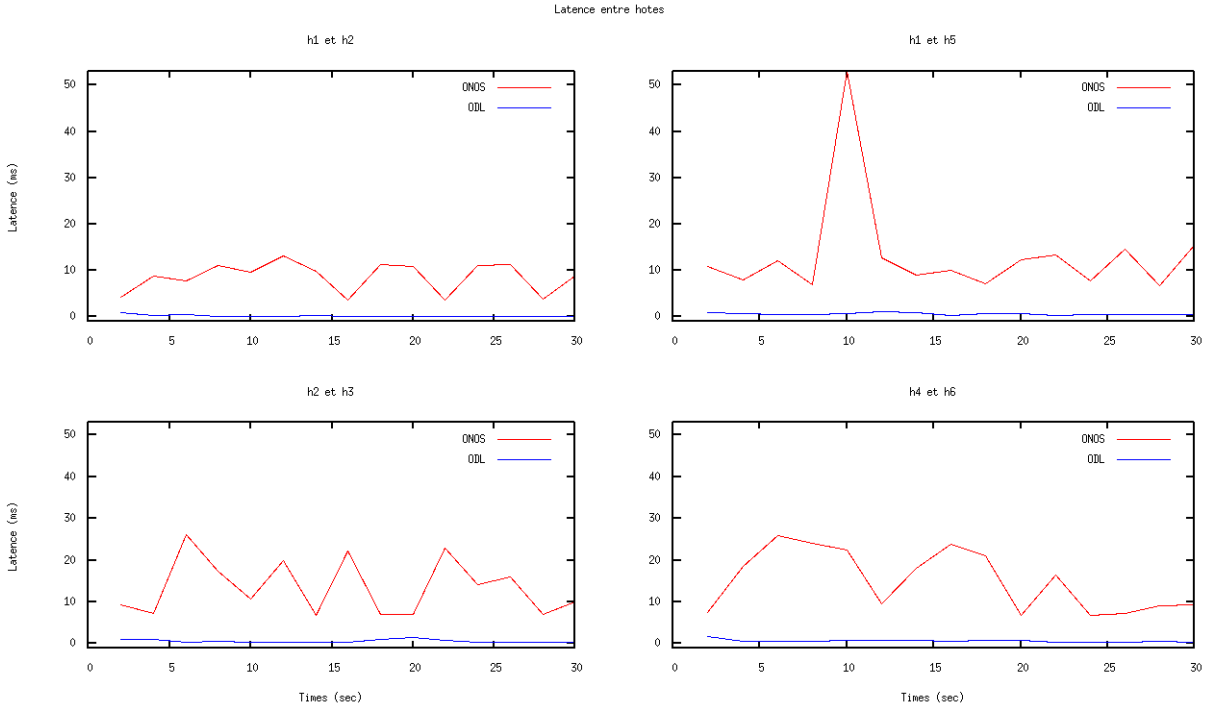

FIGURE 3.20 – Résultats des tests de la latence

Le tableau suivant résume les valeurs moyennes de nos tests de latence pour les deux contrôleurs OpenDaylight et ONOS (la présentation des valeurs de tests est présenté sous cette forme : min/avg/max/mdev)

TABLE 3.7 – Résultats moyens de la latence

Paramètre	Test pour	OpenDaylight	ONOS
Latence	h1 & h2	0.048/0.146/0.770/0.193 ms	3.422/8.540/13.099/3.198 ms
	h1 & h5	0.254/0.527/1.007/0.221 ms	6.668/13.295/53.444/11.087 ms
	h2 & h3	0.268/0.577/1.529/0.361 ms	6.838/13.611/26.198/6.501 ms
	h4 & h6	0.271/0.622/1.661/0.327 ms	6.682/15.109/25.940/7.065 ms
	Moyenne	0.210/0.468/1.242/0.275 ms	5.903/12.639/29.67/2.108 ms

D'après les valeurs moyennes obtenues à partir des expériences, le réseau présente les meilleures performances avec le contrôleur OpenDaylight. Clairement, on observe que le contrôleur ODL présente une latence beaucoup moins élevé et en moyenne de 26 fois que pour celle du contrôleur ONOS.

