RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPERIEURE
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ SAAD DAHLEB BLIDA

FACULTÉ DES SCIENCES

DÉPARTEMENT INFORMATIQUE

MÈMOIRE DE FIN D'ÉTUDES

Pour l'obtention

D'un diplôme Master en informatique

Spécialité : Systèmes Informatiques et Réseaux

THÈME

Étude et implémentation de l'approche SDN-Security Dans un réseau local

Réalisé par :

GAMIR HOYAM

Encadré par : Mme. NEMMAR Mehdia

Organisme d'accueil : Sonatrach

JURY :

Mme. GHEBGHOUB YASMINA	Université de Blida 1	Président
M. CHERIF ZAHAR	Université de Blida 1	Examinateur
M. BENYAHIA MOHAMED	Université de Blida 1	Promoteur

Promotion : 2022/2023

Dédicace

C'est avec profonde gratitude et sincères mots,

Que je dédie ce modeste travail de fin

d'étude

A mes chers parents

Lamine et Mehenni Kahina

Qui ont sacrifié leur vie pour notre

réussite et nous ont éclairé le

Chemin par leurs conseils judicieux.

J'espère qu'un jour, je pourrais leurs

Rendre un peu de ce

qu'ils Ont fait pour

moi,

Que dieu leur prête bonheur et longue vie.

Je dédie aussi ce travail à

Ma petite sœur Nesrine,

Toutes ma famille,

Et à tous ceux qui me sont chers.

HOYAM

Remerciement

Ma reconnaissance ainsi que ma dévotion se dirigent tout d'abord vers

Mon Créateur

Allah le tout puissant et miséricordieux. Qui m'a donné

La force et la patience d'accomplir ce modeste travail.

En second lieu, je remercie très chaleureusement mon encadreur

Madame Nemmar Mehdia pour ses conseils, pour sa spontanéité durant toute cettepériode d'encadrement.

Je remercie également mon promoteur monsieur BENYAHIA

Pour la confiance et pour avoir accepté de diriger ce travail.

Aussi, mes vifs remerciements aux membres du jury pour l'intérêt accordé à

Ce modeste travail en l'examinant minutieusement et avec attention.

Merci

Résumé

Le Software-Defined Networking est une technologie émergente qui révolutionne la gestion des réseaux en offrant une flexibilité accrue aux administrateurs réseau. Une norme populaire associée au SDN est OpenFlow, qui permet un contrôle centralisé et programmable du réseau en séparant le plan de contrôle du plan de données. Toutefois, la sécurité reste une préoccupation majeure dans le contexte du SDN. Les réseaux programmables nécessitent des mécanismes de sécurité spécifiques pour garantir une protection efficace contre les intrusions et les attaques.

Pour répondre à ce défi, notre objectif est de concevoir une solution de sécurité programmable qui tire parti des principes du SDN et intègre les fonctionnalités d'un pare-feu traditionnel. En utilisant les capacités de centralisation du contrôle et de virtualisation des ressources offertes par le SDN, nous pourrons créer une architecture de sécurité plus souple et dynamique. En associant cette architecture à des mécanismes d'automatisation, nous serons en mesure de contrôler et de protéger le trafic réseau de manière efficace au sein d'un environnement défini par logiciel.

Mots clefs : SDN, OpenFlow, pare-feu.

Abstract

Software-Defined Networking (SDN) is an emerging technology that is revolutionizing network management by providing increased flexibility to network administrators. A popular standard associated with SDN is OpenFlow, which allows centralized and programmable network control by separating the control plane from the data plane. However, security remains a major concern in the context of SDN. Programmable networks require specific security mechanisms to guarantee effective protection against intrusions and attacks.

To meet this challenge, our goal is to design a programmable security solution that takes advantage of the principles of SDN and integrates the functionalities of a traditional firewall. By using the centralized control and resource virtualization capabilities offered by SDN, we can create a more flexible and dynamic security architecture. By combining this architecture with automation mechanisms, we will be able to effectively control and protect network traffic within a software-defined environment.

Keywords : SDN, OpenFlow, firewall.

ملخص

الشبكات المعرفة بالبرمجيات هي تكنولوجيا ناشئة تحدث ثورة في إدارة الشبكات من خلال توفير مرونة أكبر لمسؤولي الشبكات. واحدة من المعايير الشهيرة المرتبطة بشبكات SDN هي OpenFlow ، والتي تتيح التحكم المركزي والقابل للبرمجة في الشبكة من خلال فصل الخطة التحكمية عن الخطة البيانية. ومع ذلك، ال تزال ألمان مشكلة رئيسية في سياق شبكات SDN. تتطلب الشبكات القابلة للبرمجة آليات أمان محددة لضمان حماية فعالة ضد االختراقات والهجمات.

للتغلب على هذا التحدي، يكمن هدفنا في تصميم حل أمان قابل للبرمجة يستفيد من مبادئ شبكات SDN ويدمج ميزات جدار الحماية التقليدي. باستخدام قدرات التحكم المركزي وتجاوز الموارد المقدمة من قبل شبكاتSDN ، سنتمكن من إنشاء بنية أمان أكثر مرونة وديناميكية. من خالل ربط هذه البنية بآليات التشغيل التلقائي، سنكون قادرين على التحكم في حركة المرور على الشبكة وحمايتها بكفاءة داخل بيئة معرفة بالبرمجيات.

كلمات اساسية: شبكات معرفة بالبرمجيات ,أوبن فلو, جدار الحماية

Table des matières

1 Introduction générale .....

Chapitre 1 : Limitation des réseaux traditionnels et le paradigme SDN.....3

1.1. Introduction 3

1.2. Les réseaux traditionnels.....3

1.3. Limitations des réseaux traditionnels et le besoin de les faire évoluer.....4

1.4. Le SDN 5

1.5. Comparaison entre SDN et le réseau traditionnel 6

1.6 Architecture SDN.....7

1.6.1 Le plan de transmission 7

1.6.2 Le plan de Contrôle 7

1.6.3 Le plan application 8

1.6.4 Les interfaces de communication 8

1.7 OpenFlow.....10

1.7.1 Commutateur OpenFlow.....10

1.7.1.1 Table de flux.....12

1.7.1.2 Table de Groupe 13

1.7.2 Canal OpenFlow.....14

1.7.3 Messages OpenFlow.....14

1.7.4 Contrôleur SDN.....15

1.8 Les implémentations Spécifique du SDN.....17

1.9 Conclusion 18

Chapitre 2 : SD-FW : Sécurité Programmable pour les réseaux SDN.....19

2.1. Introduction 19

2.2. Fondements du SD-FW.....19

2.2.1 Avantages SD-FW.....19

2.2.2 Firewall Traditionnel vs SD-FW.....20

2.3. Composants SD-FW 22

2.3.1 Controleur SD-FW 22

2.3.2 Data Plane 22

2.3.3 OpenFlow.....23

2.4 Mise en place et fonctionnalité SD-FW.....25

2.4.1 Déploiement SD-FW 25

2.4.2 Phase Opérationnelle SD-FW : (fonctionnalité).....26

Table des matières

2.5 Défis et considération du SD-FW 27
2.6 Travaux Connexe 28
2.7 Conclusion 29
Chapitre 3 : Performance des contrôleurs SDN : Analyse comparative entre RYU ET ODL 30
3.1. Introduction 30
3.2. Contrôleur SDN 30
3.2.1. RYU 30
3.2.2. OpenDayLight 31
3.3. Emulateur SDN 32
3.3.1. MININET 32
3.4. OpenvSwitch 32
3.5. Evaluation des performances RYU et ODL 33
3.5.1. CBench emulation 34
3.5.2. Configuration du banc d'essai 35
3.5.3 Tests de débit 36
3.5.4 Tests de latence 39
3.5.5 Discussion 41
3.6 Conclusion 42
Chapitre 4 : Implémentation d'un Firewall Distrbué basé sur OpenFlow : Architecture, Intégration et Performance 43
4.1. Introduction 43
4.2. Conception de l'architecture SD-FW de l'organisme d'accueil 43
4.2.1. Zone Serveurs 44
4.2.2. Zone Utilisateurs 44
4.2.3. Vlan 44
4.2.4 API Rest 45
4.3. Application Firewall Distribué basé OpenFlow 46
4.4. Expérimentation 49
4.4.1. Connexion de la topologie au controleur Ryu 49
4.4.2. Expérience 1 : Implémentation du Firewall Distrbué au niveau couche Access 50
4.4.3 Expérience 2 : Implémentation du Firewall Distrbué au niveau couche Distribution 53
4.4.4 Expérience 3 : Implémentation du Firewall Distrbué au niveau couche Core 55
4.5. Comparaison des résultats de performance pré et post-implémentation 58

4.6. Conclusion 63

Conclusion générale 64

Annexes A : MININET 66

Annexes B : OpenVswitch 68

Annexes C : Contrôleur SDN 69

Annexes D : Organisme d'accueil 73

Bibliographie 77

Liste des Figures

Figure 1 : Fonctionnements des équipements réseaux traditionnels 4
Figure 2 : Architecture bref du SDN 5
Figure 3 : Comparaison entre architecture SDN et réseau traditionnels 6
Figure 4 : Les interfaces SDN 8
Figure 5 : L'architecture SDN avec (a) les plans, (b) les couches et (c) l'architecture du design dusystème 9
Figure 6 : Le réseau OpenFlow 10
Figure 7 : Architecture d'un commutateur OpenFlow 11
Figure 8 : Processus de transmissions des paquets au sein des commutateurs OpenFlow 12
Figure 9 : Contenu d'entrées de flux 12
Figure 10 : Entrée de groupe table 13
Figure 11 : Echanges des messages OpenFlow entre le contrôleur et le commutateur 15
Figure 12 : Pare-feu dans un réseau traditionnel 20
Figure 13 : Pare-feu dans un réseau SDN 21
Figure 14 : Structure SD-FW 24
Figure 15 : Configuration système SD-FW 25
Figure 16 : L'architecture du contrôleur RYU 31
Figure 17 : Mininet 32
Figure 18 : Scénario de test 35
Figure 19 : Configuration de la VM du contrôleur ODL 35
Figure 20 : Configuration de la VM du contrôleur Ryu 36
Figure 22 : Reconnaissance des switches 3
Figure 23 : Performance RYU 41
Figure 24 : Performance ODL 42
Figure 25 : architecture du siège DP. [Groupe Sonatrach] 43
Figure 26 : architecture SD-FW minimisée du siège DP 44
Figure 27 : Scénario APIRest 45
Figure 28 : Code Source de la topologie SDN partie 1 45
Figure 29 : Code Source de la topologie SDN partie 2 46
Figure 30 : Diagramme de démarrage de l'application 47
Figure 31 : code représentatif de la capacité d'auto-découverte des commutateurs 47
Figure 32 : **:** Mécanisme application firewallofl.py 48
Figure 33 : code représentatif de la capacité de journalisations des connexions bloqués 48
Figure 34 : Emplacement code firewallodl.py 48
Figure 35 : Topologie sous Mininet 49
Figure 36 : Ryu en cours d'exécution 49
Figure 37 : Connexion réussite entre les commutateurs et le contrôleur 50
Figure 38 : Etat initiale des 7 Firewall 50
Figure 39 : Configuration Vlan SDN 50
Figure 40 : Activé pare-feu sur switch A3 51
Figure 41 : **:** Intégration des Règles Firewall pour les Vlan 51
Figure 42 : Output Ovs A3 51
Figure 43 : Résultats du test de connectivité entre les machines h5 et h6 52

Figure 44 : Output Ovs A4 52
Figure 45 : Résultats du test de connectivité machine h8 vers h9 52
Figure 46 : Journal des paquets bloqués par le pare-feu 1 53
Figure 47 : Résultats du test de connectivité h9 vers h7 54
Figure 48 : Résultats du test de connectivité machine h7 vers h9. 54
Figure 49 : Journal des paquets bloqués par le pare-feu 2 54
Figure 50 : Output Ovs D2. 5
Figure 51 : Confirmation des règles pare-feu C1.
..... 56
Figure 52 : Output Ovs C1 56
Figure 53 : Test de Connectivité UDP 1 56
Figure 54 : Test de Connectivité TCP 1. 57
Figure 55 : Test de Connectivité TCP 2. 57
Figure 56 : Journal des paquets TCP bloqués par le pare-feu. 57
Figure 57 : Test de Connectivité UDP 2. 57
Figure 58 : **:** Journal des paquets UDP bloqués par le pare-feu. 58
Figure 59 : Ping avant l'implémentation du pare-feu distribué 59
Figure 60 : Ping après l'implémentation du pare-feu distribué 59
Figure 61 : temps de réponse avant et après l'implémentation du firewallofl sur D2. 60
Figure 62 : Variation du temps de réponse (RTT) avant et aprés l'implémentation du
pare-feudistribué 61
Figure 63 : Ubuntu sur VirtualBox 66
Figure 64 : Clonage dans mininet 66
Figure 65 : Installation mininet 67
Figure 66 : Installation mininet terminée 67
Figure 67 : Installation ovs 68
Figure 68 : Service Ovs 68
Figure 69 : Controleur ODL lancé 69
Figure 70 : Interfcace web représentant l'accueil du controleur ODL 71
Figure 71 : Connexion à l'interface web du contrôleur Opendaylight 72
Figure 72 : RYU en cours d'éxécution 73
Figure 73 : Organigramme de la Sonatrach 74
Figure 74 : Organigramme de la Division Production 75

Liste des Tableaux

Tableau 1 : Tableau comparatif entre le SDN et les réseaux traditionnels..... 6
Tableau 2 : récapitulatif des travaux connexes sur les pares-feux SDN…….29
Tableau 3 : Règles Pare-feu Couche Access 50
Tableau 4 : Règles Pare-feu Couche Distribution..... 53
Tableau 5 : Règles Pare-feu Couche Core..... 55
Tableau 6 : Aspects du pare-feu - OpenFlow vs Traditionnel 63

Liste des abréviations

API : Application Programming Interface

SDN : Software Defined Network

TCP : Transmission Control Protocol

UDP : User Datagram Protocol

ICMP : Internet Control Message Protocol

OVS : OpenvSwitch

SMB : Server Message Block

VLAN : Virtual Local Area Network

DHCP : Dynamic Host Configuration Protocol

DNS : Domain Name System

JVM : Java Virtual Machine

SBI : Sounthbound Interface

NBI : Northbound Interface

SD-FW : SDN Firewall

IP : Internet Protocol

WAN : Wide Area Network

RTT : Round-trip delay

INTRODUCTION GENERALE :

L'architecture réseau traditionnelle a été le fondement de la connectivité et des communications pendant de nombreuses années. Cependant, l'évolution d'Internet, l'explosion des services d'information et l'émergence de nouvelles technologies ont révélé des limites majeures de cette architecture classique en termes de gestion complexe, de flexibilité limitée et de difficulté à adapter les politiques de sécurité aux besoins actuels. Face à ces défis, le paradigme émergent du Software Defined Networking a émergé comme une solution prometteuse pour repenser l'architecture réseau et renforcer la sécurité.

La problématique majeure de l'architecture réseau traditionnelle réside dans sa complexité de gestion accrue et sa faible adaptabilité aux nouvelles exigences du monde connecté. La configuration et la gestion des réseaux traditionnels nécessitent des interventions manuelles sur chaque équipement, ce qui limite leur évolutivité et leur flexibilité. De plus, la mise en place de politiques de sécurité cohérentes et efficaces à travers l'ensemble du réseau traditionnel peut s'avérer difficile en raison de la dispersion des contrôles de sécurité.

Le SDN, en revanche, propose une approche radicalement différente en séparant le plan de contrôle du plan de données. Cette séparation permet de centraliser et de programmer la gestion du réseau, offrant ainsi une flexibilité accrue et une automatisation des tâches de gestion. Avec le SDN, les politiques de sécurité peuvent être personnalisées plus facilement et déployées de manière cohérente sur l'ensemble du réseau. De plus, le SDN offre une meilleure visibilité du réseau, ce qui facilite la détection des menaces de sécurité et la réponse aux incidents.

Cependant, la sécurité demeure une préoccupation essentielle dans le contexte du SDN. Les réseaux basés sur le SDN nécessitent des mécanismes de sécurité adaptés pour garantir une protection efficace contre les attaques et les intrusions. C'est dans ce contexte que les pare-feu SDN jouent un rôle crucial.

En fusionnant les fonctionnalités du SDN et des pare-feu traditionnels, les pare-feu SDN offrent une approche plus souple et évolutive de la sécurité réseau. Ils permettent de centraliser et de simplifier la gestion des politiques de sécurité, tout en bénéficiant des avantages du SDN en termes de flexibilité et de programmabilité. Cette combinaison offre une meilleure protection contre les attaques et une capacité de réponse plus rapide face aux menaces émergentes.

OBJECTIF DE L'ORGANISME D'ACCUEIL :

L'entreprise SONATRACH a identifié la nécessité d'améliorer la gestion de son système réseau en adoptant une approche centralisée. Dans cette optique, elle a choisi d'adopter une architecture SDN, une alternative moderne, économique et plus facile à administrer que l'architecture traditionnelle. Cette démarche permet à SONATRACH de répondre à ses besoins de manière plus efficace tout en réduisant les coûts opérationnels.

Dans sa démarche d'adoption de l'architecture SDN, SONATRACH accorde une grande importance à la sécurité de son réseau. À cet égard, l'intégration de pare-feu constitue un élément clé de cette transition. Les pares-feux sont déployés dans le cadre de l'architecture SDN afin de renforcer la sécurité globale du réseau de SONATRACH. Ils agissent comme une couche de défense supplémentaire en contrôlant et en filtrant le trafic réseau, ce qui permet de protéger efficacement les infrastructures contre les attaques potentielles.

Cette combinaison d'une architecture SDN et de pare-feu offre à SONATRACH une solution globale qui lui permettra de bénéficier d'une gestion centralisée efficace tout en assurant un niveau de sécurité plus élevé pour ses infrastructures.

PLAN DU MEMOIRE :

Ce mémoire retrace les différentes étapes qui ont donné lieu à cette solution. Des recherches bibliographiques ont été menées sur l'approche SDN, Ainsi, Ce travail est structuré en quatre chapitres suivit d'une conclusion générale, à savoir :

- Chapitre 1 : Une revue de littérature sur le paradigme du Software Defined Networking.
- Chapitre 2 : Se concentre sur l'intersection entre le paradigme SDN et Firewall
- Chapitre 3 : Une étude comparative entre deux contrôleurs SDN populaires : Ryu et OpenDaylight L'objectif est d'évaluer les performances de ces deux contrôleurs dans le contexte d'une application Firewall.
- Chapitre 4 : Une vision détaillée de l'intégration de l'application firewall dans l'architecture SDN proposée de SONATRACH, en mettant en évidence les considérations architecturales, les mécanismes de distribution des fonctionnalités de pare-feu et les résultats de l'étude de performance.

Dans la conclusion générale, nous allons récapituler les principaux résultats obtenus et présenter quelques perspectives pour des travaux futures.

Chapitre 1

Limitations des réseaux traditionnels et le paradigme SDN

1.1 Introduction :

Au fil des années, l'architecture réseau traditionnelle a été la norme pour connecter les ordinateurs et les appareils dans les entreprises et les organisations. Cependant, cette architecture a des limites qui peuvent affecter les performances et la sécurité du réseau. Avec l'explosion des données et la demande croissante de performances élevées, les limites de l'architecture traditionnelle deviennent de plus en plus évidentes.

Le paradigme Software-Defined Networking est une nouvelle approche qui vise à résoudre ces limitations en fournissant une meilleure gestion du réseau. Le SDN permet aux administrateurs de réseau de contrôler et de configurer le réseau de manière centralisée, ce qui facilite la gestion et l'optimisation des performances.

SDN est une technologie qui a gagné en popularité auprès des entreprises et des fournisseurs de services réseau, notamment **Google**, l'un des premiers adoptants du SDN et a utilisé cette technologie pour gérer son réseau interne [1].

Dans ce chapitre, nous allons introduire brièvement les réseaux traditionnels et le besoin des réseaux programmables. Nous présentons le paradigme SDN, ainsi, nous citons les principaux avantages et défis à surmonter grâce à l'intégration du SDN.

1.2 Les réseaux traditionnels :

Les réseaux traditionnels, également connus sous le nom de réseaux basés sur le matériel, sont des réseaux informatiques où la gestion des équipements réseau tels que les routeurs, les commutateurs, les pare-feu et autres appareils se fait localement sur chaque appareil, à l'aide de lignes de commande ou d'interfaces graphiques utilisateur (GUI) fournies par le fabricant.