3.9.2 Débit

La figure ci-après où l'axe des X représente le temps, et l'axe des Y le débit en Mbyts/sec présente le débit entre les quatre couples des hôtes, pendant une durée de 30 secondes du test iperf TCP. Inversement au cas de la latence, un débit brut élevé est vu pour le cas du couple h1 et h2 avec une valeur maximale de 3835 MBytes/sec et de 14.2 Mbytes/sec en valeur minimale

pour le couple d'hôtes h4 et h6 dans le cas du contrôleur OpenDaylight. Alors que pour le contrôleur ONOS n'a pu transférer qu'environ 1.76 MBytes/sec en valeur maximale, et 0.50 Mbytes/sec en valeur minimale avec les mêmes couples des hôtes que celle mentionnés dans le cas d'ODL. Le débit le plus élevé été pour le couple h1 et h2 et le mois élevé été pour le couple h4 et h6. Pendant tout l'intervalle du temps ODL a toujours surpassé ONOS en terme de débit.

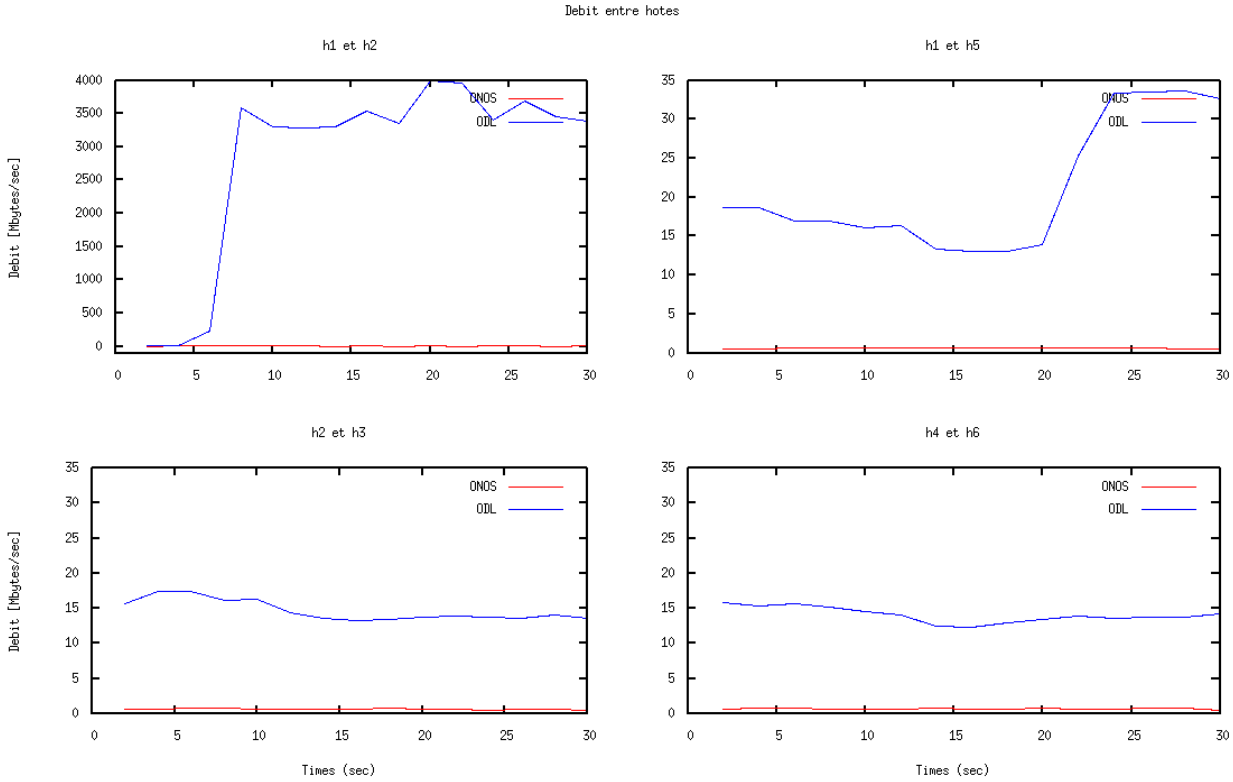

FIGURE 3.21 – Résultats des tests du débit

Le tableau suivant résume les valeurs moyennes de nos tests de débit pour les deux contrôleurs OpenDaylight et ONOS :