La principale caractéristique des réseaux traditionnels se compose d'éléments de mise en réseau qui possèdent un couplage des deux plans, à savoir : Le plan de contrôle autrement dit l'intelligence du boitier, il est responsable mécanismes de transmission de couche 2 et 3, notamment la table de voisinage, la table topologique, la table de routage, etc. Le plan de données, quant à lui, est responsable de la transmission des données à travers le réseau en fonction des décisions prises par le plan de contrôle. Ce qui signifie que la gestion de réseau est distribuée sur plusieurs équipements de mise en réseau.

Cette architecture réseau peut être efficace pour les petits réseaux, mais elle peut devenir difficile à gérer et à maintenir pour les réseaux de grande taille et complexes.

La figure ci-dessous montre l'architecture des équipements réseau traditionnels et leurs fonctionnements.

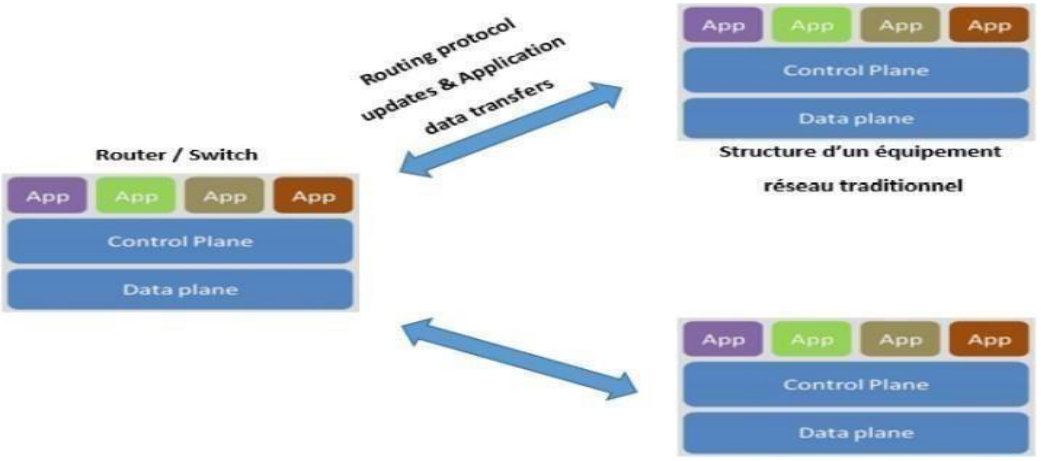

Figure 1 : Fonctionnements des équipements réseaux traditionnels [2].

1.3 Limitations des réseaux traditionnels et le besoin de les faire évoluer :

Les architectures des réseaux traditionnels ont été conçues pour répondre aux besoins de leur époque. Aujourd'hui, les applications sont souvent hébergées dans le cloud et sont accessibles de n'importe où dans le monde. Les données sont plus volumineuses et plus complexes, ce qui a augmenté la demande de bande passante et de performances de réseau. La sécurité est devenue une préoccupation majeure avec l'augmentation du nombre de menaces de sécurité et de cyberattaques.

Les réseaux traditionnels ne sont pas conçus pour répondre à ces exigences actuelles,Voici quelques-unes des limitations des réseaux classiques [3] :

- Complexité de la gestion de réseau : Dans les réseaux traditionnels, la gestion de réseau est distribuée sur plusieurs équipements de mise en réseau, ce qui rend la gestion de réseau complexe et difficile. Les administrateurs doivent configurer et gérer chaque équipement de manière indépendante, ce qui peut entraîner des erreurs et une inefficacité dans la gestion de réseau.

- Manque de flexibilité et d'agilité : Les réseaux traditionnels sont souvent rigides et difficiles à faire évoluer pour répondre aux besoins changeants de l'entreprise. Les processus manuels de configuration et de mise à jour des équipements de réseau peuvent prendre beaucoup de temps, ce qui peut retarder les changements nécessaires pour répondre aux exigences de l'entreprise.

- Faible visibilité du réseau : Les réseaux traditionnels peuvent avoir une visibilité limitée sur l'état du réseau, ce qui peut rendre la détection et la résolution des problèmes difficiles. Les équipements de mise en réseau peuvent ne pas fournir

Suffisamment d'informations sur les performances du réseau, ce qui peut rendre difficile la détection des goulets d'étranglement ou des problèmes de sécurité.

- Difficulté de mise en œuvre de politiques de sécurité : Les réseaux traditionnels peuvent être difficiles à sécuriser car la mise en œuvre des politiques de sécurité Nécessite une configuration manuelle sur chaque équipement de réseau. Cela peut rendre la gestion de la sécurité inefficace et difficile à maintenir.

Pour faire face à ces défis, les nouvelles architectures de réseau telles que le SDN proposent une approche plus centralisée et programmable de la gestion de réseau. Le SDN permet une gestion de réseau plus flexible et agile, une visibilité améliorée sur l'état du réseau, et une mise en œuvre plus facile des politiques de sécurité.

1.4 Le SDN :

Le SDN est l'acronyme du Software Defined Networking, ou bien réseau définie par logiciel. Selon [4], 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.

Les réseaux définis par logiciels (SDN) ont introduit la séparation du plan de contrôle du plan de données (voire la figure ci-dessous), 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 En ce qui concerne le plan de données réside encore sur le commutateur ou le routeur [5].

Entre ces derniers, SDN utilise une interface de communication ouverte appelée OpenFlow que nous allons détailler plus tard.

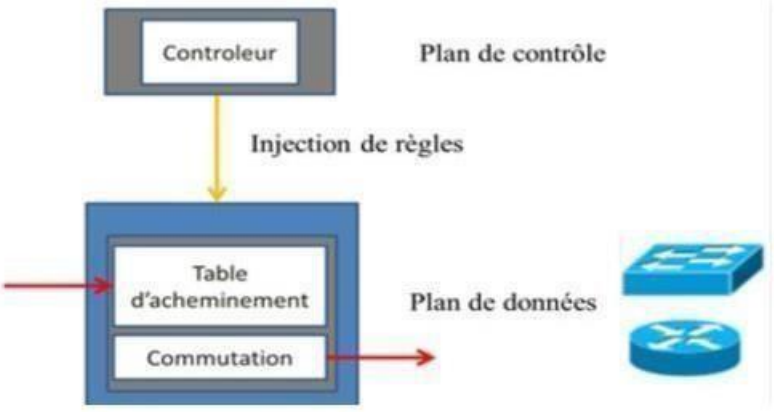

Figure 2 : Architecture bref du SDN [6].

1.5 Comparaison entre SDN et le réseau traditionnel : Les réseaux traditionnels et les réseaux SDN diffèrent dans la façon dont ils gèrent le trafic réseau et le contrôle des commutateurs réseau. Voici un tableau comparatif entre ces derniers :

Aspect de comparaison	Réseaux traditionnels	Réseaux SDN
Architecture	Basée sur une architecture en trois couches (accès, distribution, cœur)	Sépare la couche de contrôle de lacouche de transfert
Contrôle	Décentralisé, chaque dispositif prend des décisions de traitement de paquets Indépendamment	Centralisé, un contrôleur de réseau prenddes décisions de traitement des paquets
Programmabilité	Programmés via des CLI ou des GUI spécifiques à chaque dispositif	Programmables via des API standardisées, Ce qui permet aux développeurs de personnaliser le comportement du réseau
Séparation des couches	La couche de contrôle est intégrée avec la couche de transfert	La couche de contrôle est séparée de lacouche de transfert
Automatisation	Peut être difficile à automatiser en raison de l'absence de standardisation et du contrôle décentralisé	Facile à automatiser grâce à la programmablité et au contrôle centralisé
Flexibilité	Peut manquer de flexibilité dans la gestion du réseau	Offre une plus grande flexibilité dans lagestion du réseau
Coût	Peut-être moins coûteux pour de petits environnements	Peut nécessiter un matériel et des logiciels spécifiques
Innovation	Environnement de Tests suffisants	Environnement de Tests Limités

Tableau 1 : Tableau comparatif entre le SDN et les réseaux traditionnels.

En ce qui concerne la différence architecturelle entre SDN et réseau classique est illustrée dans la figure ci-dessous.

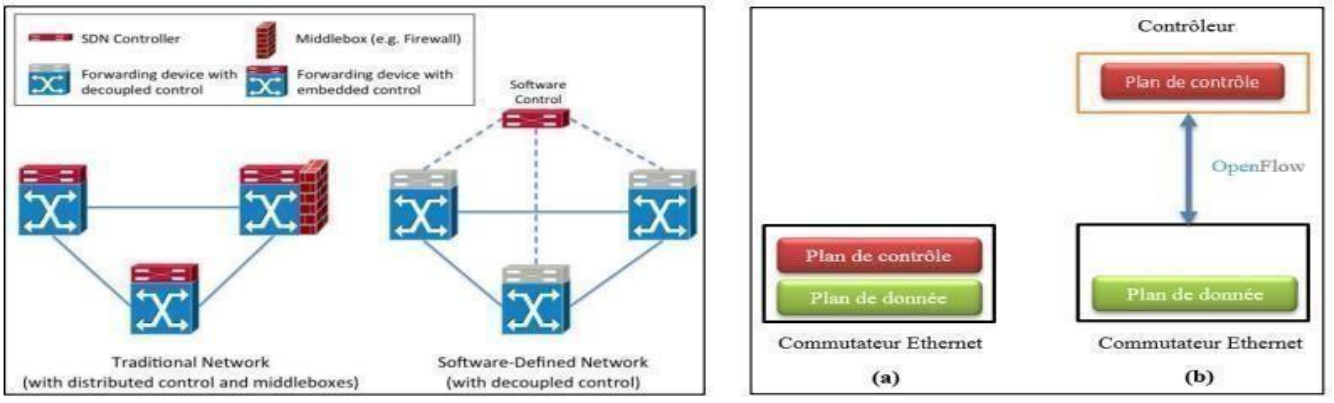

Figure 3 : Comparaison entre architecture SDN et réseau traditionnel [7].

1.6 Architecture SDN :

L'architecture SDN est conçue pour ouvrir le réseau aux applications en permettant une plus grande programmabilité, une meilleure intégration avec les applications et une automatisation plus efficace des tâches de gestion réseau.

La structure de SDN se compose de trois parties principales [8]. Le niveau le plus bas, inclut le Data plane, au plus haut niveau dont l'existence de l'application plane, et le control plane se trouve entre les deux.

La communication entre les contrôleurs et le Data plane est gérée via le SBI (South- Bound Interface). Et la communication entre les applications et le contrôleur est assurée par NBI (Nourth-Bound Interface) [9].

La figure suivante illustre l'architecture du SDN, et les détaille seront donnés dans les sections qui suivent :

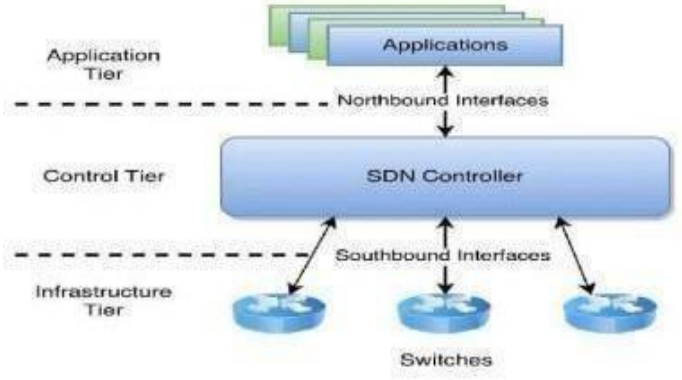

Figure 3 : Architecture bref du SDN [9].

1.6.1 Le plan de transmission (Data plane) :

C'est la couche la plus basse. Elle sert à présenter les dispositifs de commutation comme le commutateur, le routeur..., dans le plan de donnée. Ces dispositifs ont deux fonctions bien distinctes. Premièrement, ils sont responsables de la collecte des statuts du réseau, ils vont les stocker temporairement dans les dispositifs locaux puis ils les envoient au contrôleur. Les statuts du réseau peuvent contenir des informations comme la topologie du réseau, statistique des trafics, et l'usage du réseau. Deuxièmement, ils sont responsables du traitement de paquets basés sur les règles fournis par le contrôleur.

1.6.2 Le plan de contrôle (Control plane) :

Il sert en tant que pont entre la couche infrastructure et la couche application via ses deux interfaces (southbound interface et northbound interface).

Le plan de contrôle est chargé d'indiquer au plan de transmission comment traiter les paquets du réseau, en traduisant les requêtes faites par les applications en langage compréhensible par les composants d'infrastructure réseau et inversement, il remonte aux applications SDN les événements détectés par les composants actifs.

1.6.3 Le plan d'application (Management plane) :

Elle contient les applications SDN qui sont désignées pour accomplir les besoins de l'utilisateur. A travers la plateforme programmable fournit par la couche de contrôle, vise à garantir que l'ensemble du réseau fonctionne de manière optimale en communiquant avec le plan de contrôle [10].

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 dernierstemps, les algorithmes remplacent la plupart des interventions humaines. Les fonctionnalités du plan de gestion comprennent généralement les éléments suivants :

- La gestion de la configuration.
- La gestion des pannes et de la surveillance.

1.6.4 Les Interfaces de communications :

Il existe principalement trois types d'interfaces permettent aux contrôleurs de communiquer avec leur environnement : interface Sud, Nord et Est/Ouest.

- **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, servent à programmer les équipements de transmission, en exploitant l'abstraction du réseau fourni par le plan de contrôle. L'avantage le plus précieux du SDN provient de sa capacité à prendre en charge et à permettre des applications innovantes.

- **Interface Sud**

Les interfaces Sud ou (South-Bound APIs) représentent les interfaces de communication, qui permettent au contrôleur SDN d'interagir avec les équipements de la couche d'infrastructure, tel que les switches, et les routeurs.

Le protocole le plus utilisé, et le plus déployé comme interface Sud est le protocole OpenFlow, qui a été standardisé par l'ONF. Sa dernière version est 1.5, plus de détails sur ce protocole sera donnée dans la prochaine section. Il existe plusieurs d'autres alternatives d'interface Sud, tels que Forces, ou OpenVswitch, Data base (OVSDB), mais le protocole OpenFlow est actuellement le standard de facto, qui est largement accepté et répandu dans les réseaux SDN.

- **Interfaces Est/Ouest**

Les interfaces Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs, pour synchroniser l'état du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible [11].

La figure 4 ci-dessous, nous permet d'illustrer les interfaces Nord et Sud de l'architecture SDN, déjà expliqué ci-dessus.

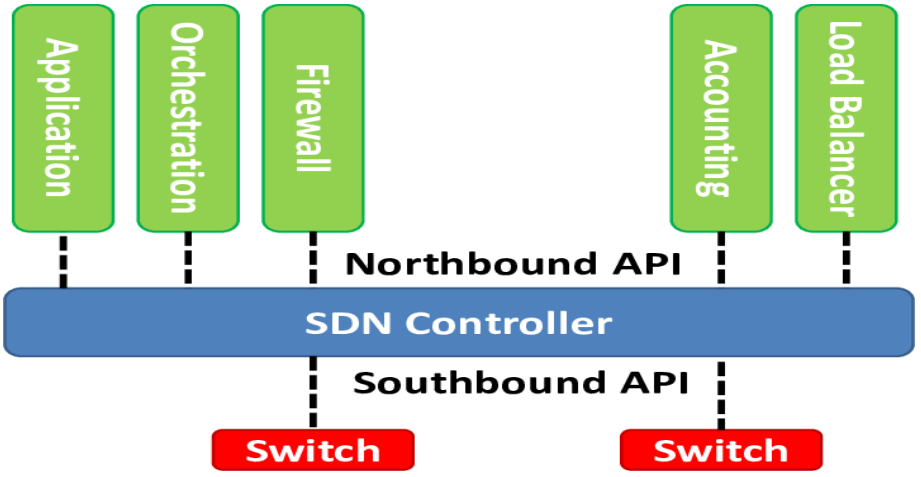

Figure 4 : Les interfaces SDN [10].

Pour plus de précision, la figure 5 suivante nous permet de décrire le réseau SDN avec les couches SDN qui sont déjà expliquées précédemment et représentées, dans le centre (b) de la figure où il y a les applications réseaux, langage de programmations, virtualisation basé sur les langages, interface orientée nord, système d'exploitation du réseau, hyperviseur de réseau, interface orientée sud et l'infrastructure du réseau. A l'extrémité gauche (a) représente une vue orientée-plan incluant le plan de gestion, le plan de contrôle et le plan des données et (c) à l'extrémité droite représente l'architecture du design du système.

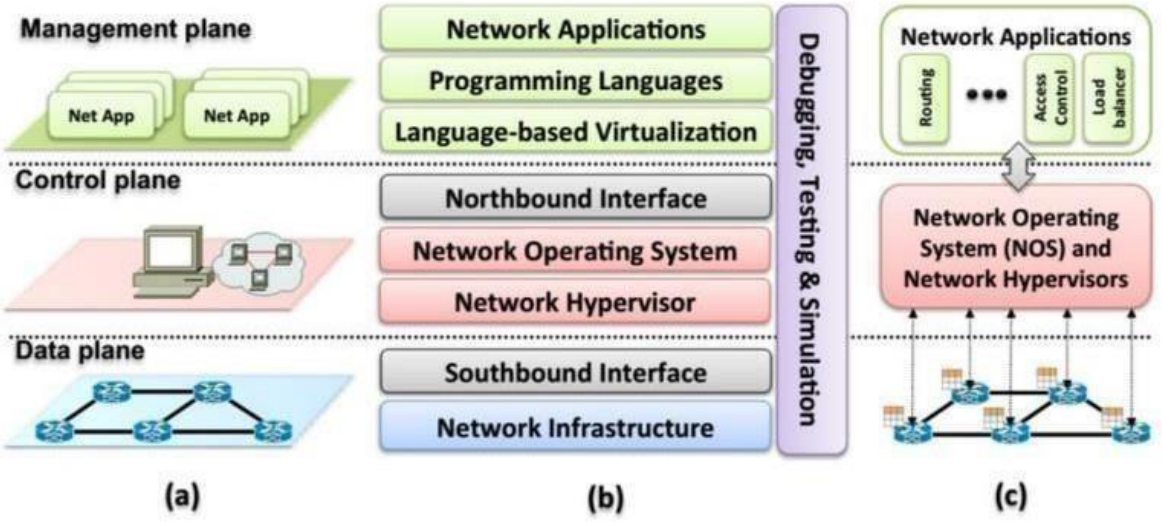

Figure 5 : L'architecture SDN avec (a) les plans, (b) les couches et (c) l'architecture du design du système [12].

1.7 OpenFlow :

OpenFlow est le protocole de communication le plus avancé entre un plan de contrôle (logiquement centralisé) et le plan de données, Il est standardisé par l'Open Networking Fondation (ONF) et implémenté par de nombreux équipementiers, dont HP, Cisco, IBM, et Juniper [27].

Il utilisé pour la gestion centralisée et programmable des réseaux de données. Ce dernier, il permet aux contrôleurs de réseau de transmettre des instructions aux commutateurs de réseau pour programmer leur plan de données et obtenir des informations sur l'état du réseau.

La figure suivante illustre les éléments principaux d'un réseau OpenFlow, à savoir : les commutateurs OpenFlow, les contrôleurs et finalement le protocole OpenFlow qui définit les messages échangés entre les deux types d'équipements et permet de standardiser la communication.

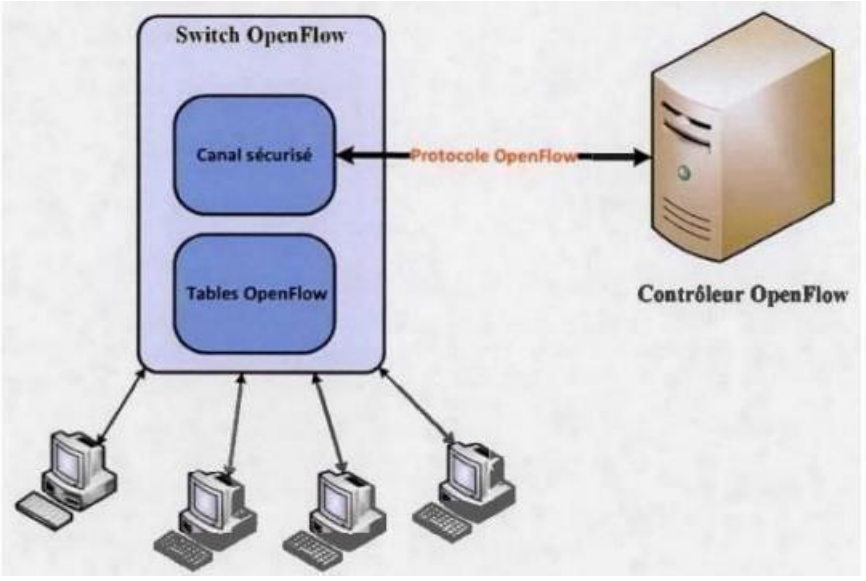

Figure 6 : Le réseau OpenFlow [28].

Il est important de noter que l'utilisation d'OpenFlow nécessite un commutateur de réseau qui prend en charge le protocole OpenFlow. De nombreux commutateurs de réseau modernes sont compatibles avec OpenFlow.

1.7.1 Commutateur OpenFlow :

Par abus de langage, tous les équipements de transmission SDN soient appelés Switch, ils sont programmés à partir d'un contrôleur SDN et offrent des fonctionnalités plus avancées que les simples switchs traditionnels en termes de flexibilité, de scalabilité et de contrôle Un commutateur OpenFlow est composé de :

- Plusieurs tables de flux pour les correspondances des entrées.
- Une table de groupe pour l'acheminement des paquets.
- Un canal OpenFlow connecté à un contrôleur externe.

La figure suivante nous permet d'illustrer ces composants :

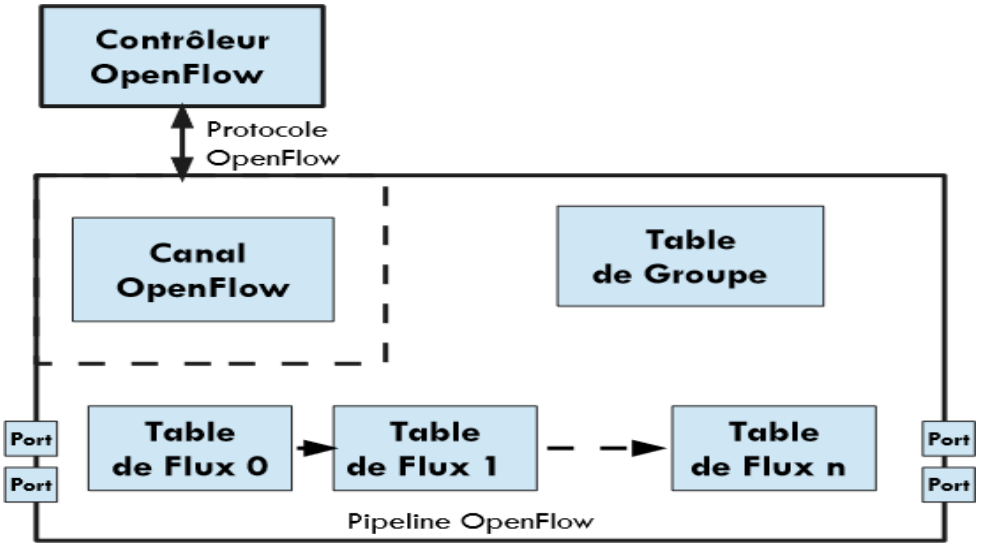

Figure 7 : Architecture d'un commutateur OpenFlow [26].

Switch SDN possède deux fonctionnalités principales, à savoir :

- Fonction de support du contrôle : Cette fonction permet au commutateur de communiquer avec le contrôleur SDN via le protocole OpenFlow. Le commutateur envoie des informations sur l'état du réseau au contrôleur, tandis que le contrôleur envoie des instructions de transfert de paquets au commutateur. Cette communication permet au contrôleur de programmer le commutateur pour prendre des décisions en temps réel en fonction des conditions du réseau.

- Fonction d'acheminement des données : Cette fonction permet au commutateur de transférer les données entre les équipements du réseau et les systèmes de terminaison. Le commutateur accepte les flux de données entrants et les relaie sur un chemin de commutation qui a été calculé et établi à partir des règles définies par les applications SDN. Ces règles sont passées au contrôleur SDN qui les redescend au commutateur pour une mise en œuvre précise.

Le fonctionnement d'un switch OpenFlow suit l'algorithme suivent [14] :

```
Receive packet;
Check flow table ;
If {
The packet rule is in the flowtable;
Send out the packet following the rule; }
Else
Send packet to the controller; }
```

La figure suivante illustre le comportement d'un commutateur OpenFlow :

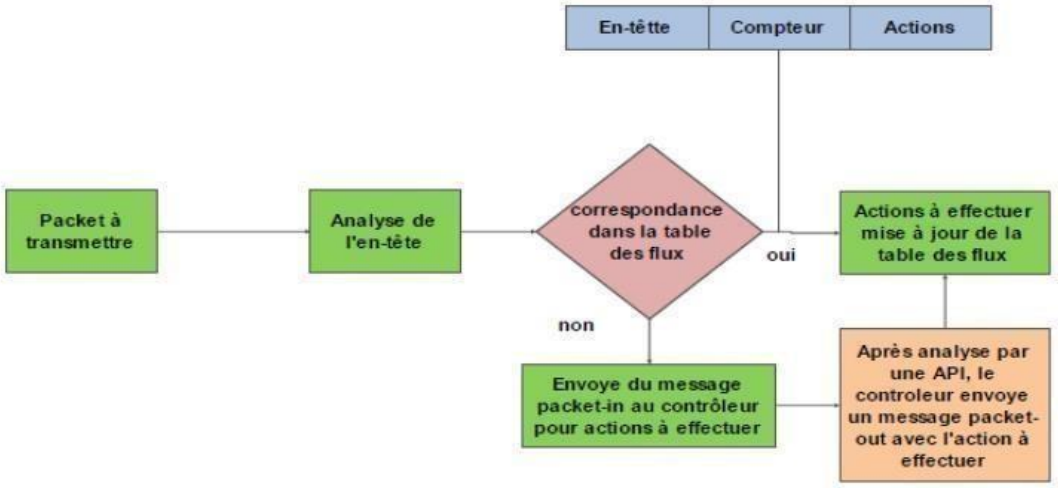

Figure 8 : Processus de transmissions des paquets au sein des commutateurs OpenFlow [56].

1.7.1.1 Table de flux :

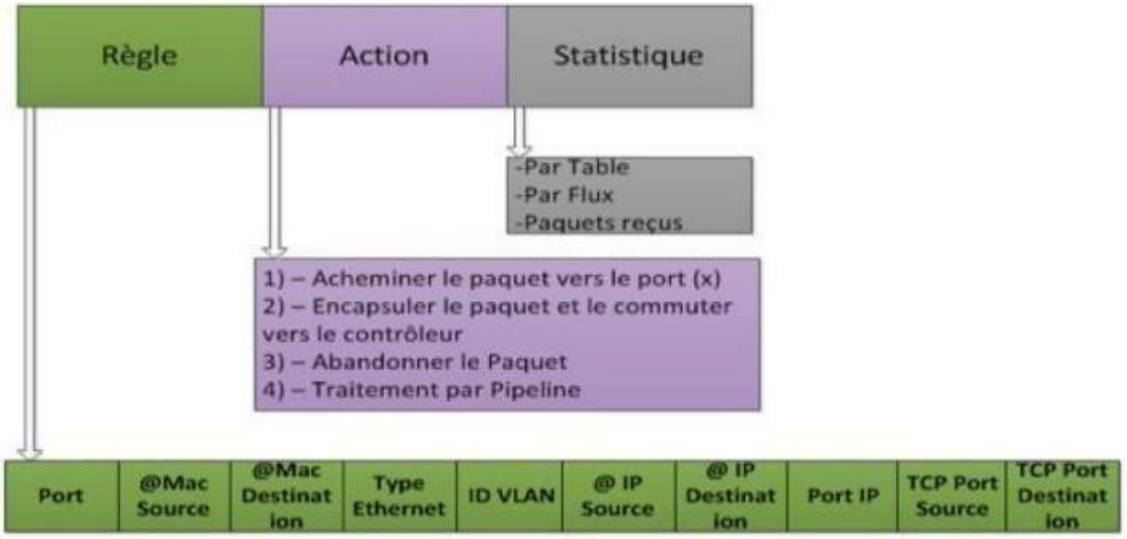

Figure 9 : Contenu d'entrées de flux [57].

Tel que le démontre la figure 9, la table de flux d'un commutateur OpenFlow est une table de correspondance qui permet de déterminer comment les paquets de données sont transférés entre les différents ports du commutateur en fonction de critères tels que l'adresse source, l'adresse de destination, le protocole de transport et d'autres critères. La table de flux est programmée par le contrôleur de réseau à l'aide du protocole OpenFlow.

La table de flux est organisée en entrées, chacune contenant une règle de flux qui décrit comment les paquets de données correspondant à cette règle doivent être transférés.

Chaque entrée de la table de flux contient les informations suivantes [57] :

- Des champs de correspondance : Ces champs spécifient les critères que le commutateur doit utiliser pour identifier les paquets de données correspondant à cette
- Règle de flux : Les critères peuvent inclure l'adresse source, l'adresse de destination, le protocole de transport, le port d'entrée et d'autres critères.
- Les compteurs de flux : ils permettent de comptabiliser les paquets et les octets correspondant à une entrée de la table de flux. Ces compteurs peuvent être utilisés pour mesurer la charge de trafic sur un port ou pour surveiller la qualité de service.
- Une action : Cette action spécifie ce que le commutateur doit faire avec les paquets de données correspondant à cette règle de flux. L'action peut être de transférer le paquet vers un port de sortie spécifique, de le laisser tomber, de le rediriger vers le contrôleur de réseau ou d'appliquer une autre action spécifique.
- Une priorité : Cette priorité indique l'ordre dans lequel les règles de flux doivent être appliquées lorsque plusieurs règles correspondent à un même paquet de données.

Lorsqu'un paquet de données est reçu par le commutateur OpenFlow, il est comparé aux règles de flux de la table de flux dans l'ordre de priorité décroissante. La première règle correspondante est alors appliquée au paquet de données. Si aucune règle ne correspond au paquet de données, il est généralement envoyé vers le contrôleur de réseau pour traitement.

1.7.1.2 Table de Groupe :

Une table de groupe dans le protocole OpenFlow est une table qui contient des entrées de groupe qui permettent de définir des actions à appliquer sur des ensembles de flux de paquets. Chaque entrée de groupe décrit une action à appliquer sur un ou plusieurs flux de paquets en utilisant une combinaison d'actions et de ports de sortie. Par exemple, plutôt que d'appliquer une série d'actions sur chaque flux de paquets individuellement, il est possible de définir des entrées de groupe qui spécifient un ensemble d'actions à appliquer sur plusieurs flux de paquets en même temps [29].

Cela permet d'améliorer l'efficacité du traitement des paquets et de réduire la charge de travail sur les contrôleurs de réseau, en définissant des règles d'actions qui peuvent être appliquées sur plusieurs flux de paquets simultanément.

Figure 10 : Entrée de groupe table [29].

La figure 12 illustre la structure d'une entrée de groupes.

- Un identifiant de groupe (Group ID) : il s'agit d'un identifiant unique associé à chaque entrée de groupe.
- Un type de groupe (Group Type) : il indique le type d'actions que doit prendre le switch pour les paquets associés à cette entrée de groupe.
- Counters : Ensemble des compteurs qui vont être mise à jour une fois qu'un paquet est traité par le groupe.
- Action Buckets : Contient une liste ordonnée de conteneurs d'actions, où chaque conteneur contient une série d'actions à exécuter et les paramètres associés.

1.7.2 Canal OpenFlow :

Le canal OpenFlow est un canal de communication bidirectionnel entre un contrôleur SDN et un switch OpenFlow. Il est utilisé pour échanger des messages entre les deux entités, tels que des commandes de configuration, des requêtes de statistiques et des notifications d'événements.

Il est établi lors de la phase d'initialisation du protocole, lorsque le switch envoie un message HELLO au contrôleur pour annoncer sa présence et établir une connexion. Une fois la connexion établie, le contrôleur et le switch peuvent échanger des messages OpenFlow pour configurer le switch, installer des flux de trafic et récupérer des statistiques de trafic.

Les messages échangés sur le canal OpenFlow sont des messages structurés, codés dans un format binaire défini par le protocole OpenFlow [58]. Ils sont utilisés pour configurer la table de flux du switch, qui détermine comment le switch traite le trafic réseau, ainsi que pour installer des règles de filtrage et de redirection de trafic.

1.7.3 Messages OpenFlow :

Les messages OpenFlow échangés entre le contrôleur et le commutateur, détaillés sur la figure 13, sont responsables de la détection des fonctionnalités, de la configuration, de la programmation du commutateur et de la récupération d'informations.

- HELLO : C'est le premier message envoyé par le switch pour annoncer sa présence et établir une connexion avec le contrôleur.
- La paire de messages FEATURES REQUEST et FEATURES REPLY utilisés par le contrôleur pour interroger le commutateur sur les capacités du switch, telles que le nombre de ports et les fonctionnalités prises en charge.
- SET_CONFIG et GET_CONFIG : Ce paire de message utilisé par le contrôleur pour configurer et récupérer les paramètres de configuration du switch, tels que le mode de traitement des paquets et le délai d'expiration des flux.
- PACKET_IN : Ce message est envoyé par le switch pour signaler au contrôleur qu'il a reçu un paquet qui ne correspond pas à une entrée dans sa table de flux.
- FLOW_MOD : Ce message est utilisé pour installer, modifier ou supprimer des entrées de flux dans la table de flux du switch.

- STATS REQUEST et STATS REPLY : Ce paire de message utilisé pour recueillir les statistiques du commutateur par le contrôleur
- BARRIER_REQUEST et BARRIER_REPLY : Ces messages sont utilisés pour synchroniser les opérations entre le contrôleur et le switch, garantissant que toutes les opérations précédentes ont été effectuées avant de poursuivre.

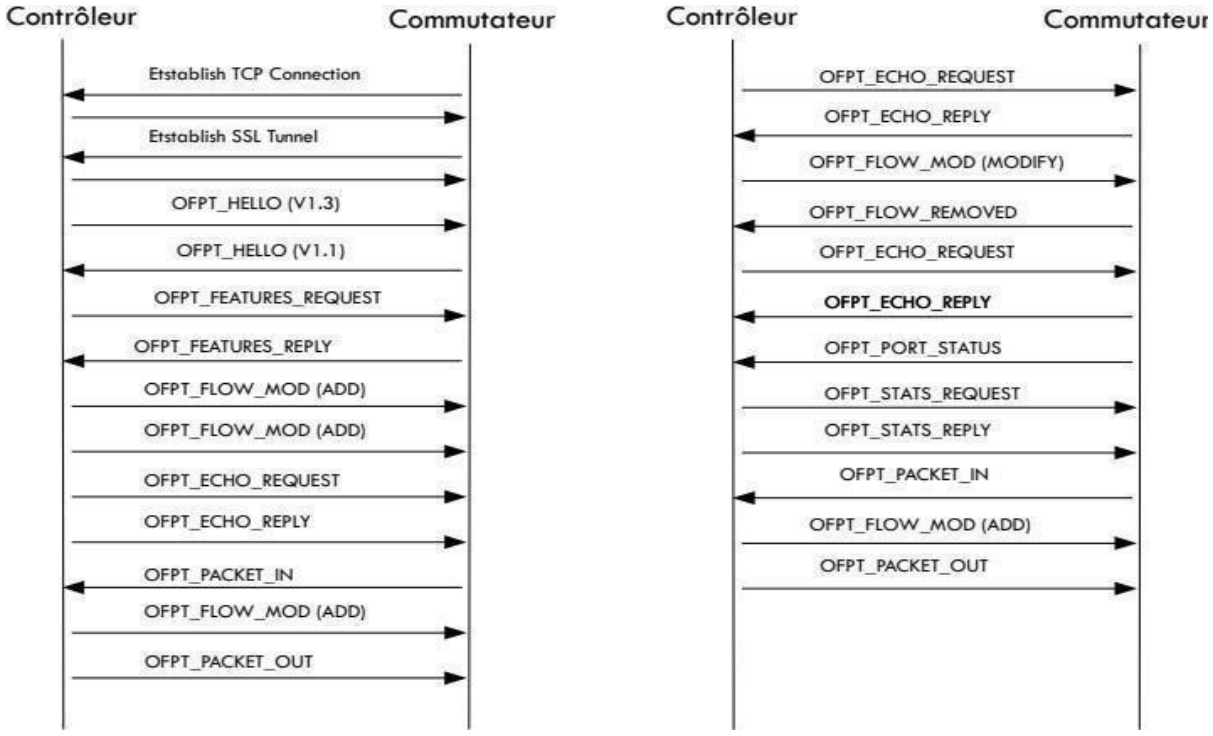

Figure 11 : Echanges des messages OpenFlow entre le contrôleur et le commutateur [59].

1.7.4 Contrôleur SDN :

Le contrôleur SDN est la partie la plus intelligente et le composant clé de l'architecture SDN, il est essentiel de prêter une attention appropriée à toute proposition ou conception d'un contrôleur SDN.

Les contrôleurs SDN communiquent avec les commutateurs SDN via un canal de commande pour mettre en œuvre des actions réseau telles que la commutation et le routage de paquets. Les commutateurs SDN sont programmables et peuvent être contrôlés à distance par le contrôleur SDN.

Les applications SDN peuvent interagir avec le contrôleur SDN pour prendre des décisions relatives au réseau, à travers la vue globale du réseau fournit par le contrôleur aux applications, qui peuvent ainsi utiliser ces informations pour prendre des décisions telles quela sélection du chemin de transfert de données, l'optimisation des performances du réseau, la sécurité et la surveillance du réseau.

Pour son fonctionnement, le contrôleur suit l'algorithme suivent [14] :

```
Receive packets from the switch
If
Non Ethernet packet        // from the switch
Packet dropped;
Else {
 Check MAC and IP address;
Send rule to the switch //flow-mod;
(Example:source:10.0.0.1- Destination:10.0.0.3-Port: 1)                    }
Switch can send the packet to the destination.
```

Quelques contrôleurs SDN : Il existe plusieurs contrôleurs SDN disponibles sur le marché. Voici quelques-uns des plus connus :

- **NOX** [16] : 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,

- **POX** [17] : 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 ore un support pour la virtualisation. Son langage de programmation est Python, orant également une plateforme de développement très rapide.

- **Floodlight** [18] : 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.

- **OpenDaylight** [15] : 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.

- **Ryu** [19] : 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 une série de fonctionnalités réutilisables pour d'autres protocoles.

- **ONOS** [20] : Ce contrôleur permet de contrôler le cloud an 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.

- **IRIS** [21] : 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.

- **Beacon** [22] : 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.

- **OpenMul** [23] **:** 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.

- **Maestro** [24] **:** 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.

1.8 Les implémentations Spécifique du SDN :

L'architecture SDN offre de nombreuses implémentations spécifiques qui permettent de répondre à des besoins variés en matière de réseau, en offrant une gestion plus flexible, évolutive, sécurisée et automatisée pour différents types de réseaux et de centres de données. En voici quelques exemples :

SD-WAN : (Software-Defined Wide Area Network) est une implémentation spécifique de SDN qui permet de combiner plusieurs connexions Internet (fibre, ADSL, 4G, etc.) pour créer un réseau WAN hybride. En utilisant le SD-WAN, les entreprises peuvent améliorer la performance de leurs applications cloud et SaaS, tout en réduisant les coûts de connectivité WAN [32].

SD-DC : (Software-Defined Data Center) est une approche de gestion de centre de données basée sur SDN. Elle utilise des technologies de virtualisation telles que la virtualisation de serveur, de stockage et de réseau, ainsi que des technologies de conteneurisation pour permettre une gestion dynamique et évolutive des ressources du centre de données [31].

SD-Access : (Software-Defined Access) est une solution innovante qui simplifié la gestion des réseaux de campus, en utilisant des techniques de virtualisation et de segmentation pour créer des zones logiques isolées, qui permettent de contrôler l'accès des utilisateurs et des périphériques. Il offre une gestion centralisée et automatisée des politiques de sécurité, d'authentification et d'autorisation, permettant aux entreprises de garantir un accès sécuriséaux ressources du réseau [30].

1.9 Conclusion :

Le SDN est une architecture de réseau révolutionnaire offrant de nombreux avantages par rapport aux réseaux traditionnels. En permettant une gestion centralisée et programmable du réseau, le SDN offre une plus grande flexibilité, une évolutivité accrue, une meilleure utilisation des ressources réseau et une simplification de la gestion du réseau.

Cependant, la sécurité du SDN constitue un défi majeur qui doit être abordé lors de la mise en place d'un réseau SDN. Dans ce contexte, le SDN Firewall émerge comme une solution essentielle pour renforcer la sécurité des réseaux définis par logiciel.