TABLE 3.8 – Résultats moyens du débit

Paramètre	Test pour	OpenDaylight	ONOS
Débit	h1 & h2	2835 MBytes/sec	1.26 MBytes/sec
	h1 & h5	21.2 MBytes/sec	0.82 MBytes/sec
	h2 & h3	14.8 MBytes/sec	0.73 MBytes/sec
	h4 & h6	14.2 MBytes/sec	0.56 MBytes/sec
	Moyenne	721.3 Mbyts/sec	0.842 Mbyts/sec

D'après les valeurs moyennes obtenues à partir des expériences, ODL a pu transférer en moyenne de 721.3 Mbyts/sec, tandis qu'ONOS n'a pu transférer que 0.842 Mbyts/sec. le réseau présente les meilleures performances avec le contrôleur OpenDaylight avec un débit plus élevé d'environ 850 fois que le débit moyen du contrôleur ONOS !

3.9.3 Gigue

La figure suivante où l'axe des X représente le temps, et l'axe des Y la gigue en ms montre la gique obtenue dans la topologie Data Center dans les tests iperf UDP pour les deux contrôleurs étudiés. Comme on peut l'observer, ODL a fait une valeur maximal de gigue de 0.210 ms, et pour ONOS elle est de 2.632 ms. Dans les quatre cas de couples d'hôtes, la gigue obtenue en utilisant le contrôleur ONOS est supérieur à celle du contrôleur ODL.

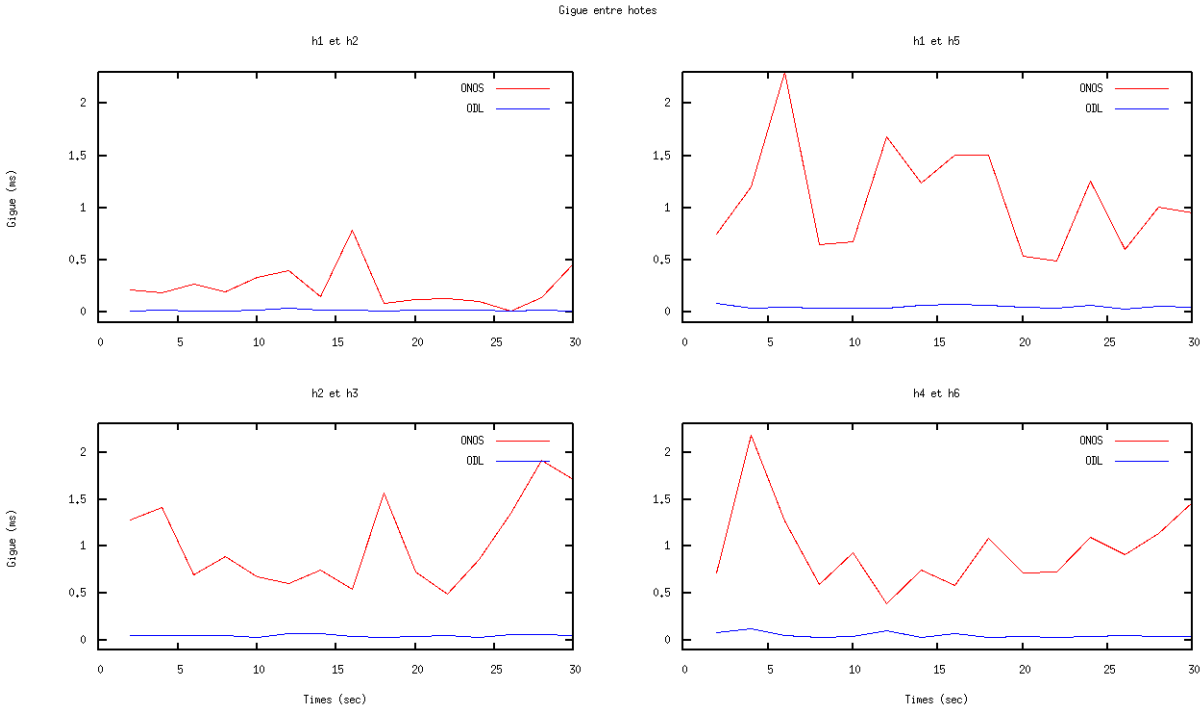

FIGURE 3.22 – Résultats des tests de la Gigue

Le tableau suivant résume les valeurs moyennes pour les quatre couples d'hôtes, ainsi que la moyenne globale pour tous les tests de gigue pour les deux contrôleurs OpenDaylight et ONOS :