Le prochain chapitre se concentrera en détail sur le SD-FW, en explorant ses fonctionnalités, ses mécanismes de sécurité et les avantages qu'il apporte. Cette étude nous permettra de mieux appréhender l'impact du SD-FW sur la sécurisation des réseaux SDN.

Chapitre 2

SD-FW : Sécurité Programmable pour les réseaux SDN.

2.1 Introduction :

Avec l'émergence du SDN, la sécurité a dû s'adapter pour répondre aux exigences de cette nouvelle architecture programmable. Dans ce contexte, le pare-feu SDN se présente comme une avancée majeure dans le domaine de la sécurité des réseaux définis par logiciel. En combinant les capacités programmables du SDN avec les fonctionnalités de filtrage et de contrôle du trafic des pare-feu traditionnels, le pare-feu SDN offre une approche plus souple et adaptative pour garantir la sécurité des réseaux SDN.

Ce chapitre mettra en lumière le SD-FW en tant que solution de sécurité programmable, en examinant en détail ses fondements, ses avantages et ses fonctionnalités clés en matière de sécurité. De plus, nous étudierons les mécanismes de fonctionnement du pare-feu SDN, notamment l'utilisation des contrôleurs SDN pour centraliser la gestion de la sécurité.

2.2 Fondements du SD-FW :

SD-FW est un mécanisme de sécurité qui combine les principes du SDN avec les fonctionnalités d'un pare-feu traditionnel pour contrôler et protéger le trafic réseau au sein d'un environnement de réseau défini par logiciel.

Un pare-feu SDN améliore la sécurité des dispositifs réseau SDN en intégrant une couche logicielle de sécurité [46]. Il tire parti de la séparation entre le plan de contrôle et le plan de transfert de données pour permettre une gestion centralisée et programmable des politiques de sécurité.

2.2.1 Avantages SD-FW :

Les pare-feu SDN offrent de nombreux avantages grâce à leurs caractéristiques clés : [47]

- Visibilité étendue : Grâce à leur architecture centralisée, les pare-feu SDN offrent une visibilité étendue sur le trafic réseau. Cela permet une surveillance approfondie et une détection plus efficace des activités malveillantes ou non conformes.

- Gestion centralisée : une gestion centralisée des politiques de sécurité à partir d'un contrôleur logiciel. Cela simplifie la configuration et la gestion des règles de pare-feu à travers l'ensemble du réseau.

- Flexibilité : Les règles de sécurité peuvent être rapidement mises à jour et adaptées en fonction des changements dans le réseau ou des besoins de l'entreprise. Cela permet une flexibilité accrue pour s'adapter aux nouvelles menaces ou aux exigences de sécurité changeantes.

- Programmabilité : Les pare-feux SDN sont programmables, ce qui signifie que les politiques de sécurité peuvent être adaptées et ajustées plus facilement en fonction des besoins spécifiques du réseau. Les modifications de règles peuvent être appliquées rapidement et de manière flexible grâce à la nature logicielle du pare-feu SDN.

- Sécurité avancée : Les pare-feu SDN permettent la mise en œuvre de mécanismes de sécurité avancés, tels que la détection d'intrusion basée sur le comportement, la prévention des fuites de données et la segmentation du réseau. Ces fonctionnalités renforcent la protection contre les menaces potentielles.

- Automatisation et orchestration : Les pare-feu SDN peuvent être facilement automatisés et orchestrés à l'aide du contrôleur logiciel. Cela permet une gestion efficace des politiques de sécurité, des déploiements rapides et une réponse automatisée aux menaces, ce qui réduit le besoin d'interventions manuelles et accélère les opérations.

2.2.2 Firewall Traditionnel vs SD-FW :

Un pare-feu traditionnel et un pare-feu SDN diffèrent à plusieurs égards :

Tout d'abord, l'architecture est distincte. Un pare-feu traditionnel est généralement un appareil dédié, autonome, conçu spécifiquement pour la sécurité réseau. Il est souvent situé à un point central du réseau et agit comme une barrière entre le réseau interne et externe. En revanche, un pare-feu SDN fait partie d'un environnement de réseau défini par logiciel, où le contrôle est centralisé, programmable, virtualisé et exécuté sur des commutateurs réseau compatibles avec le SDN, comme les montrent les schémas ci-dessous, on peut observer la différence entre firewall traditionnelle et firewall SDN en termes d'architecture :

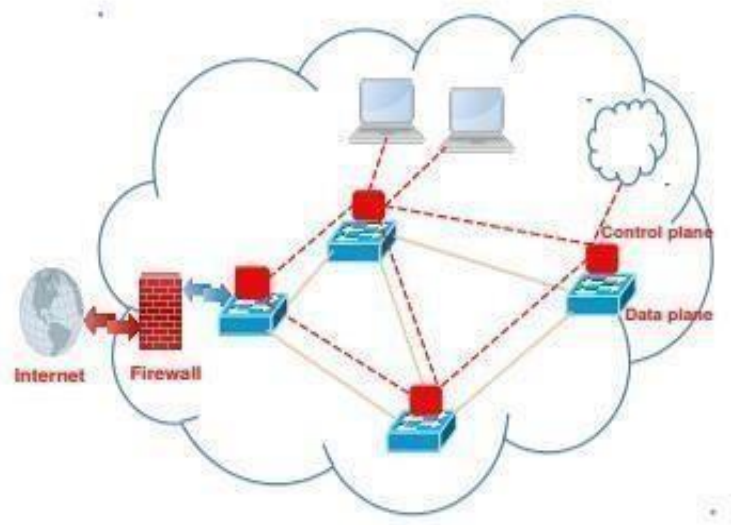

Figure 12 : Pare-feu dans un réseau traditionnel [48].

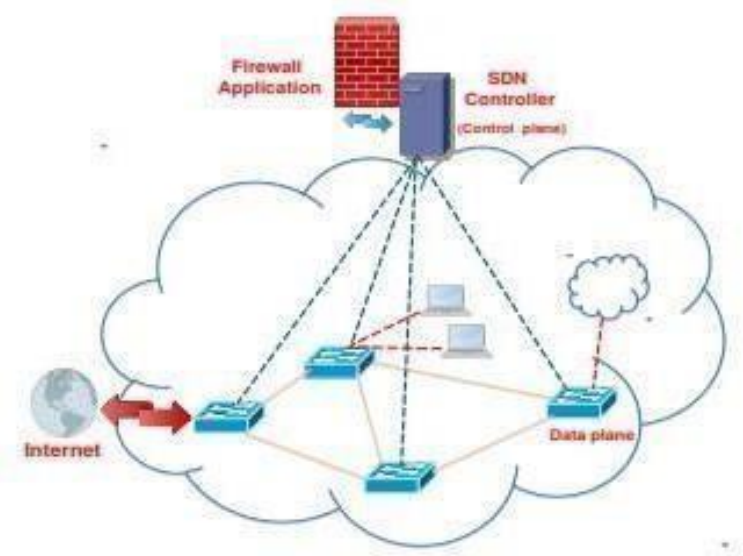

Figure 13 : Pare-feu dans un réseau SDN [48].

Une autre différence notable réside dans la gestion des politiques de sécurité. Dans un pare-feu traditionnel, les politiques de sécurité sont généralement configurées localement sur chaque appareil, ce qui peut entraîner une gestion décentralisée et complexe.

En revanche, un pare-feu SDN bénéficie d'un contrôle centralisé via un contrôleur logiciel. Les politiques de sécurité peuvent être définies et gérées de manière globale, offrant ainsi une vue d'ensemble et une gestion simplifiée du réseau.

La programmabilité est également un aspect distinctif du pare-feu SDN. Avec un pare-feu traditionnel, les fonctionnalités et les politiques de sécurité sont généralement fixes et ne peuvent pas être facilement modifiées ou adaptées. En revanche, un pare-feu SDN offre une programmabilité flexible, ce qui permet d'ajuster les politiques de sécurité en fonction des besoins du réseau. Cela permet une plus grande agilité et une réponse plus rapide aux nouvelles menaces ou aux changements de l'environnement réseau.

En termes de visibilité, un pare-feu SDN bénéficie d'une vue plus étendue sur le trafic réseau grâce à la séparation entre le plan de contrôle et le plan de transfert de données. Il peut examiner de manière approfondie les paquets et les connexions, facilitant ainsi la détection des activités malveillantes et l'application de politiques de sécurité plus précises.

Enfin, un pare-feu basé sur logiciel peut être intégré plus facilement avec d'autres services et applications réseau. Il peut fonctionner avec des systèmes de gestion des identités, des systèmes de détection d'intrusion, des systèmes de prévention des intrusions, etc. Cette intégration permet une orchestration et une automatisation plus efficaces de la sécurité réseau.

2.3 Composants SD-FW :

2.3.1 Contrôleur SD-FW :

Dans un firewall basé sur SDN, le contrôleur SDN joue un rôle central dans la gestion et le contrôle du pare-feu. Il intègre l'application pare-feu au sein de son architecture [48]. Voici les principaux rôles du contrôleur SDN dans ce contexte :

- Gestion de la politique de sécurité : Le contrôleur SDN gère la politique de sécurité du pare-feu en intégrant les règles de filtrage, d'authentification, de contrôle d'accès et autres directives de sécurité. Il fournit une interface centralisée pour configurer et contrôler ces règles, ce qui facilite leur gestion et leur mise à jour.

- Programmation des commutateurs : communique avec les commutateurs SDN du réseau pour programmer les règles de pare-feu au niveau du data plane. Il utilise des protocoles de communication tels qu'OpenFlow pour définir les actions à prendre sur les paquets en fonction des règles de sécurité spécifiées.

- Collecte d'informations sur le réseau : surveille l'état du réseau et collecte des informations sur les flux de données, les adresses IP, les protocoles utilisés, etc. Cette connaissance du contexte réseau permet au contrôleur de prendre des décisions informées lors de l'application des règles de sécurité.

- Détection d'intrusion et gestion des événements de sécurité : intègre des mécanismes de détection d'intrusion pour identifier les activités suspectes ou malveillantes. Il peut recevoir des événements de sécurité des commutateurs SDN et prendre des mesures en temps réel pour répondre aux incidents de sécurité détectés.

- Répartition de charge : Le contrôleur peut gérer la répartition de charge des flux de données en distribuant équitablement les règles de pare-feu entre les commutateurs SDN. Cela permet de réduire les goulets d'étranglement et d'améliorer les performances du pare-feu.

- Mise à jour dynamique des règles de pare-feu : Le contrôleur SDN facilite la mise à jour dynamique des règles de pare-feu en fonction des besoins du réseau. L'administrateur peut ajouter, supprimer ou modifier les règles de sécurité via le contrôleur, qui les distribue ensuite aux commutateurs SDN concernés pour une mise en œuvre immédiate.

2.3.2 Data Plane :

Le data plane est un composant essentiel d'un pare-feu basé sur SDN, responsable du traitement des paquets de données et de l'application des règles de sécurité. Les commutateurs programmables utilisés dans le data plane jouent un rôle clé dans ce processus.

Voici une description du plan de données d'un pare-feu SDN et du processus de transfert des paquets [49] :

- Architecture SD-FW: Le plan de données d'un pare-feu SDN est généralement composé de commutateurs SDN, qui agissent comme des points de contrôle de trafic. Ces commutateurs peuvent être spécifiquement conçus pour le pare-feu ou être des commutateurs génériques programmables.

- Transfert des paquets : Lorsqu'un paquet de données arrive à un commutateur SDN du plan de données, plusieurs étapes se produisent :

    1. Classification : Le commutateur examine les en-têtes du paquet (tels que l'adresse source, l'adresse de destination, le port source/destination, etc.) pour déterminer les actions à prendre sur le paquet.

    2. Correspondance des règles : Le commutateur compare les en-têtes du paquet aux tables de flux pour trouver une correspondance. Cela peut inclure des règles de filtrage, de traduction d'adresses (NAT), de chiffrement, etc. Si aucune correspondance n'est trouvée dans la table de flux d'un commutateur, celui-ci envoie une requête au contrôleur pour demander une décision. Le contrôleur peut alors programmer une nouvelle règle de flux sur le commutateur pour gérer ce type de paquet à l'avenir.

    3. Action : Une fois qu'une correspondance est trouvée, le commutateur exécute l'action spécifiée dans la règle correspondante. Cela peut inclure l'acceptation du paquet, le rejet, la modification des en-têtes, le redirigèrent vers un autre commutateur, etc.

    4. Traitement supplémentaire : Selon les politiques de sécurité définies, le pare-feu peut effectuer d'autres traitements sur les paquets, tels que la détection d'intrusion, l'inspection des charges utiles, l'analyse du trafic, etc.

    5. Transmission du paquet : Une fois que le paquet a été traité conformément aux règles de sécurité, le commutateur SDN transmet le paquet vers sa destination finale.

- Communication avec le contrôleur SDN : Pendant tout le processus de transfert des paquets, le commutateur SDN communique avec le contrôleur SDN pour obtenir des instructions, signaler des événements de sécurité et mettre à jour les règles de sécurité si nécessaire.

2.3.3 OpenFlow :

Le SD-FW et OpenFlow sont deux concepts étroitement liés dans le domaine des réseaux définis par logiciel, OpenFlow fournit le protocole de communication qui permet au contrôleur SDN d'exercer un contrôle direct sur les commutateurs réseau.

Cela permet au pare-feu SDN de programmer les règles de filtrage et de prendre des décisions en temps réel sur le traitement du trafic réseau.

Grâce à OpenFlow, le contrôleur SDN peut configurer de manière centralisée les règles de filtrage et les politiques de sécurité, qui sont ensuite distribuées aux commutateurs via le protocole OpenFlow. Cela élimine la nécessité de configurer individuellement chaque commutateur, simplifiant ainsi la gestion et la mise en œuvre des règles de sécurité.

En outre, OpenFlow permet une réactivité accrue du pare-feu SDN aux événements du réseau. Lorsqu'une nouvelle connexion est établie ou lorsqu'un comportement suspect est détecté, le contrôleur SDN peut immédiatement ajuster les règles de filtrage pour répondre à ces événements. Cela permet au pare-feu SDN d'adapter dynamiquement son comportement en fonction de la situation du réseau, renforçant ainsi la sécurité du réseau.

En combinant le contrôleur SDN, le data plane et OpenFlow, comme illustre la figure 14, le pare-feu basé sur SDN offre une gestion centralisée, une flexibilité dans la définition des règles de sécurité, une détection d'intrusion en temps réel et une mise à jour dynamique des politiques de sécurité. Cela permet d'améliorer la sécurité et les performances du réseau, tout en offrant une approche plus agile et évolutive pour la protection des systèmes d'information.

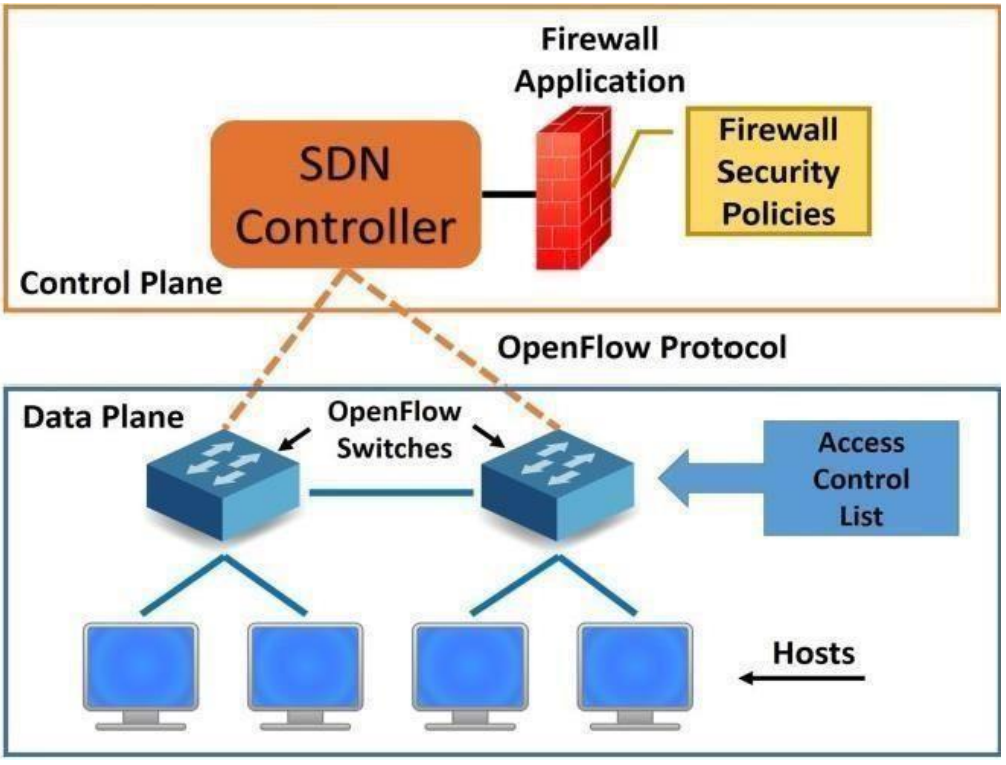

Figure 14 : Structure SD-FW [50].

2.4 Mise en place et fonctionnalité SD-FW :

2.4.1 Déploiement SD-FW :

L'étape essentielle du déploiement du système SD-FW au sein de l'infrastructure consiste à le configurer en écoutant attentivement les événements du réseau [51], la figure suivante illustre le processus de cette étape :

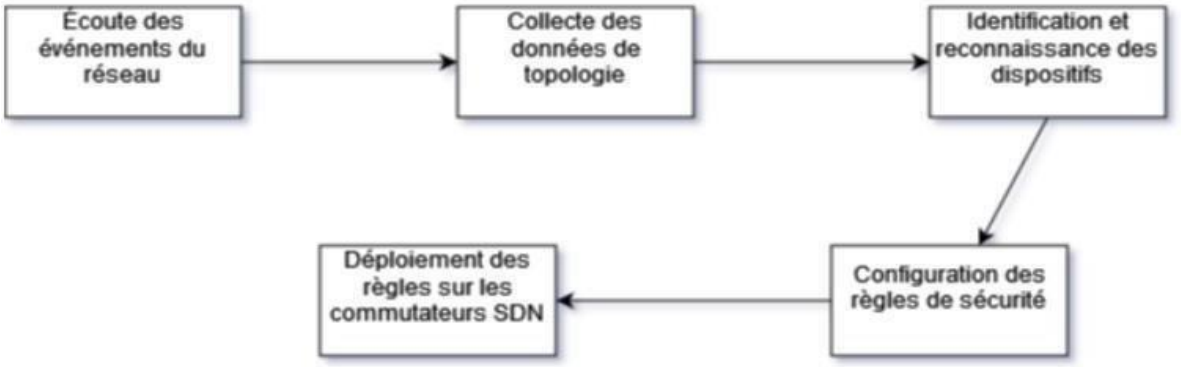

Figure 15 : Configuration système SD-FW.

Voici une explication du processus représenté :

Etape 1 : Le pare-feu SDN doit écouter les événements du réseau tels que les connexions établies, les nouveaux dispositifs connectés, les modifications de la topologie, etc. Cela peut être réalisé en utilisant les fonctionnalités de contrôle et de surveillance du contrôleur SDN. L'écoute des événements du réseau permet au pare-feu de comprendre la topologie du réseau.

Etape 2 : En écoutant les événements du réseau, le pare-feu SDN collecte les données de topologie qui décrivent la structure et les connexions du réseau. Cela inclut les commutateurs SDN, les ports, les liens entre les commutateurs, les adresses MAC et IP des dispositifs connectés, etc. Ces données de topologie sont essentielles pour prendre des décisions de filtrage appropriées.

Etape 3 : En analysant les événements du réseau et les données de topologie collectées, le pare-feu SDN est en mesure d'identifier et de reconnaître les dispositifs connectés au réseau. Il peut s'agir d'ordinateurs, de serveurs, de périphériques IoT, etc. Cette étape est importante car elle permet de différencier les dispositifs autorisés des dispositifs non autorisés et d'établir des politiques de sécurité spécifiques pour chaque type de dispositif.

Etape 4 : Une fois que le pare-feu SDN a identifié et reconnu les dispositifs connectés, il peut configurer les règles de sécurité appropriées. Cela peut inclure l'autorisation ou le blocage de certaines adresses IP ou MAC, l'application de politiques de sécurité spécifiques en fonction des caractéristiques des dispositifs, la détection d'anomalies ou de comportements suspects, etc. Les règles de sécurité sont configurées pour protéger le réseau contre les attaques et les menaces potentielles.

Etape 5 : Une fois que les règles de sécurité sont configurées, le pare-feu SDN déploie ces règles sur les commutateurs SDN du réseau. Cela permet aux commutateurs de prendre des décisions de filtrage en fonction des règles définies par le pare-feu. Les règles sont généralement distribuées et programmées dans les tables de flux des commutateurs SDN pour un traitement local, rapide et efficace du trafic.

2.4.2 Phase Opérationnelle SD-FW : (fonctionnalité)

Une fois que les règles de flux ont été programmés et que les commutateurs SDN ont été mis à jour, le pare-feu SDN entre dans sa phase opérationnelle [51]. Pendant cette phase, le pare-feu est actif et exerce ses fonctionnalités, notamment :

A. Filtrage et blocage des paquets, le processus peut être expliqué comme suit :

1. Capture des paquets : Les paquets de données transitant à travers le réseau sont capturés par les appareils du plan de données, tels que les commutateurs SDN.

2. Analyse des paquets : Les paquets capturés sont analysés en fonction des règles de filtrage définies dans le pare-feu SDN. Ces règles peuvent spécifier des critères tels que les adresses IP source/destination, les ports, les protocoles, etc.

3. Correspondance des règles : Les paquets sont comparés aux règles de filtrage qui sont traduites en tables de flux dans un ordre spécifié. Si un paquet correspond à une règle spécifique, une action correspondante est déclenchée.

4. Action de filtrage : Selon les règles de filtrage, différentes actions peuvent être prises. Par exemple, le paquet peut être autorisé à passer, bloqué, redirigé vers une autre destination, marqué pour une inspection plus approfondie, ou encore soumis à une action spécifique, comme le blocage de l'adresse IP source.

5. Programmation des règles de traitement : Le contrôleur SDN programme les règles de traitement sur les appareils du plan de données pour appliquer les décisions prises. Ces règles sont appliquées localement par les commutateurs SDN, ce qui permet un filtrage et un blocage efficaces des paquets au niveau du réseau.

Le comportement d'un pare-feu SDN peut être à la fois actif et proactif, cela signifie :

- Comportement actif : Un pare-feu SDN agit de manière active en détectant et en réagissant aux événements en temps réel. Lorsqu'un paquet correspond à une règle de filtrage qui nécessite une action, le pare-feu prend immédiatement des mesures pour filtrer ou bloquer le paquet en fonction de la politique de sécurité définie. Cela permet une réponse rapide et efficace aux menaces ou aux tentatives d'intrusion.

- Comportement proactif : Un pare-feu SDN adopte une approche proactive en mettant en place des mesures de sécurité préventives pour réduire les risques potentiels.

Cela peut inclure l'utilisation de techniques de détection d'anomalies, d'apprentissage automatique ou de modèles comportementaux pour identifier les activités suspectes avant qu'elles ne deviennent des menaces réelles. Le pare-feu peut également mettre à jour automatiquement les règles de filtrage en fonction des nouvelles menaces connues pour renforcer la sécurité globale du réseau.

En combinant à la fois un comportement actif et proactif, un pare-feu SDN offre une approche plus dynamique et adaptable à la sécurité du réseau, en détectant et en bloquant les menaces en temps réel tout en prévenant activement les risques potentiels. Cela permet de renforcer la sécurité globale du réseau et de protéger les ressources et les données sensibles.

B. La surveillance et l'analyse du trafic :

Sont des fonctionnalités clés des pare-feu SDN, Voici comment ces fonctionnalités contribuent à la sécurité :

Surveillance du trafic : SD-FW surveille en continu le trafic réseau en analysant les flux de données qui traversent le réseau. Ils collectent des informations sur les adresses source et destination, les protocoles utilisés, les ports de communication, etc. Cette surveillance permet de détecter les comportements anormaux, les tentatives d'intrusion ou les activités suspectes.

Analyse du trafic : pare-feu SDN analyse le trafic en appliquant des règles de filtrage et des mécanismes de détection d'intrusion. Il examine chaque paquet de données pour identifier les menaces potentielles, les attaques de sécurité, les virus ou les logiciels malveillants. L'analyse du trafic permet de détecter et de bloquer les activités indésirables afin de prévenir les violations de sécurité.

Identification des schémas de trafic : Le SD-FW peut analyser les schémas de trafic et les comportements usuels pour détecter les anomalies. En utilisant des techniques de machine learning et de modélisation du trafic, il peut identifier les variations suspectes, les pics de trafic inhabituels ou les modèles de communication atypiques. Cela permet de repérer les attaques sophistiquées qui peuvent contourner les règles de filtrage traditionnelles.

Réponse proactive : Le pare-feu SDN prend des mesures proactives pour contrer les menaces détectées. Par exemple, il bloque instantanément le trafic suspect, envoyer des alertes aux administrateurs réseau, appliquer des politiques de sécurité supplémentaires ou déclencher des mécanismes de défense automatiques pour contenir une attaque en cours.

2.5 Défis et considérations du SD-FW :

La mise en œuvre d'un SDN Firewall comporte certains défis et considérations qu'il est important de prendre en compte. Dans cette section, nous explorerons les principaux défis auxquels sont confrontés les déploiements de SDN Firewall, ainsi que les considérations essentielles pour garantir son bon fonctionnement :

- Coordination avec d'autres éléments du réseau : Dans un environnement SDN, le SD-FW doit être capable de coopérer et de communiquer avec d'autres éléments du réseau, tels que les commutateurs SDN et les contrôleurs. Une coordination efficace est essentielle pour garantir une protection cohérente et une réponse appropriée aux menaces.

- Performances et latence : L'introduction d'un SD-FW peut avoir un impact sur les performances du réseau, notamment en termes de latence. Il est important de réaliser des tests et des évaluations approfondis pour s'assurer que le SD-FW n'introduit pas de goulot d'étranglement et n'affecte pas négativement les performances du réseau.

- Sécurité du contrôleur SDN : Étant donné que le SD-FW dépend du contrôleur SDN pour sa gestion et sa coordination, il est crucial de garantir la sécurité et la fiabilité du contrôleur. Des mesures de sécurité appropriées doivent être mises en place pour protéger le contrôleur contre les attaques potentielles.

- Choix d'un contrôleur performant : Étant donné que l'application Firewall doit réagir en temps réel pour analyser et filtrer le trafic réseau, le contrôleur joue un rôle crucial dans la prise de décision rapide et efficace. Lors de la sélection d'un contrôleur SDN, plusieurs facteurs doivent être pris en compte pour assurer des performances optimales, notamment : la capacité de traitement, la latence, la scalabilité et Les fonctionnalités de programmabilité.

2.6 Travaux Connexe :

Le paradigme du SDN a émergé comme une réponse aux limitations des réseaux traditionnels. Le SDN offre plusieurs avantages qui ont attiré l'attention des chercheurs pour le déploiement de pare-feu basé sur le concept de SDN (SD-FW).

Les travaux connexes réalisés par plusieurs chercheurs ont exploré les possibilités offertes par l'utilisation du SDN pour la mise en place de pare-feu, Voici le tableau résumant les travaux connexes du SD-FW :

Auteur	Caractéristiques du pare-feu	Avantages	Inconvénients
Tariq [52]	Pare-feu de couche 2 centralisé basé sur des adresses MAC	Restreint l'accès à des segments de réseau spécifiques	Ne prend pas en compte d'autres champs d'en-tête tels que TCP et UDP
Kaur [53]	Pare-feu centralisé avec un seul commutateur configuré en tant que pare-feu	Configuration simplifiée	Point de défaillance unique, surcharge de règles possible

Pena [54]	Pare-feu distribué basé sur le SDN avec focus sur le trafic ICMP	Inspection et filtrage du trafic ICMP	Ne prend pas en compte d'autres types de trafic comme TCP et UDP
Wang et al [55]	Pare-feu SDN basé sur l'apprentissage automatique pour détecter et bloquer les attaques réseau en temps réel	Utilise l'apprentissage automatique pour des décisions de filtrage intelligentes	N/A

Tableau 2 : Tableau récapitulatif des travaux connexes sur les pares-feux SDN

Ces travaux connexes mettent en évidence différentes approches et techniques dans le domaine des pare-feu basés sur le concept de SDN. Ils contribuent à l'évolution des pare-feu SDN en proposant des solutions centrées sur l'analyse des adresses MAC, la centralisation ou la distribution du pare-feu, la prise en compte spécifique du trafic ICMP ou l'utilisation de l'apprentissage automatique pour la détection d'attaques en temps réel.

Notre travail (exposé et détaillé dans le chapitre 4 de notre mémoire) s'inscrit dans le contexte des travaux connexes : firewall basé sur le concept de SDN, en intégrant les idées et les avancées réalisées par d'autres chercheurs dans le domaine. Nous avons développé et implémenté un pare-feu SDN distribué basé sur le protocole OpenFlow, conçu pour prendre en charge les protocoles TCP, UDP et ICMP, ce qui lui permet d'inspecter et de filtrer efficacement différents types de trafic. L'un des aspects clés de notre travail est la capacité du pare-feu à surveiller et à analyser le réseau en temps réel.

2.7 Conclusion :

Ce chapitre a mis en évidence l'importance et la pertinence du pare-feu SDN en tant que solution de sécurité programmable dans les réseaux SDN. Nous avons exploré en détail les fondements, les avantages et les fonctionnalités clés du pare-feu SDN, mettant en évidence sa capacité à combiner la programmabilité du SDN avec les fonctionnalités de filtrage et de contrôle du trafic des pare-feu traditionnels.

À la lumière de ces discussions, nous pouvons conclure que le SD-FW offre une solution prometteuse pour garantir la sécurité des réseaux SDN, en combinant flexibilité, programmabilité et capacités de contrôle avancées. Cela nous motive à poursuivre nos recherches et à approfondir notre compréhension de cette technologie.

Le chapitre suivant se concentrera sur une étude comparative des performances des deux contrôleurs SDN populaires, Ryu et OpenDaylight, dans le contexte de la mise en œuvre de notre application firewall. Cette étude nous permettra de sélectionner le contrôleur le plus adapté à nos besoins et de passer à la phase de mise en œuvre de notre pare-feu SDN distribué.

Chapitre 3

Performance des contrôleurs SDN : Analyse comparative entre RYU et ODL

3.1 Introduction :

La technologie SDN a révolutionné les outils de gestions et d'exploitations des réseaux, grâce au contrôleur SDN qui est le cœur de cette innovation, il fournit une interface de gestion et de programmation centralisée pour l'ensemble de la plateforme réseau (les actifs et les services associés).

Cependant, la performance des contrôleurs SDN est un aspect crucial et essentielle pour garantir un fonctionnement efficace du réseau et offrir des services de qualités aux utilisateurs.

Dans ce chapitre, nous allons examiner la performance de deux des contrôleurs SDN les plus populaires, RYU et ODL, en termes de latence et de débit.

Nous discuterons des tests et mesures de performance effectuées pour chacun des contrôleurs, nous analyserons également les résultats des tests et fournirons une comparaison objective entre les deux contrôleurs RYU et ODL en termes de performance.

3.2 Contrôleur SDN :

3.2.1 RYU :

RYU est un Framework de contrôleur SDN open source développée en Python, qui permet aux utilisateurs de programmer des applications réseau personnalisées à l'aide d'interfaces de programmation d'application (API) simples.

Le contrôleur RYU est basée sur un modèle d'exécution de programme événementiel, où chaque événement est traité par un module séparé en utilisant des tâches asynchrones. Le noyau RYU est conçu pour être léger et extensible, ce qui permet aux développeurs de personnaliser le comportement du contrôleur en utilisant des modules de traitement d'événements supplémentaires [33].

L'architecture du contrôleur RYU peut être divisée en trois couches principales : la couche de transport, la couche de traitement d'événements et la couche d'interface utilisateur.

La couche de transport est responsable de la communication entre le contrôleur RYU et les commutateurs du réseau. Cette couche utilise des protocoles standard tels qu'OpenFlow pour communiquer avec les commutateurs.

La couche de traitement d'événements du réseau, tels que les paquets entrants et les changements d'état de la topologie du réseau. Cette couche est constituée de plusieurs modules, chacun étant responsable d'un type spécifique de traitement d'événements.

Enfin, la couche d'interface utilisateur fournit une interface de programmation pour les développeurs d'applications réseau, ce qui leur permet de créer des applications personnalisées qui peuvent communiquer avec le contrôleur RYU.

La figure 16 suivante illustre les composants du contrôleur RYU qui facilitent le développement d'applications réseau et la gestion du réseau.

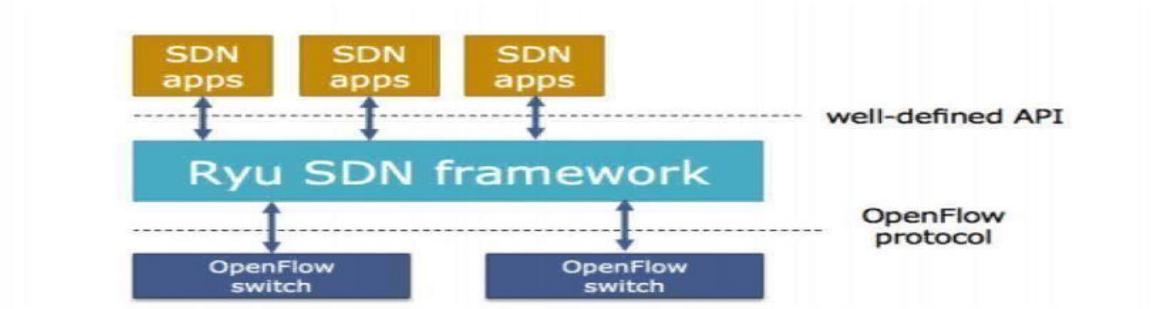

Figure 16 : L'architecture du contrôleur RYU [34].

3.2.2 OpenDaylight :

OpenDaylight est un Framework de contrôleur SDN open source conçu pour faciliter la migration vers une architecture de réseau SDN. Le contrôleur ODL est développé en Java et fournit une plate-forme modulaire permettant de personnaliser et d'automatiser des réseaux de toutes tailles et de toutes échelles.

L'architecture OpenDaylight se compose de plusieurs composants de base, notamment le noyau du contrôleur, les interfaces de programmation d'applications (API), les services de gestion du réseau, les bibliothèques de fonctions et les outils de développement. Ces composants sont organisés de manière modulaire, ce qui permet aux utilisateurs de personnaliser le contrôleur en fonction de leurs besoins spécifiques [35].

La figure 17 montre la structure et les composants du contrôleur OpenDaylight .

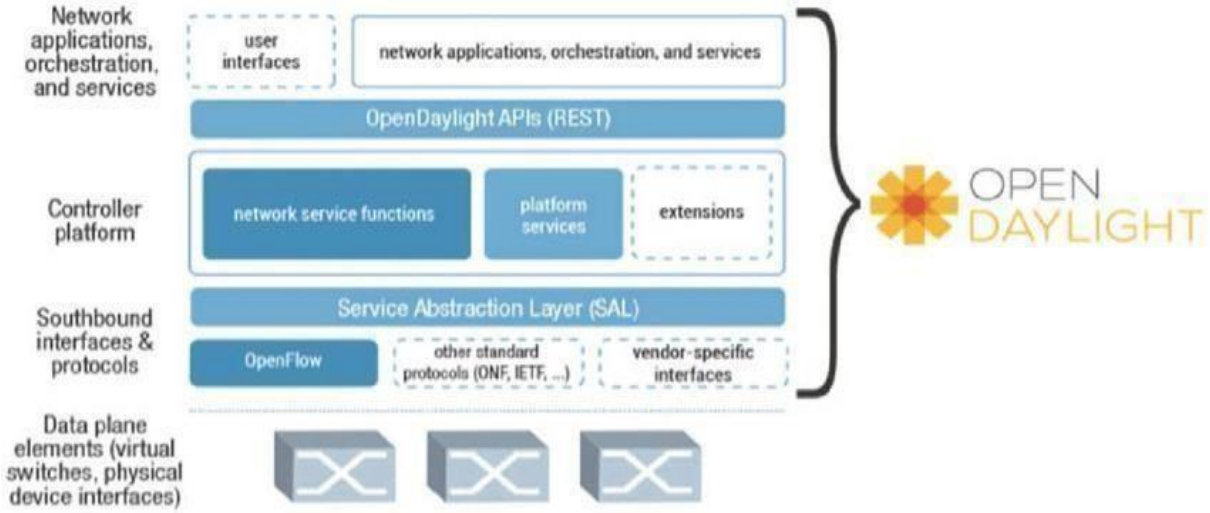

Figure 17 : L'architecture du contrôleur ODL [35].

3.3 Emulateur SDN :

Un émulateur SDN est une plateforme logicielle qui permet de reproduire un environnement SDN. Il émule les fonctionnalités des commutateurs, des routeurs et du contrôleur SDN pour permettre aux utilisateurs de créer des topologies de réseau virtuelles, configurer les commutateurs virtuels, gérer les flux de données, et tester des applications et des protocoles SDN dans un environnement contrôlé. Cela offre aux développeurs et aux ingénieurs réseau la possibilité de simuler et de valider des configurations, des politiques de gestion, des algorithmes de routage, et de mesurer les performances du réseau avant de le déployer dans un environnement réel. L'émulateur SDN facilite ainsi le développement, le test et la validation des solutions SDN, tout en réduisant les risques et en assurant une meilleure stabilité du réseau.

3.3.1 MININET :

Mininet est un émulateur réseau open source basé sur Linux, utilise la virtualisation pour créer des machines et des commutateurs virtuels (voir figure 18), qui peuvent être connectés via des liens de réseau virtuels pour simuler un réseau physique. Il prend également en charge plusieurs contrôleurs SDN, tels que OpenDaylight et RYU, qui peuvent être utilisés pour gérer le réseau virtuel créé. Les utilisateurs peuvent également personnaliser la topologie du réseau virtuel créé en utilisant des scripts Python pour définir les paramètres de configuration et les comportements des différents composants du réseau [36]. Il permet également aux utilisateurs de programmer le comportement de ces commutateurs à l'aide de protocoles tels que OpenFlow.

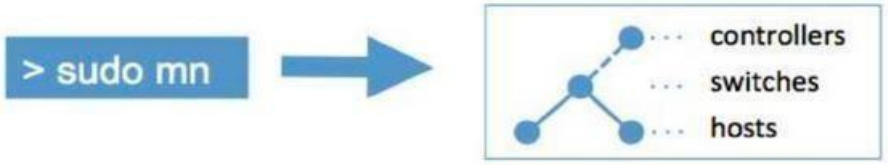

Figure 18 : Mininet [36].

Mininet utilise le commutateur OpenvSwitch qui supporte le protocole OpenFlow pour la programmation du réseau, permettant ainsi de simuler et de tester des topologies réseaux complexes.

3.4 OpenvSwitch :

Dans le cadre de notre émulation, OpenVswitch occupe une place centrale en tant que commutateur virtuel largement utilisé dans le domaine du SDN, En tant que logiciel open source sous licence Apache 2.0, OVS offre une solution flexible et performante pour interconnecter les machines virtuelles et créer des réseaux virtuels [37].

Pour fonctionner comme un switch traditionnel, OpenVswitch utilise la notion de ports. Chaque port est associé à une ou plusieurs interfaces, qui correspondent aux interfaces logiques ou physiques du système hôte.

OpenvSwitch offre un large éventail de fonctionnalités, notamment le support des tags VLAN (norme 802.1Q), le protocole Spanning Tree, la gestion de la qualité de service et bien d'autres [38].

De plus, OpenvSwitch prend également en charge le protocole OpenFlow, qui permet de programmer le comportement du switch via des règles et des actions définies par l'application SDN.
Voici quelques commandes fréquemment utilisées pour interagir avec OpenvSwitch : [39]

Afficher la configuration actuelle d'OVS :

```
ovs-vsctl show
```

Ajouter une interface au bridge OVS :

```
ovs-vsctl add-port <nom-du-bridge> <nom-de-l-interface>
```

Afficher les ports d'un bridge OVS :

```
ovs-vsctl list-ports <nom-du-bridge>
```

Configurer une règle OpenFlow sur un bridge OVS :

```
ovs-ofctl add-flow <nom-du-bridge> <règle-OpenFlow>
```

Redémarrer le service OVS :

```
sudo systemctl restart openvswitch
```

3.5 Évaluation des performances RYU et OpenDaylight :

L'évaluation des performances des solutions de contrôle de réseau est essentielle pour garantir des réseaux efficaces et réactifs. Dans cet esprit, nous nous intéressons à deux technologies de contrôle de réseau, RYU et OpenDaylight.