TABLE 3.9 – Résultats moyens de la gigue

Paramètre	Test pour	OpenDaylight	ONOS
Gigue	h1 & h2	0.009 ms	0.267 ms
	h1 & h5	0.040 ms	0.787 ms
	h2 & h3	0.041 ms	0.860 ms
	h4 & h6	0.042 ms	0.824 ms
	Moyenne	0.033 ms	0.684 ms

En ce qui concerne la gigue moyenne, pour OpenDaylight elle n'était que d'environ 0,033 ms en moyenne, alors qu'elle était de 0.684 ms en moyenne dans le cas du contrôleur ONOS, ce qui est considérablement plus élevé par rapport à ODL d'environ 20 fois. Donc OpenDaylight a surpassé aussi ONOS en terme de gigue moyenne.

3.10 Table comparatif des deux contrôleurs ODL et ONOS

La table suivante résume une comparaison générale des deux contrôleurs étudié : OpenDaylight et ONOS .

Fonctionnalités	OpenDaylight	ONOS
REST API	Bien, mais un peu pas claire	Claire et offre plus de fonctionnalités
Bugs	Fait parfois des bugs inattendu et il faut le réexécuter. La suppression d'une règle d'OFM le bloque	Il ne fait pas généralement de bugs
Règles de flux	OFM : plus facile lors de l'application des règles, pas besoin d'écrire un script (il est généré)	REST API : Il faut entrer un script Json qui ne detecte pas où est l'erreur si elle existe
Actualisation	Il faut toujours actualiser l'interface après application de n'importe quel règle	L'actualisation se fait automatiquement
Arrêt d'exécution	Il ne faut pas réintroduire les règles à nouveau, elles sont enregistrées	Il faut réintroduire tout les règles à nouveau à chaque fois
Performance	Meilleur performance en terme de débit, latence et de la gique	Offre de performance moins meilleur

Table 3.10 – Table comparative générale entre les contrôleurs ODL et ONOS

3.11 Conclusion

Nous avons vu dans ce qui précède, la configuration d'une architecture réseau SDN dans un environnement Data Center implémenté avec des solutions open source. La mise en œuvre du réseau proposé nous a permis d'expérimenter les avantages de la virtualisation (qui nous a permis à travers notre machine seulement, de construire l'ensemble de notre réseau émulé et contrôlée), ainsi que les avantages du réseau SDN par rapport au réseau traditionnel, ce qui offre de nombreux avantages opérationnels, il accroît la flexibilité et accélère les délais de mise sur le marché des nouvelles applications.

On évaluant les performances de notre réseau avec les outils proposés, cela nous a conduits comme résultats à ce que le contrôleur OpenDaylight surpasse le contrôleur ONOS en terme de latence de 26 fois moins en moyenne, de débit d'environ 850 fois plus élevé et même de gigue, avec une moyenne de 20 fois moins.

Conclusion générale

L'automatisation ouvre tellement de nouvelles perspectives des réseaux, entre autres le pouvoir de renforcer l'agilité, la sécurité ou encore les capacités et taille 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.

Grâce alors au SDN, réseau définie par logiciel qui fait la séparation de la logique de contrôle du réseau des périphériques réseau, en favorisant sa centralisation du contrôle : Il permet la séparation entre la définition des politiques de réseau, leur mise en œuvre dans le matériel de commutation et le transfert du trafic. La clé de la flexibilité souhaitée réside en décomposant le problème du contrôle du réseau en éléments traitables. le SDN facilite la création et l'introduction de nouvelles abstractions dans le réseau. Les réseaux donc 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.