L'objectif de cette évaluation est d'analyser les performances de RYU et ODL en termes de débit et de latence, deux métriques clés pour mesurer l'efficacité et la réactivité d'un système de contrôle de réseau. Le débit fait référence à la quantité de données qu'un système peut traiter en un temps donner, tandis que la latence correspond au temps nécessaire pour qu'un système réponde à une demande spécifique. Nous avons utilisé une approche pratique courante en utilisant un émulateur pour simuler des environnements réseau réalistes.

L'émulateur offre un environnement virtuel où les performances des systèmes peuvent être évaluées de manière contrôlée et reproductible.

Il permet de simuler divers scénarios réseau, en manipulant des paramètres tels que la topologie du réseau, le trafic et les charges de travail. Ainsi, il est possible d'observer et de mesurer les performances des systèmes dans des conditions spécifiques [40].

En évaluant le débit, nous cherchons à déterminer la capacité de chaque système à traiter les flux de données avec efficacité, en mesurant la quantité de données qu'ils peuvent gérer par unité de temps. Cette évaluation nous permettra de comprendre si RYU et ODL peuvent gérer des charges de travail réseau importantes et maintenir des performances constantes.

La latence, quant à elle, est un aspect important à prendre en compte pour mesurer la réactivité d'un système de contrôle de réseau. Nous mesurerons le temps écoulé entre l'envoi d'une requête à chaque système et la réception de la réponse correspondante, afin de déterminer leur capacité à répondre rapidement aux demandes réseau en temps réel.

En comparant les performances de RYU et ODL en termes de débit et de latence, nous pourrons évaluer leurs capacités respectives dans des situations réelles. Cette évaluation basée sur l'émulateur nous permettra de mieux comprendre les avantages et les limites de chaque solution.

3.5.1 CBench émulation :

L'évaluation des performances des contrôleurs dans les réseaux définis par logiciel (SDN) est cruciale pour garantir des réseaux efficaces et réactifs. Cbench, un outil largement utilisé, offre une approche pratique pour évaluer les performances des contrôleurs SDN [43].

Cbench propose deux principaux modes de travail pour évaluer les performances des contrôleurs : le mode débit et le mode latence [44]. Dans le mode débit, Cbench envoie un flux important de paquets au contrôleur afin de déterminer le nombre maximal de paquets que celui-ci peut traiter. Cette évaluation permet de mesurer la capacité du contrôleur à gérer efficacement les charges de travail réseau élevées et à maintenir des performances stables.

D'autre part, dans le mode latence, Cbench envoie un seul paquet au contrôleur et mesure le temps nécessaire pour que le contrôleur réponde à cette demande en calculant la latence. Ce mode permet de quantifier la réactivité du contrôleur, C'est-à-dire combien de temps il faut au contrôleur pour traiter un seul paquet.

Une latence minimale est essentielle pour garantir des performances réseau optimales, en particulier dans des environnements où la rapidité de réaction est cruciale.

L'utilisation de Cbench pour évaluer les performances des contrôleurs SDN offre plusieurs avantages. Tout d'abord, Cbench est un outil bien établi et largement utilisé dans la communauté SDN, ce qui facilite la comparabilité des résultats entre différentes évaluations. De plus, les modes de débit et de latence de Cbench permettent d'évaluer les performances des contrôleurs dans des scénarios réalistes, en tenant compte à la fois de la capacité de traitement et de la réactivité [42].

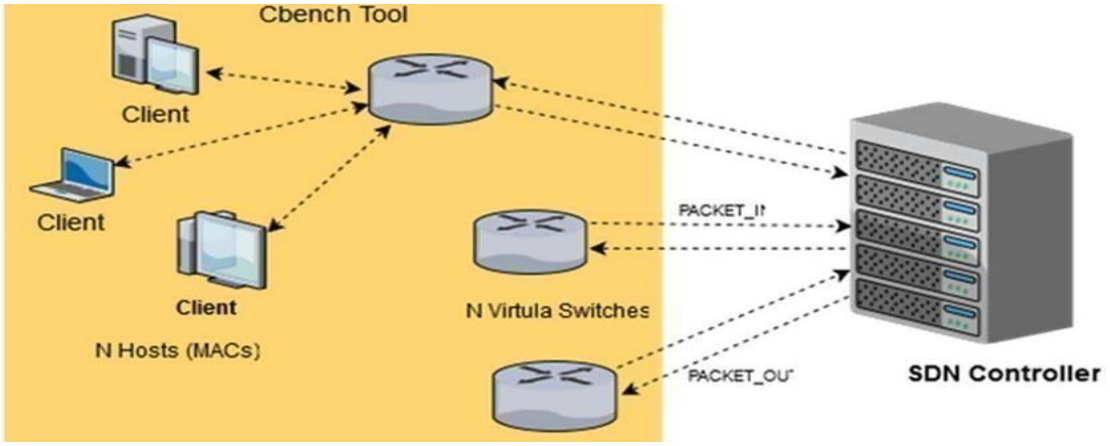

Figure 19 : Scénario de test [41].

Dans le scénario de test illustré par la figure x, on observe des échanges de messages "Packet-In" et "Packet-Out" entre un commutateur virtuel et le contrôleur SDN.

Ces échanges de messages sont pertinents pour évaluer la performance des contrôleurs Ryu et OpenDaylight. Nous mesurerons à la fois le débit et la latence pour obtenir une vision globale des performances de ces contrôleurs dans des conditions réelles.

3.5.2 Configuration du banc d'essai :

Les tests ont été effectués sur une machine dotée d'un processus Intel Core i5, 8Go de RAM fonctionnant sous Windows, à l'aide d'une machine virtuelle VirtualBox exécutante Ubuntu version 18.04 en 64 Bits.

Dans l'environnement d'émulation, chaque domaine du réseau utilise deux machines virtuelles : une pour le contrôleur SDN et une autre pour le réseau. La machine virtuelle du réseau local, utilisant Mininet, est connectée aux contrôleurs, d'abord avec ODL, puis avec RYU.

Le protocole de communication entre les commutateurs virtuels (OVS) et les contrôleurs est OpenFlow 1.3.

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

- OpenDayLight : Carbon version 0.6.4

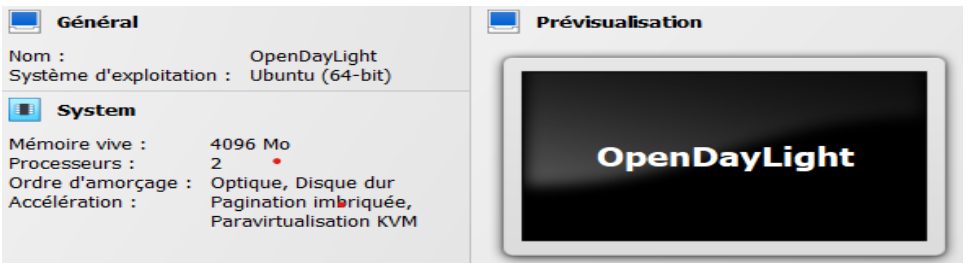

Figure 20 : Configuration de la VM du contrôleur ODL.

- RYU : version 4.15

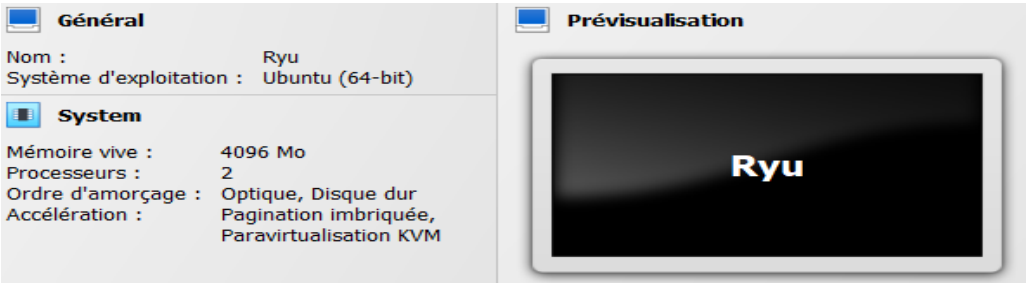

Figure 21 : Configuration de la VM du contrôleur RYU.

3.5.3 Tests de débit :

Nous avons évalué les performances de débit du contrôleur en utilisant la commande suivante :

```
hoyem@hoyem-VirtualBox:~$ cbench -c 192.168.1.33 -p 6633 -m 10000 -l 3 -s 16 -M
 100 -t
```

Voici la signification des différents paramètres :

-c : spécifie l'adresse IP du contrôleur auquel nous souhaitons nous connecter

-p : spécifie le numéro de port sur lequel le contrôleur SDN écoute les connexions. (Le port 6633 est couramment utilisé comme port par défaut pour les contrôleurs SDN qui utilisent le protocole OpenFlow)

-s : indique le nombre de commutateurs dans le réseau SDN.

-l : spécifie le nombre de tests à effectuer pour mesurer le débit. Chaque test envoie un certain nombre de paquets pour évaluer les performances.

-t : active le mode "throughput" (débit) pour mesurer les performances de débit du réseau SDN.

RYU :

Pour que CBench puisse se connecter à RYU, le contrôleur doit être en cours d'exécution et être accessible à l'adresse IP et au port spécifiés lors de l'exécution de la commande CBench.

```
cbench: controller benchmarking tool
   running in mode 'throughput'
   connecting to controller at 192.168.1.33:6633
```

La sortie ci-dessus indique que CBench est en mode "throughput" (débit) et connecté au contrôleur à l'adresse IP 192.168.1.33 sur le port 6633.

```
packet in 4 00:5e:00:00:00:04 80:00:00:00:00:04 1
packet in 2 00:5e:00:00:00:02 80:00:00:00:00:02 1
packet in 7 00:5e:00:00:00:07 80:00:00:00:00:07 1
packet in 9 00:5e:00:00:00:09 80:00:00:00:00:09 1
```

Ces lignes de journal correspondent aux paquets reçus par les commutateurs lors de l'exécution de CBench. Chaque ligne indique le numéro du commutateur, l'adresse MAC de destination du paquet, l'adresse MAC source du paquet et le nombre de paquets reçus.

Les performances mesurées lors de l'exécution des tests de benchmarking avec :

- 8 Commutateurs : 5361.68 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
RESULT: 8 switches 2 tests min/max/avg/stdev = 5338.62/5384.75/5361.68/23.06 re
sponses/s
```

- 12 Commutateurs : 5400.88 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
RESULT: 12 switches 2 tests min/max/avg/stdev = 5380.53/5421.23/5400.88/20.35 r
esponses/s
```

- 16 Commutateurs : 5431.13 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
RESULT: 16 switches 2 tests min/max/avg/stdev = 5425.29/5436.96/5431.13/5.83 re
sponses/s
```

Analyse des résultats obtenus pour les tests de performance de débit :

A partir des résultats obtenus, il est constaté une légère augmentation du débit moyen à mesure que le nombre de switch augmente, Cela indique que le contrôleur RYU peut gérer un nombre croissant de commutateurs en termes de traitement des requêtes et de réponse par seconde.

OpenDaylight :

Lors de l'exécution de la commande CBench avec différents nombres de switches 8, 12 et 16, l'interface ODL les reconnaît tous comme le montre la figure 22, cela signifie que le contrôleur OpenDaylight intégré avec l'outil CBench, a établi avec succès une connexion avec chaque switch du réseau correspondant.

Cette reconnaissance indique que le contrôleur ODL, via CBench, est capable d'identifier et de communiquer avec chaque commutateur.

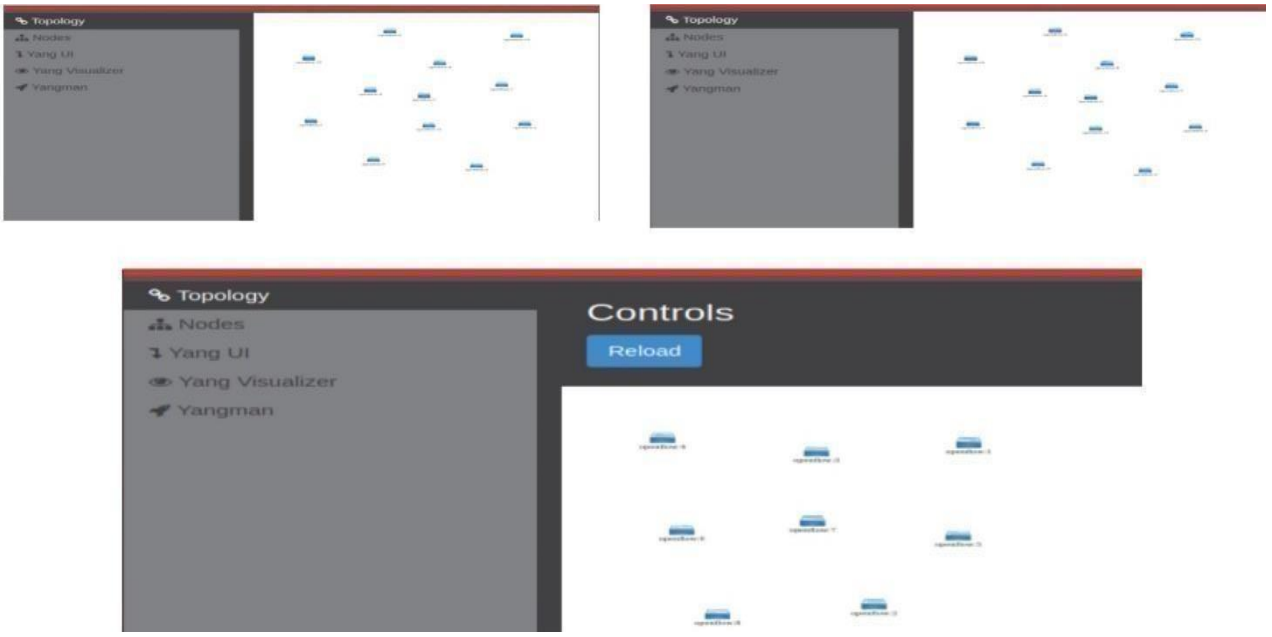

Figure 22 : Reconnaissance des switches.

Les performances mesurées lors de l'exécution des tests de benchmarking avec :

- 8 Commutateurs :

```
00:21:11.352 8   switches: flows/sec:  0  0  0  0  0  0  0  0   total = 0.00000
0 per ms
00:21:21.456 8   switches: flows/sec:  0  0  0  0  0  0  0  0   total = 0.00000
0 per ms
RESULT: 8 switches 9 tests min/max/avg/stdev = 0.00/0.00/0.00/0.00 responses/s
```

- 12 Commutateurs :

```
00:24:44.878 12  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0   tot
al = 0.000000 per ms
00:24:54.980 12  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0   tot
al = 0.000000 per ms
RESULT: 12 switches 9 tests min/max/avg/stdev = 0.00/0.00/0.00/0.00 responses/s
```

- 16 Commutateurs :

```
00:33:37.506 16  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0   total = 0.000000 per ms
RESULT: 16 switches 9 tests min/max/avg/stdev = 0.00/0.00/0.00/0.00 responses/s
```

Les résultats obtenus lors du test de débit avec le contrôleur ODL ne sont pas significatifs, ce qui laisse supposer que le contrôleur peut ne pas être en mesure de traiter efficacement un grand nombre de paquets simultanément. Cette situation est conditionnée par deux paramètres :

- Limitations de capacité de traitement du contrôleur.
- Configuration non appropriée pour le traitement des paquets.

3.5.4 Tests de latence :

Pour mesurer la latence du contrôleur, nous avons utilisé la ligne de commande suivante :

```
hoyem@hoyem-VirtualBox:~$ cbench -c 192.168.1.33 -p 6633 -m 10000 -l 3 -s 8 -M
100
```

RYU :

La sortie de la commande CBench après l'exécution du benchmark de latence :

```
cbench: controller benchmarking tool
   running in mode 'latency'
   connecting to controller at 192.168.1.33:6633
```

Les résultats obtenus lors de l'exécution des tests de benchmarking en mode latency :

- 8 Commutateurs : 3995.60 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
RESULT: 8 switches 2 tests min/max/avg/stdev = 3509.20/4482.00/3995.60/486.40 r
esponses/s
```

- 12 Commutateurs : 3626.80 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
RESULT: 12 switches 2 tests min/max/avg/stdev = 3598.60/3655.00/3626.80/28.20 r
esponses/s
```

- 16 Commutateurs : 3730.70 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
RESULT: 16 switches 2 tests min/max/avg/stdev = 3725.40/3736.00/3730.70/5.30 re
sponses/s
```

Les résultats du test montrent que la latence est relativement stable, la moyenne de latence a diminué entre 8 et 12 commutateurs, et n'a pas augmenté de manière significative entre 12 et 16 commutateurs, cela signifie que le système est capable de traiter efficacement les requêtes malgré l'augmentation de la charge.

OpenDaylight :

Voici les résultats obtenus lors du test de performance en mode latence pour le contrôleur ODL :

- 8 Commutateurs : 0.40 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
23:03:02.383 8   switches: flows/sec:  2  2  2  2  2  0  2  0   total = 0.00120
0 per ms
23:03:12.486 8   switches: flows/sec:  4  4  4  4  4  6  4  6   total = 0.00359
9 per ms
23:03:22.589 8   switches: flows/sec:  0  0  0  0  0  0  0  0   total = 0.00000
0 per ms
23:03:32.689 8   switches: flows/sec:  0  0  0  0  0  0  0  0   total = 0.00000
```

```
RESULT: 8 switches 9 tests min/max/avg/stdev = 0.00/3.60/0.40/1.13 responses/s
```

- 12 Commutateurs : 0.53 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
23:07:28.035 12  switches: flows/sec:  2  2  2  2  2  2  2  2  2  2  2  2   tot
al = 0.002400 per ms
23:07:38.138 12  switches: flows/sec:  4  4  4  4  4  4  4  4  4  4  4  4   tot
al = 0.004799 per ms
23:07:48.238 12  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0   tot
al = 0.000000 per ms
```

```
RESULT: 12 switches 9 tests min/max/avg/stdev = 0.00/4.80/0.53/1.51 responses/s
```

- 16 Commutateurs : 4.05 responses/s : la moyenne du nombre de réponses par seconde obtenue lors des tests.

```
22:42:31.774 16  switches: flows/sec:  2  2  2  2  2  2  2  2  2  0  0  2  2  0
  0  0   total = 0.021919 per ms
22:42:32.875 16  switches: flows/sec:  0  0  0  0  0  0  0  0  0  2  2  0  0  2
  2  2   total = 0.010000 per ms
22:42:33.975 16  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0   total = 0.000000 per ms
22:42:35.075 16  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0   total = 0.000000 per ms
22:42:36.179 16  switches: flows/sec:  0  0  0  0  0  0  1  4  4  4  4  4  4  4
  0  0   total = 0.028901 per ms
22:42:37.280 16  switches: flows/sec:  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
RESULT: 16 switches 15 tests min/max/avg/stdev = 0.00/28.90/4.05/8.83 responses
/s
```

Les résultats du test montrent que le débit de traitement des flux par les commutateurs simulés est assez faible, avec une moyenne de 0,53/4.05/0.40 réponses par seconde. Cela peut être considéré comme relativement lent.

Cette performance insuffisante peut être préjudiciable aux applications qui nécessitent des temps de réponse rapides.

Ces retards peuvent entraîner des délais indésirables dans la transmission des données et affecter la réactivité globale du réseau.

3.5.5 Discussion :

Comme indiqué dans la représentation graphique de la figure 22, les résultats du test de latence obtenu pour RYU étaient constants et inférieurs à ceux du test de débit. Ces résultats suggèrent que RYU offre de bonnes performances en termes de latence, car ils sont stables et relativement bas. Une faible latence est essentielle pour les applications réseau réactives.La consistance des résultats renforce l'idée que RYU est capable de maintenir une latence réduite de manière fiable.

En ce qui concerne le test de débit, les résultats pour RYU ont augmenté avec le nombre de commutateurs, ce qui indique une évolutivité positive. Cela signifie que RYU est capable de gérer efficacement un trafic réseau croissant.

Concernant les résultats du test de débit pour ODL (Voir figure 24), ils étaient non significatifs, affichant un résultat de 0 rps (requêtes par seconde). Ces résultats indiquent que ODL a éprouvé des difficultés à gérer le trafic à des débits élevés. Une absence totale de réponses suggère un problème majeur de performance.

Quant à le test de latence pour ODL, les résultats étaient très faibles, avec une moyenne de 0,5 rps. Cela suggère que les temps de réponse d'ODL étaient significativement lents. Des temps de réponse élevés peuvent indiquer des problèmes de performance.

Les résultats des tests suggèrent que RYU affiche de meilleures performances que ODL, RYU a montré des performances en termes de latence constantes et inférieures, ce qui est bénéfique pour les applications réactives en temps réel. De plus, RYU a démontré une évolutivité positive en termes de débit, avec des résultats qui augmentent avec le nombre de switch.

En revanche, ODL a rencontré des difficultés à gérer le trafic à des débits élevés, avec des résultats nuls et des temps de réponse lents.

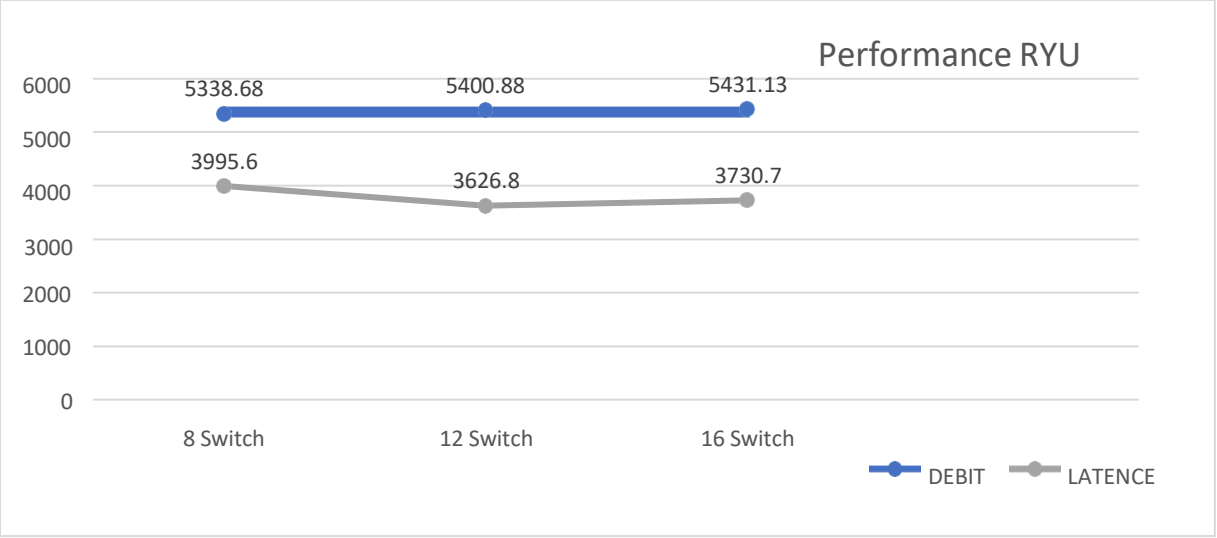

Figure 23 : Performance RYU.

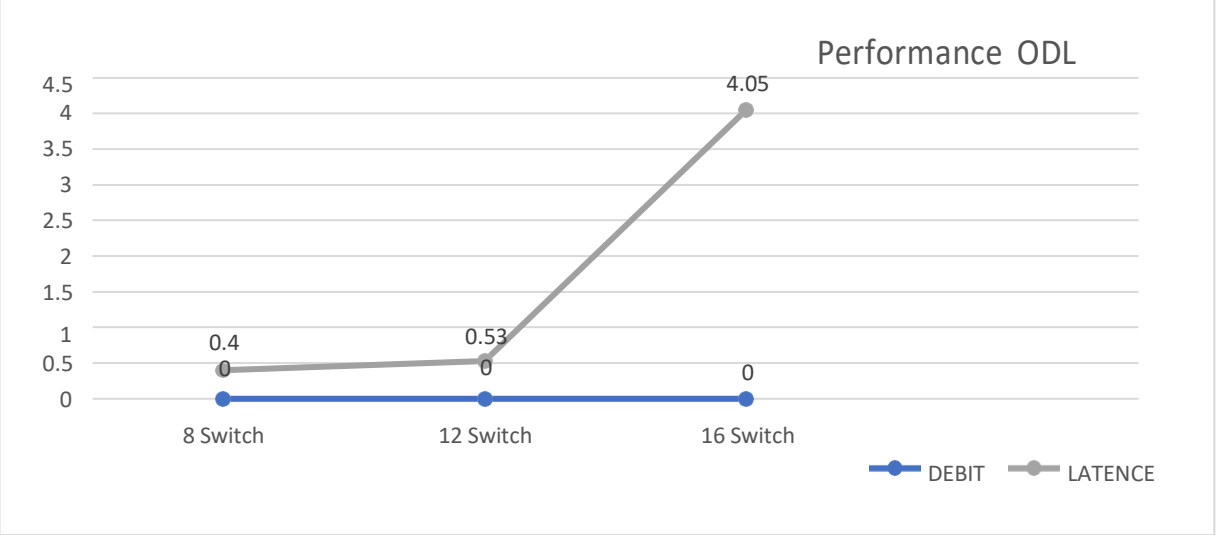

Figure 24 : Performance ODL.

3.6 Conclusion :

Dans ce chapitre, nous avons examiné les outils essentiels pour le déploiement d'un réseau SDN, tels que le contrôleur, l'émulateur Mininet et l'OpenVswitch. De plus, nous avons comparé les performances des contrôleurs OpenDaylight et RYU en utilisant l'outil CBench pour évaluer leur débit et leur latence.

Les résultats des tests de performances suggèrent que RYU se distingue comme un choix prometteur pour l'implémentation de l'application Firewall dans le prochain chapitre. Les performances stables en termes de latence et l'évolutivité positive en termes de débit observées avec RYU sont des avantages majeurs pour une gestion et un contrôle efficace du réseau. Ces résultats démontrent la réactivité et l'efficacité de RYU dans le traitement du trafic réseau.

En utilisant RYU comme contrôleur SDN pour notre application Firewall, nous pourrons tirer parti de ses performances optimales pour analyser et filtrer le trafic réseau de manière efficace et en temps réel. Cela contribuera à renforcer la sécurité du réseau SDN en offrant une réponse rapide aux menaces et en appliquant les politiques de sécurité définies.

Dans le prochain chapitre, nous nous concentrerons en détail sur l'application Firewall dans un environnement SDN, en explorant ses fonctionnalités, son architecture et les mécanismes de sécurité associés.

Chapitre 4

Implémentation d'un FIREWALL Distribué basé sur OpenFlow: Architecture, Intégration et Performance.

4.1 Introduction :

Ce chapitre se concentre sur l'implémentation d'un pare-feu distribué basé sur le protocole OpenFlow version 1.3, en mettant l'accent sur la conception de l'architecture réseau SDN spécifique à l'organisme d'accueil, l'intégration de l'application Firewall dans le contrôleur RYU, ainsi que l'analyse des performances.

Nous commencerons par présenter la conception de l'architecture réseau SDN adaptée à l'organisme d'accueil. Nous détaillerons les différents éléments de cette architecture et les considérations pratiques nécessaires pour assurer une infrastructure réseau adéquate à l'implémentation de la solution. Ensuite, nous aborderons la logique et le processus de fonctionnement de l'application intégré dans le contrôleur.

Dans la partie expérimentale, nous présenterons les résultats des tests effectués et nous analyserons ces résultats. Nous aborderons notamment les performances en termes de latence.

4.2 Conception de l'architecture SD-FW de l'organisme d'accueil :

Dans le cadre de l'implémentation d'un pare-feu distribué basé sur OpenFlow, notre approche s'est inspirée de l'architecture traditionnelle de l'organisme d'accueil (DP Sonatrach) présentée dans la figure 25. Le passage à une architecture SDN a permis la mise en place d'une solution SD-FW.

En reprenant les fondements de la topologie existante, nous avons conçu une architecture SDN minimisée qui a permis de simplifier et de rationaliser la structure globale du réseau. La partie système et serveurs a été connectée à la partie networking, ce qui nous a permis de minimiser l'architecture en regroupant les utilisateurs, les services et les serveurs sur la même maquette proposée. (Voir figures 26)

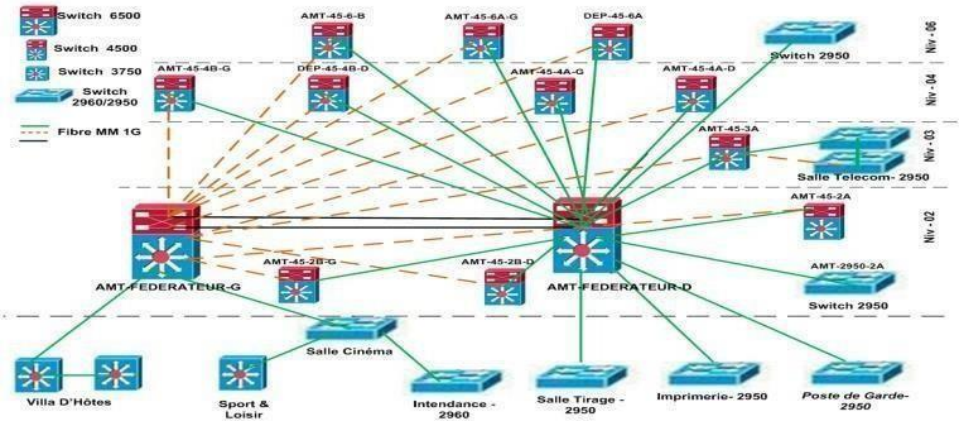

Figure 25 : architecture du siège DP. [Groupe Sonatrach]

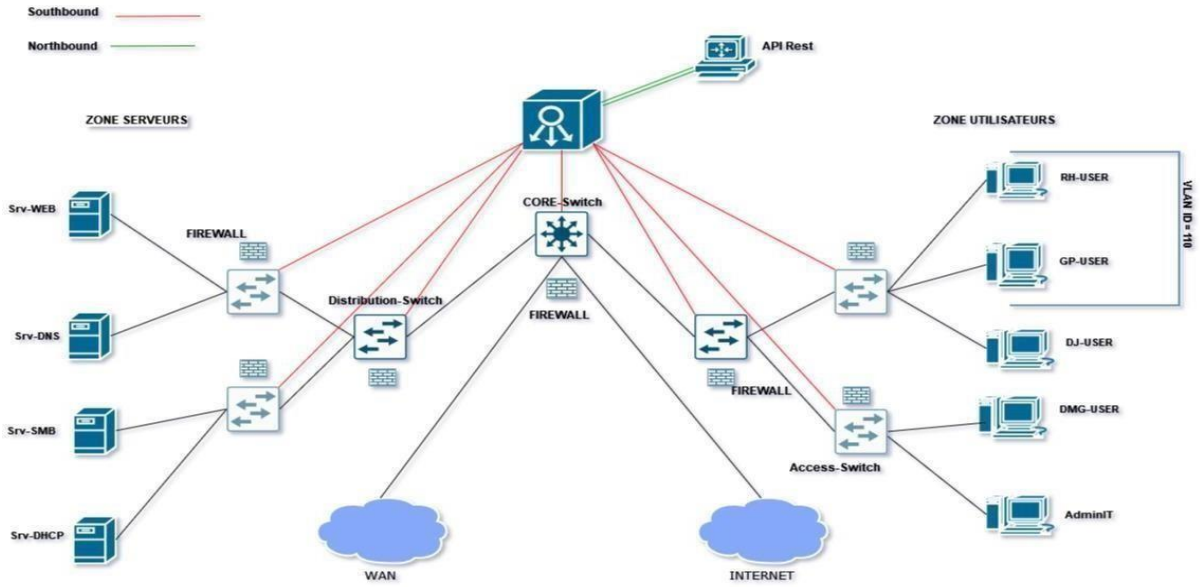

Figure 26 : architecture SD-FW minimisée du siège DP

Pour la conception de notre topologie SD-FW, nous avons choisi une architecture three-tier. Ce choix est justifié par plusieurs raisons. Premièrement, cette architecture permet une séparation claire des différentes fonctionnalités et responsabilités au sein du réseau. La couche Core est responsable de l'acheminement et du traitement efficace du trafic, la couche Distribution assure la connectivité entre les différents sous-réseaux, et la couche Access gère l'accès des utilisateurs au réseau. En déployant notre solution de pare-feu SDN à chaque niveau de l'architecture, nous bénéficions d'une approche de sécurité multicouche.

4.2.1 Zone Serveurs :

La zone serveurs est une partie spécifique du réseau. Dans ce contexte, nous avons mentionné quatre types de serveurs distincts : WEB, DNS, DHCP et SMB, utilisent principalement les protocoles TCP et UDP pour communiquer et échanger des données sur le réseau. Le protocole TCP assure une transmission fiable des données, tandis que le protocole UDP est souvent utilisé pour des communications en temps réel ou pour des transmissions de données plus légères et moins critiques.

4.2.2 Zone Utilisateurs :

Du côté des utilisateurs, plusieurs directions sont présentes, notamment RH, DMG, DP et DJ et ADMINEIT, Cependant, seul la machine admineIT dispose d'un accès privilégié et est responsable de la gestion à distance de l'ensemble des serveurs et équipements de l'infrastructure.

4.2.3 Vlan :

Dans la conception proposée, un VLAN avec l'ID 110 a été créé pour regrouper les départements RH et DP. Ce VLAN permet d'isoler le trafic réseau de ces derniers, offrant ainsi une segmentation logique et limitation de diffusions des données sensibles et confidentielles des départements en question. De plus, cela facilite la gestion des politiques de sécurité, des autorisations d'accès et des paramètres spécifiques à ces départements.

4.2.4 API Rest :

Dans les réseaux définis par logiciel, les API REST sont souvent utilisées pour permettre la communication et l'interaction entre les différentes entités du réseau.

Les API REST fournissent une architecture simple et standard pour l'échange des données entre le contrôleur SDN et les autres éléments du réseau, tels que les commutateurs, les routeurs et les applications. Elles utilisent les méthodes HTTP (comme GET, POST, PUT et DELETE) pour effectuer des opérations sur les ressources exposées par l'API.

Lorsqu'une requête est effectuée vers le contrôleur SDN via une API REST, celui-ci peut traiter la demande et renvoyer les résultats au format JSON. Le JSON est particulièrement adapté pour représenter les données structurées et est facilement interprétable par les applications clientes.

L'utilisation d'API REST dans un environnement SDN présente plusieurs avantages, permet une programmabilité et une automatisation accrue du réseau. Les développeurs peuvent utiliser les API REST pour créer des applications qui interagissent avec le contrôleur SDN, ce qui facilite la configuration, la gestion et la surveillance du réseau. [45]

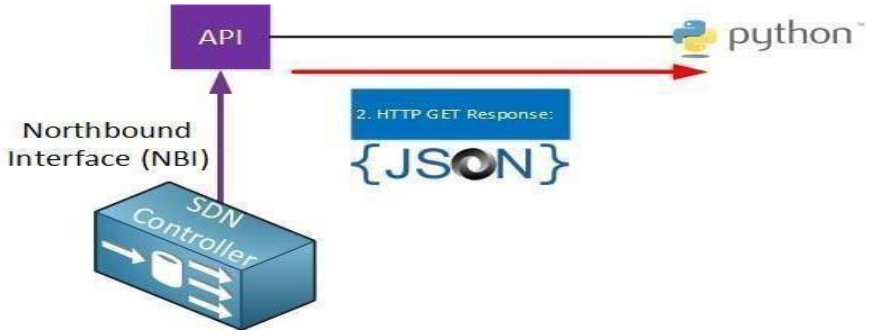

Figure 27 : Scénario APIRest [45].

Mininet présente l'avantage de permettre la personnalisation et la création des topologies selon les choix de l'utilisateur. Nous avons mis en œuvre la maquette proposée en utilisant un script écrit en langage Python : (Voir Figure 28 et 29)

```
  GNU nano 2.9.3                    NetworkTopo.py

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')
  h2 = self.addHost('h2')
  h3 = self.addHost('h3')
  h4 = self.addHost('h4')
  h5 = self.addHost('h5')
  h6 = self.addHost('h6')
  h7 = self.addHost('h7')
  h8 = self.addHost('h8')
  h9 = self.addHost('h9')
# Coeur
  C1 = self.addSwitch('C1', dpid='0000000000000001')
# Distribution
  D1 = self.addSwitch('D1', dpid='0000000000000002')
```

Figure 28 : Code Source de la topologie SDN partie 1.

```
D2 = self.addSwitch('D2', dpid='0000000000000003')
# Acccess
A1 = self.addSwitch('A1', dpid='0000000000000004')
A2 = self.addSwitch('A2', dpid='0000000000000005')
A3 = self.addSwitch('A3', dpid='0000000000000006')
A4 = self.addSwitch('A4', dpid='0000000000000007')
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(h6, A3)
self.addLink(h7, A3)
self.addLink(h8, A4)
self.addLink(h9, A4)
self.addLink(C1, D1)
self.addLink(C1, D2)
self.addLink(D1, A1)
self.addLink(D1, A2)
self.addLink(D2, A3)
self.addLink(D2, A4)
def runNetworkTopo() :
topo = NetworkTopo()
net = Mininet(topo=topo)
topo = NetworkTopo()
net = Mininet(topo=topo)
net.start()
CLI( net )
net.stop()
if __name__ == '__main__' :
 setLogLevel('info')
 runNetworkTopo()
topos={ 'networktopo': ( lambda: NetworkTopo() ) }
```

Figure 29 : Code Source de la topologie SDN partie 2.

Paramètre dpid est utilisé pour définir l'ID du périphérique d'un commutateur dans le contrôleur SDN, il est représenté sous forme de nombre hexadécimal.

4.3 Application Firewall Distribué basé OpenFlow :

Le framework Ryu est une plateforme de développement en Python spécifiquement conçue pour répondre aux besoins des environnements de réseaux définis par logiciel. Il fournit un ensemble complet d'outils, de bibliothèques et d'API qui simplifient le processus de développement d'applications SDN.

L'application s'exécute sur le contrôleur Ryu, responsable de la gestion du réseau. Grâce à son apprentissage automatique, l'application est capable de reconnaître et découvrir dynamiquement les commutateurs actifs présents dans le réseau sans nécessiter une configuration manuelle préalable. Cela lui permet d'ajouter une couche de sécurité en appliquant des règles de pare-feu sur chaque commutateur détecté. Elle prend en charge la segmentation du réseau à l'aide des ID vlan ce qui permet une automatisation efficace de l'application des règles de pare-feu spécifiques à chaque vlan.

La figure suivante illustre processus de démarrage et d'intégration de l'application dans un environnement SDN :

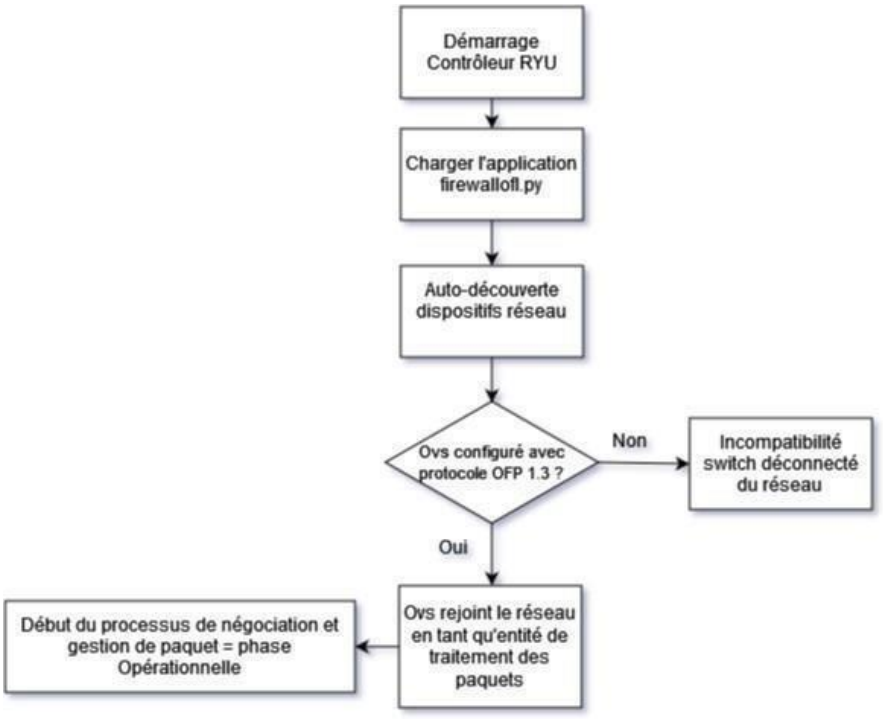

Figure 30 : Diagramme de démarrage de l'application.