Notre démarche dans ce projet consiste à développer dans un premier lieu une application de gestion de réseau de type DATA CENTER dans un environnement SDN implémenté en utilisant l'émulateur Mininet qui est une plateforme SDN open source populaire et flexible, ce qui nous a permis d'expérimenter ces avantages ainsi que son efficacité par rapport aux réseaux traditionnels. Dans un deuxième temps, nos travaux ont consisté à étudier différent contrôleurs SDN proposées et opté à l'utilisation de deux d'entre eux dans le monde open source dans le but d'identifier lequel est le mieux adapté à répondre aux besoins des réseaux. C'est ainsi que les contrôleurs ONOS et OpenDaylight par la diversité et le nombre des fonctions offertes se sont imposés comme les principaux candidats.

Nous avons par la suite concentré nos efforts au contrôle de notre topologie de type DATA CENTER en se basant sur le contrôleur ODL seulement dans un premier lieu, puis sur le contrôleur ONOS dans un deuxième lieu, tout en testant la conformité de ce contrôle par rapport à notre architecture. Enfin une étude comparative des deux contrôleurs choisit a été faite afin de faire l'évaluation des performances de notre réseau en terme de débit, de latence et de la gigue moyenne sous leurs contrôle.

Les résultats obtenus ont montré que le contrôleur OpenDaylight offre les meilleures performances en terme de débit, latence et même de gigue pour notre cas d'étude (Data Center avec 16 nœuds), qu'ONOS qui présente de performance moins meilleur. Cependant, et selon nos travaux qui ont été effectués sur les deux contrôleurs, ONOS permet une facilité de gestion de la topologie, élément-clé pour la raison d'être du SDN qui est facilité de la gestion de

la topologie, et de mise en œuvre, la facilité du déploiement et de configuration d'un environnement réseau. Ainsi, et pour intégrer une approche SDN dans une architecture réseau, il est primordial d'étudier le meilleur choix de contrôleur adapté au besoin, aux dimensions, et aux spécificités de l'architecture réseau cible.

Perspectives

Divers améliorations pourraient être apportées à ce projet pour une plus grande efficacité :

- L'implémentation des solutions SDN pour un réseau réel.
- En plus de l'évaluation qui est mise en œuvre dans ce mémoire, l'évolutivité et la fiabilité sont deux facteurs qui sont importants à considérer lors du choix d'un contrôleur OpenFlow, ainsi il est aussi nécessaire d'effectuer un autre test approfondit pour les contrôleurs OpenFlow les plus récents concernant aussi la sécurité et la qualité de service et ceci avec différentes tailles de réseau.
- Dans ce travail, nous nous sommes concentrés sur l'évaluation des contrôleurs en mode simple. Une vue meilleure à visions d'améliorer l'évolutivité, la fiabilité et les performances du réseau est de mettre en œuvre un regroupement de contrôleurs distribués dans les réseaux SDN, ie, un fonctionnent en mode cluster des contrôleurs SDN.

Bibliographie

[1] "red hat," site figure. [online]. avaliable at : https://www.redhat.com/fr/topics/containers/containers-vs-vms.

[2] "blogs univ poitier," site figure. [online]. avaliable at : https://blogs.univ-poitiers.fr/f-launay/2018/01/15/principe-du-sdn-dans-une-architecture-reseau-classique/.

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, 103(1) :14–76, January 2015.

[4] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab. SDN controllers : A comparative study. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pages 1–6, April 2016. Journal Abbreviation : 2016 18th Mediterranean Electrotechnical Conference (MELECON).

[5] "univ paris diderot," site figure. [online]. avaliable at : https://sysblog.informatique.univ-paris-diderot.fr/2020/03/11/opendaylight-mais-que-ce-que-cest/.

[6] *Réseaux informatiques - Notions fondamentales (Normes, Architecture, Modèle OSI, TCP/IP, Ethernet, Wi-Fi, ...).* 12 janvier 2009.

[7] Open networking foundation, "software-defined networking : The new norm for networks". [online]. avaliable https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/, 2012.

[8] "vmware,". site officiel. [online]. avaliable at : https://www.vmware.com/fr/company.html.

[9] A. C. Jaramillo, R. Alcivar, J. Pesantez, and R. Ponguillo. Cost Effective test-bed for Comparison of SDN Network and Traditional Network. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–2, November 2018. Journal Abbreviation : 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC).

[10] A. M. Joy. Performance comparison between Linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 342–346, March 2015. Journal Abbreviation : 2015 International Conference on Advances in Computer Engineering and Applications.

[11] M. Skulysh and O. Klimovych. Approach to virtualization of Evolved Packet Core Network Functions. In *The Experience of Designing and Application of CAD Systems in Microelectronics*, pages 193–195, February 2015. Journal Abbreviation : The Experience of Designing and Application of CAD Systems in Microelectronics.

[12] Mohamed Boucadair and Christian Jacquenet. Software-Defined Networking : A Perspective from within a Service Provider Environment. RFC 7149, March 2014.

[13] D. Tatang, F. Quinkert, J. Frank, C. Röpke, and T. Holz. SDN-Guard : Protecting SDN controllers against SDN rootkits. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 297–302, November 2017. Journal Abbreviation : 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN).

[14] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, Jamal Hadi Salim, David Meyer, and Odysseas Koufopavlou. Software-Defined Networking (SDN) : Layers and Architecture Terminology. RFC 7426, January 2015.

[15] M. Shin, K. Nam, and H. Kim. Software-defined networking (SDN) : A reference architecture and open APIs. In *2012 International Conference on ICT Convergence (ICTC)*, pages 360–361, October 2012. Journal Abbreviation : 2012 International Conference on ICT Convergence (ICTC).

[16] "open vswitch," site officiel. [online]. avaliable at : `https://www.openvswitch.org/`.

[17] M. P. Fernandez. Comparing OpenFlow Controller Paradigms Scalability : Reactive and Proactive. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 1009–1016, March 2013. Journal Abbreviation : 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA).

[18] OpenFlow Switch Consortium et al. Openflow switch specification version 1.0. 0. *http ://www. openflowswitch. org/documents/openflow-spec-v1. 0.0. pdf*, 2009.

[19] Y. Li and M. Chen. Software-Defined Network Function Virtualization : A Survey. *IEEE Access*, 3 :2542–2553, 2015.

[20] Ihssane Choukri, Mohammed OUZZIF, and Khalid Bouragba. Software Defined Networking (SDN) : Etat de L'art. In *Colloque sur les Objets et systèmes Connectés*, CASABLANCA, Morocco, June 2019. Ecole Supérieure de Technologie de Casablanca (Maroc), Institut Universitaire de Technologie d'Aix-Marseille (France).

[21] K. N. Thovheyi and T. Zuva. Impact of I/O workloads on Ram Performance for Virtual Systems. In *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–6, November 2019. Journal Abbreviation : 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC).

[22] Z. Li. Comparison between common virtualization solutions : VMware Workstation, Hyper-V and Docker. In *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, pages 701–707, November 2021. Journal Abbre-

viation : 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC).

[23] D. T. Vojnak, B. S. Đorđević, V. V. Timčenko, and S. M. Štrbac. Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation. In *2019 27th Telecommunications Forum (TELFOR)*, pages 1–4, November 2019. Journal Abbreviation : 2019 27th Telecommunications Forum (TELFOR).

[24] "mininet," site officiel. [online]. avaliable at : https://www.mininet.org/.

[25] S. Wang. Comparison of SDN OpenFlow network simulator and emulators : EstiNet vs. Mininet. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, June 2014. Journal Abbreviation : 2014 IEEE Symposium on Computers and Communications (ISCC).

[26] J. Shin B. Lee, S. H. Park and S. Yang. "nox : Towards an operating system for networks". [online]. avaliable at : http://www.opencontrail.org/, 2014.

[27] J. Shin B.Lee, S.H.Park and S. Yang. "pox controller manual current documentation". [online]. avaliable at :https://noxrepo.github.io/pox-doc/html/.

[28] "floodlight documentation.," site officiel. [online]. avaliable at : https://floodlight.atlassian.net/, 2011.

[29] David Erickson. The beacon openflow controller. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, page 13–18, New York, NY, USA, 2013. Association for Computing Machinery.

[30] J. Pettit et al. ACM SIGCOMM Computer Communication Review . [Online]. Avaliable at : http://www.opencontrail.org/ N. Gude, T. Koponen. "iris : The openflow based recursive sdn controller"., 2008.

[31] Part of LF Networking (LFN). "opendaylight documentation". site officiel. [online]. avaliable at : https://www.opendaylight.org, 2013.

[32] url = https ://osrg.github.io/ryu/index.html Ryu SDN Framework Community, title = "Ryu Controller". [Online]. Avaliable at : https://osrg.github.io/ryu/index.html.

[33] "openmul sdn platform," site officiel. [online]. avaliable at : http://www.openmul.org/openmul-controller.html, 2011.

[34] A.Cox Z.Cai and E.T.S.Ng. "maestro : A system for scalable openflow control,". [online]. avaliable at : http://www.cs.rice.edu/{~}eugeneng/papers/TR10-11.pdf, 2011.

[35] Hosted by The Linux Foundation. "onos documentation". site officiel. [online]. avaliable at : https://wiki.onosproject.org, December 5, 2014.

[36] A. Shirvar and B. Goswami. Performance Comparison of Software-Defined Network Controllers. In *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pages 1–13, February 2021. Journal Abbreviation : 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT).

[37] A. Bhardwaj, Z. Zhou, and T. A. Benson. A Comprehensive Study of Bugs in Software Defined Networks. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 101–115, June 2021. Journal Abbreviation : 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).

[38] "iperf," site officiel. [online]. avaliable at : `https://iperf.fr/iperf-download.php`.

Annexes

Annexe A

Mininet

1 L'image de la VM de MINIET qui existe dans le site officiel de `https://github.com/mininet/mininet/wiki/Mininet-VM-Images`.

2 Pour créer la topologie du réseau sous fichier python, voici les commandes a utilisées :

```
# Create hosts.
host = self.addHost( <host name> )
# Create a switch
switch = self.addSwitch( <name>, protocols='OpenFlow13')
#Configuration of links
configuration = dict(bw=<bw>, delay=<delay>,
max_queue_size=<max_queue_size>, loss=0,
use_htb=True)
# Add links between the switch and each host
addlink = self.addLink( <name1>, <name2>,
**configuration)
```

Script Python de la topologie proposée

from mininet.topo import Topo
from mininet.cli import CLI
from mininet.net import Mininet
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel, info
from mininet.node import Node, RemoteController, OVSSwitch
from mininet.util import irange
from mininet.link import TCLink

class NetworkTopo(Topo) :
def build(self) :

```
h1 = self.addHost( 'h1', ip='192.168.40.1/24')
h2 = self.addHost( 'h2', ip='192.168.40.2/24')
h3 = self.addHost( 'h3', ip='192.168.40.3/24')
h4 = self.addHost( 'h4', ip='192.168.40.4/24')
h5 = self.addHost( 'h5', ip='192.168.40.5/24')
h6 = self.addHost( 'h6', ip='192.168.40.6/24')
h7 = self.addHost( 'h7', ip='192.168.40.7/24')
h8 = self.addHost( 'h8', ip='192.168.40.8/24')
# Coeur
C1 = self.addSwitch( 'C1', dpid='0000000000000001',protocols='OpenFlow13' )
C2 = self.addSwitch( 'C2', dpid='0000000000000002',protocols='OpenFlow13' )
# Distribution
D1 = self.addSwitch( 'D1', dpid='0000000000000003',protocols='OpenFlow13' )
D2 = self.addSwitch( 'D2', dpid='0000000000000004',protocols='OpenFlow13' )
# Acccess
A1 = self.addSwitch( 'A1', dpid='0000000000000005',protocols='OpenFlow13' )
A2 = self.addSwitch( 'A2', dpid='0000000000000006',protocols='OpenFlow13' )
A3 = self.addSwitch( 'A3', dpid='0000000000000007',protocols='OpenFlow13' )
A4 = self.addSwitch( 'A4', dpid='0000000000000008',protocols='OpenFlow13' )
self.addLink(h1, A1)
self.addLink(h2, A1)
self.addLink(h3, A2)
self.addLink(h4, A2)
self.addLink(h5, A3)
self.addLink(h6, A3)
self.addLink(C2, D1)
self.addLink(C2, D2)
self.addLink(h7, A4)
self.addLink(h8, A4)
self.addLink(C1, D1)
self.addLink(C1, D2)
self.addLink(D1, D2)
self.addLink(D1, A1)
self.addLink(D1, A2)
self.addLink(D1, A3)
self.addLink(D1, A4)
self.addLink(D2, A1)
self.addLink(D2, A2)
self.addLink(D2, A3)
self.addLink(D2, A4)
def runNetworkTopo() :
# Create and test a simple network
topo = NetworkTopo()
```

```
# Create a network based on the topology using OVS and controlled by
# a remote controller
net = Mininet(topo=topo)
# Actually start the network
net.start()
# Drop the user in to a CLI so user can run commands
CLI( net )
# After the user exits the CLI, shutdown the network
net.stop()
if __name__ == '__main__' :
# Tell mininet to print useful information
setLogLevel('info')
runNetworkTopo()
# Allows the file to be imported using 'mn --custom <filename> --topo minimal'
topos = 'networktopo' : NetworkTopo
```

Annexe B

OpenDaylight

Installation d'ODL

Pour faire fonctionner ODL sur une plateforme Karaf, il faut télécharger Java et configurer la variable d'environnement JAVA_ HOME.

```
# sudo apt-get -y update
# sudo apt-get -y upgrade
# sudo apt-get -y install unzip
# sudo apt-get -y install openjdk-8-jre
# sudo update-alternatives –config java
# ls -l /etc/alternatives/java
# export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
# echo JAVA_HOME
# curl -XGET -O https://nexus.opendaylight.org/
content/repositories/#opendaylight.release/org/
opendaylight/integration/karaf/0.8.4/karaf-0.8.4.
zip
# unzip karaf-0.8.4.zip
# cd karaf-0.8.4/bin
# ./karaf
```

OpenDaylight Features

Commandes pour activer les fonctionnalités features d'ODL :

```
# feature:install odl-restconf-all odl-l2switch-all
odl-mdsal-all odl-dlux-all odl-openflowplugin-all
odl-yangtools-common
```

odl-restconf-all : Permet d'accéder à l'API RESTCONF ;
odl-l2switch-all : Fournit des fonctionnalités réseau similaires à celles d'un commutateur Ethernet ;
odl-mdsal-all : Permet d'accéder à l'API Yang ;
odl-dlux-all : Interface utilisateur graphique d'OpenDaylight ;
odl-openflowplugin-all : ajoute des flux par défaut au contrôleur ;
odl-yangtools-common : pour installer les concepts et utilitaires communs.

OpenFlow Manager

```
# sudo apt-get update
# sudo apt-get install -y npm
# sudo apt-get install –y nodejs
# git clone https://github.com/CiscoDevNet/
OpenDayLight-OpenFlow-App
# cd OpenDayLight-OpenFlow-App/
# sudo npm install -g grunt-cli
# sudo grunt
Utiliser le serveur web pour utiliser OFM
http://192.168.40.141:9000/#/openflow_manager/index
```

Annexe C

ONOS

Presque les mêmes étapes pour l'installation du contrôleur ODL, on doit installer curl en plus. Voici les étapes :

```
# sudo apt-get -y update && upgrade
# sudo apt-get install curl
# sudo apt-get install software-properties-common -y
&&
# sudo add-apt-repository ppa:webupd8team java -y &&
# echo "oracle-java8-installer shared/
# accepted-oracle-license-v1-1 select true"| sudo
debconf-set-selections && sudo apt-get install
oracle-java8-installer
# oracle-java8-set-default -y
site pour télécharger version ONOS
https://wiki.onosproject.org/display
# sudo wget -c http://repo1.maven.org/maven2/org/
onos<version>.tar.gz
# sudo tar xzf onos-2.0.0.tar.gz
# /opt/onos/bin/onos-service start
# sudo cp /opt/onos/init/onos.initd /etc/init.d/onos
# sudo cp /opt/onos/init/onos.service
/etc/systemd/system/
# sudo systemctl daemon-reload
# sudo systemctl enable onos
Pour ouvrir la fenetre GUI d'ONOS :
http://<IP_ADR>:8181/onos/ui/index.html
username : onos
password : rocks
Pour ouvrir le REST API d'ONOS :
http://<IP_ADR>:8181/onos/v1/docs/#/
```

Configurations nécessaires :

app activate org.onosproject.oneping

cfg set org.onosproject.fwd.ReactiveForwarding packetOutOnly true