Voici un extrait du code représentatif de la capacité d'auto-découverte dans le réseau :

La classe FirewallOfsList est utilisée pour gérer une liste de commutateurs, elle hérite de la classe dict et fournit une méthode appelée get_ofs qui permet de récupérer les commutateurs en fonction de leur ID. (Voir Figure 31)

```
class FirewallOfsList(dict):
    def __init__(self):
        super(FirewallOfsList, self).__init__()

    def get_ofs(self, dp_id):
        if len(self) == 0:
            raise ValueError('firewall sw is not connected.')
        dps = {}
        if dp_id == REST_ALL:
            dps = self
        else:
            try:
                dpid = dpid_lib.str_to_dpid(dp_id)
            except:
                raise ValueError('Invalid switchID.')

            if dpid in self:
                dps = {dpid: self[dpid]}
            else:
                msg = 'firewall sw is not connected. : switchID=%s' % dp_id
                raise ValueError(msg)
        return dps
```

Figure 31 : code représentatif de la capacité d'auto-découverte des commutateurs.

L'application supporte les protocoles à savoirs : TCP, UDP et ICMP. L'administrateur configure la politique de sécurité souhaitée à travers l'interface RESTful. Lorsqu'une demande est reçue, l'application traite les informations fournies et génère les règles de sécurité correspondantes. Ces règles sont ensuite programmées automatiquement sur les commutateurs via le protocole OpenFlow, qui les appliquent au trafic réseau en conséquence. La figure 32 explique le mécanisme de l'application :

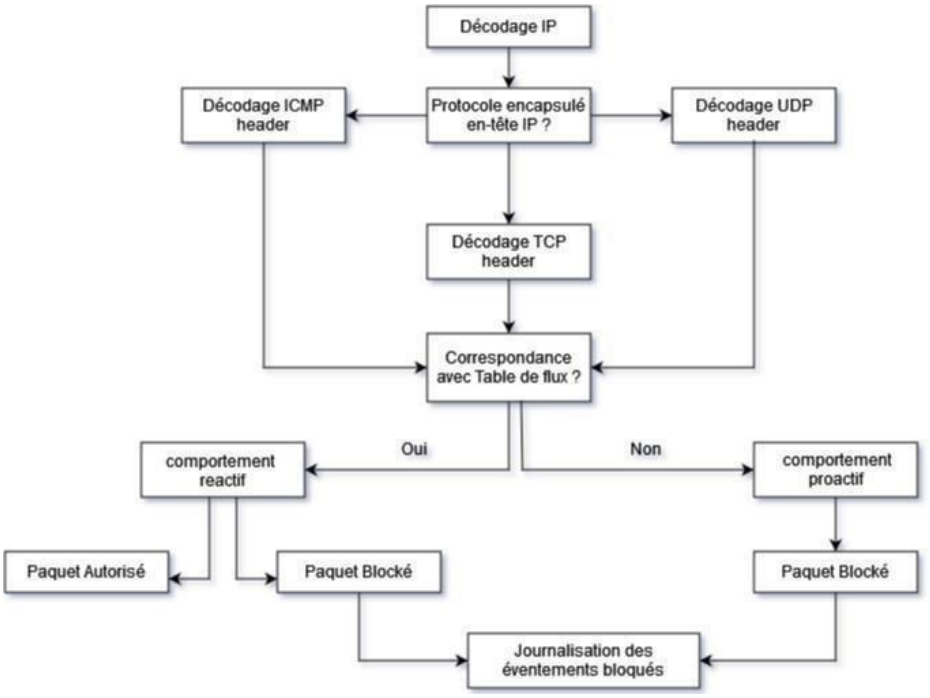

Figure 32 : Mécanisme application firewallofl.py

Tout au long du processus, l'application surveille en permanence le trafic réseau, détecte les connexions indésirable, grâce à l'automatisation, elle offre un comportement proactif pour bloquer ces connexions, et génère des journaux (logs) pour enregistrer les événements de sécurité détectés. Ces journaux sont automatiquement affichés sur le terminal du contrôleur offrant une visibilité en temps réel et simplifiant la gestion des informations de journalisation. (Voir figure 33)

```
    @staticmethod
    def packet_in_handler(msg):
        pkt = packet.Packet(msg.data)
        dpid_str = dpid_lib.dpid_to_str(msg.datapath.id)
        FirewallController._LOGGER.info('dpid=%s: Blocked packet = %s',
                                         dpid_str, pkt)
```

Figure 33 : code représentatif de la capacité de journalisations des connexions bloqués.

La fonction packet_in_handler est un gestionnaire d'événements qui s'exécute lorsque le contrôleur Ryu reçoit un paquet entrant qui a été bloqué par le pare-feu. Son rôle est d'enregistrer ce paquet dans le journal du contrôleur.

Le code python firewall a été enregistré dans le répertoire Ryu sous le nom firewallofl.py, la figure 34 représente l'enregistrement du code de l'application.

Figure 34 : Emplacement code firewallodl.py

4.4 Expérimentation :

Dans la section des expériences, notre objectif est de mettre en œuvre et de tester l'application de pare-feu distribué dans le contexte de notre architecture SDN. Notre principale préoccupation est d'évaluer le fonctionnement de l'application et de vérifier sa fiabilité.

4.4.1 Connexion de la topologie au contrôleur Ryu :

Afin de démarrer l'instance de Mininet (voir figure 35) avec notre topologie personnalisée provenant du fichier NetworkTopo.py et de la connecté au contrôleur Ryu, il est nécessaire d'exécuter la commande suivante :

```
huyem@huyem-VirtualBox:~$ sudo mn --controller=remote,ip=192.168.43.37 --custom
 mininet/custom/NetworkTopo.py --topo networktopo --mac --switch ovsk -x
```

On a spécifié que le contrôleur Ryu sera exécuté à l'adresse IP 192.168.43.73 et se connectera en tant que contrôleur distant.

```
Setting remote controller to 192.168.43.37:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
A1 A2 A3 A4 C1 D1 D2
*** Adding links:
(C1, D1) (C1, D2) (D1, A1) (D1, A2) (D2, A3) (D2, A4) (h1, A1) (h2, A1) (h3, A2
) (h4, A2) (h5, A3) (h6, A3) (h7, A3) (h8, A4) (h9, A4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Running terms on :0
*** Starting controller
c0
*** Starting 7 switches
A1 A2 A3 A4 C1 D1 D2 ...
*** Starting CLI:
mininet>
```

Figure 35 : Topologie sous Mininet.

Le contrôleur Ryu doit être en cours d'exécution et chargé de l'application firewallofl.py, comme le montre la figure 36 :

```
huyem@huyem-VirtualBox:~$ ryu-manager ryu.app.firewallofl
loading app ryu.app.firewallofl
loeading app ryu.controller.ofp_handler
instantiating app None of DPset
creating context dpset
creating context wsgi
instantiating app ryu.app.firewallofl
instantiating app ryu.controller.OFPHandler
(1698) wsgi starting up on http://0.0.0.0:808
```

Figure 36 : Ryu en cours d'exécution.

ryu-manager agit comme un gestionnaire de démarrage pour le contrôleur Ryu, permettant de charger et d'exécuter l'applications Ryu en question.

Après une connexion réussite entre les 7 commutateurs de la topologie et le contrôleur, la sortie suivante apparaît :

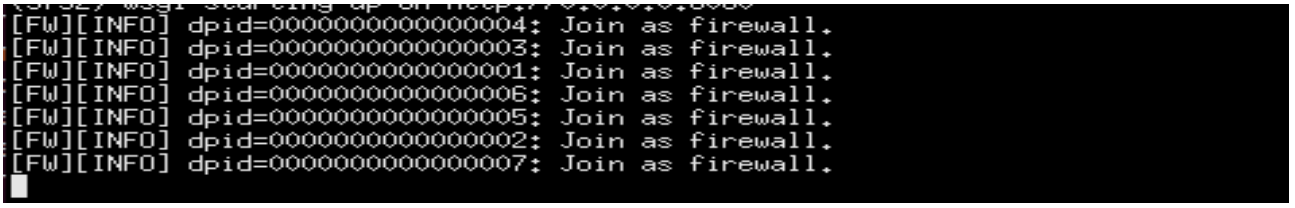

Figure 37 : Connexion réussite entre les commutateurs et le contrôleur.

Une fois l'application Firewall est lancée, l'état des pares-feux sont initialement désactivés par défauts et peuvent être activés en fonctions des besoins de sécurité de la topologie. (Voir figure 38)

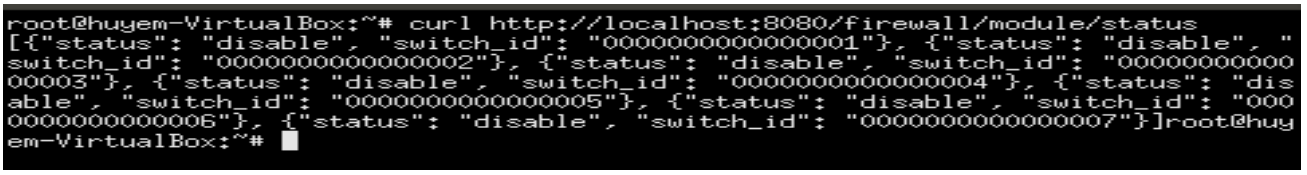

Figure 38 : Etat initiale des 7 Firewall.

4.4.2 Expérience 1 : Implémentation du Firewall Distribué au niveau couche Access :

Le tableau suivant présente les règles de pare-feu appliquées au niveau de la couche d'accès, en prenant compte le commutateur concerné, ainsi que les informations à savoir : la machine source, la machine destination, les protocoles et les actions de filtrage a effectués. Il offre une vue complète des politiques de sécurité mises en place dans le cadre de l'implémentation du Firewall Distribué.

Switch	Machine Src	Machine Dst	Protocole	Port	Vlan ID	Actions
A3	RH-USER=h5	GP-USER=h6	ICMP	-	110	ALLOW
A4	DMG-USER=h8	AdminIT=h9	ICMP	-	-	DENY

Tableau 3 : Règles Pare-feu Couche Access.

Switch A3 : Intégration des Règles de Pare-feu pour les Vlan

Les commandes suivantes ont été exécutées pour configurer l'interface réseau Vlan des machines h5 et h6, tel que représenté dans la Figure 39.

Figure 39 : Configuration Vlan SDN.

La requête suivante permet d'activer l'application de pare-feu sur le commutateur spécifié A3. En cas de succès, elle renvoie les données au format JSON, conformément à l'illustration présentée dans la figure 40.

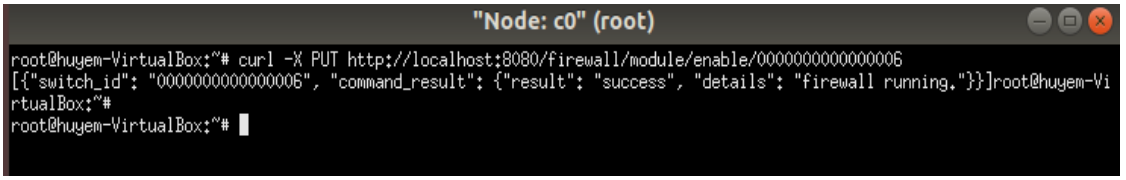

Figure 40 : Activé pare-feu sur switch A3.

Les requêtes illustrées dans la figure 41 sont utilisées pour créer une règle de pare-feu dans le cadre de la segmentation du réseau Vlan. Ces requêtes prennent en compte les identifiants spécifiques du Vlan et du commutateur en question.

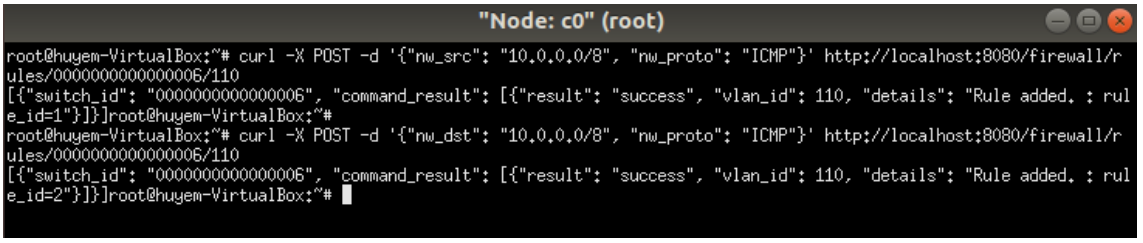

Figure 41 : Intégration des Règles Firewall pour les Vlan.

Les règles de pare-feu sont traduites en règles de flux par la suite configuré d'une façon automatique sur les commutateurs. Ces derniers définissent le comportement du trafic réseau en fonction des conditions spécifiées.

La commande présentée dans la figure 42 permet de visualiser la sortie des règles de flux configurées sur le commutateur OpenVswitch A3. Elle récupère et affiche les règles de flux actuellement en place via le protocole OpenFlow version 1.3. Cela permet d'obtenir une vue détaillée sur comportement du trafic réseau au niveau du commutateur.

Chaque ligne correspond à une règle de flux, indiquant des informations telles que le nombre de paquets et d'octets correspondants, la priorité de la règle, ainsi que les actions à effectuer en fonction des conditions spécifiées.

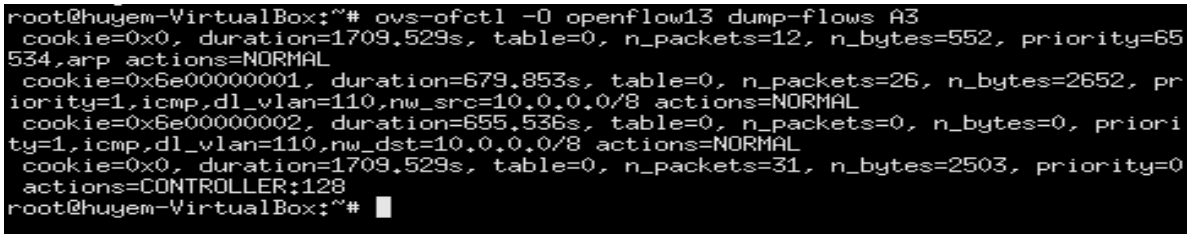

Figure 42 : Output Ovs A3.

Résultats :

La figure ci-dessus illustre une connectivité réussie entre les machines situées sur le même segment du réseau. Les résultats du test effectué à l'aide de la commande "ping" confirment que les paquets ICMP ont été envoyés et reçus avec succès, indiquant une communication fluide.

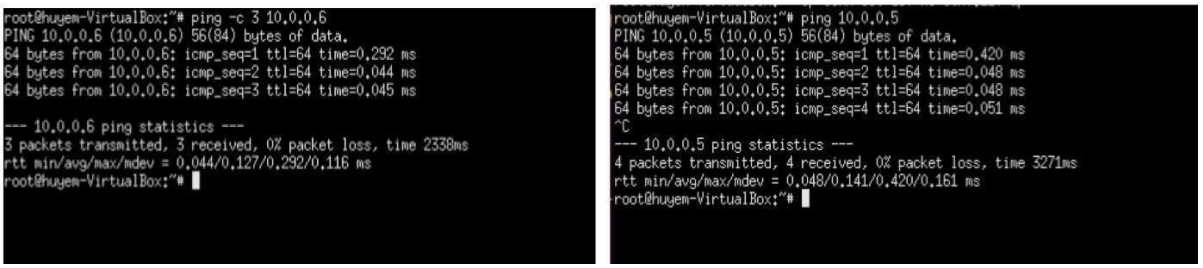

Figure 43 : Résultats du test de connectivité entre les machines h5 et h6.

Switch A4 :

L'illustration 44 présente la sortie du commutateur OpenVswitch A4 :

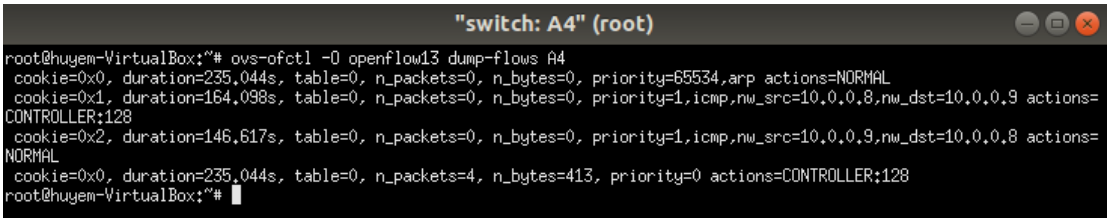

Figure 44 : Output Ovs A4.

L'expression actions=CONTROLLER :128 indique que les paquets correspondants à cette règle seront envoyés au contrôleur avec une priorité de 128. Cela signifie que lorsque des paquets ICMP provenant de l'adresse IP source 10.0.0.8 et se dirigeant vers l'adresse IP de destination 10.0.0.9 sont détectés, bloqués et par la suite ils seront transférés au contrôleur. Cette action permettra au contrôleur de générer automatiquement des journaux (logs) pour chaque paquet transmis. (Voir figure 46)

Résultats :

Les résultats de perte de paquets entre les machines h8 et h9 :

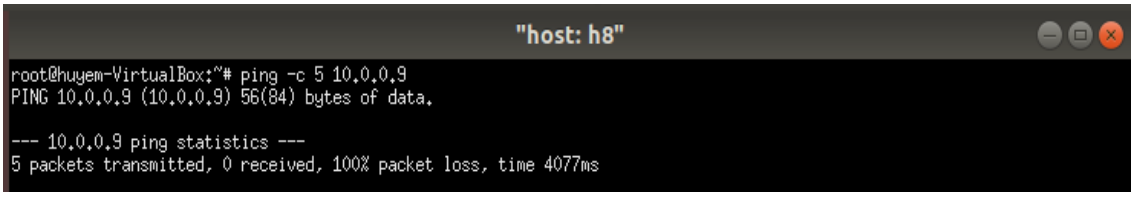

Figure 45 : Résultats du test de connectivité machine h8 vers h9.

Figure 46 : Journal des paquets bloqués par le pare-feu 1.

Les logs associés contiennent des informations sur l'événement de blocage, telles que : Dpid du commutateur, les informations relatives à la couche Ethernet du paquet, cela comprend les adresses MAC source (src), de destination (dst) du paquet et des informations sur la couche IP du paquet, cela inclut les adresses IP source (src) et de destination (dst), le type de protocole (proto), la longueur totale du paquet (total_length), le temps de vie (TTL).

Constat :

L'implémentation d'un pare-feu distribué au niveau de la couche d'accès offre de nombreux avantages significatifs en appliquant des politiques de sécurité directement sur les switchs d'accès, elle restreint la propagation des menaces potentielles à travers le réseau. De plus, en déployant le pare-feu au plus près des utilisateurs finaux, les décisions de filtrage et de contrôle du trafic sont prises de manière plus efficace et optimales. Cela réduit la charge sur le réseau central en permettant le traitement local du trafic indésirable au niveau de la couche d'accès.

4.4.3 Expérience 2 : Implémentation du Firewall Distribué au niveau couche Distribution :

Le tableau ci-dessous offre un aperçu des règles de pare-feu appliquées à la couche de distribution :

Switch	Machine-Src	Machine-Dst	Protocol	Port	Action
D2	AdminIT=h9	DJ=h7	ICMP	-	Allow

Tableau 4 : Règles Pare-feu Couche Distribution.

Résultats :

La figure ci-dessous illustre un ping réussi de la machine h9 vers h7

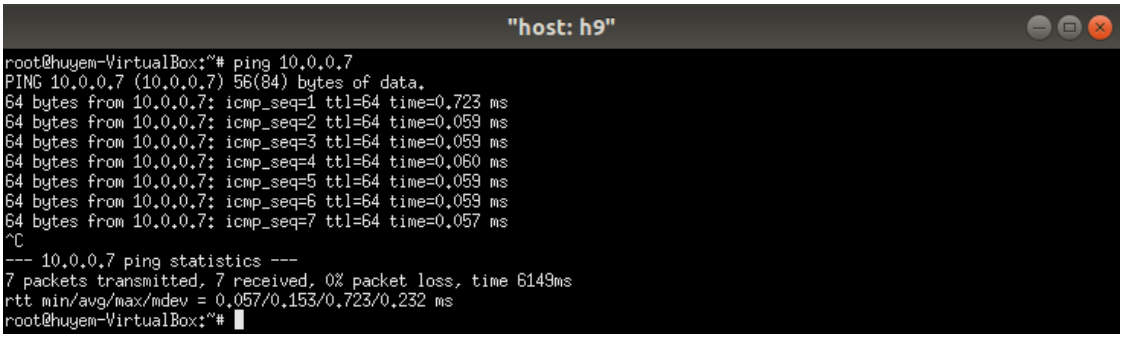

Figure 47 : Résultats du test de connectivité h9 vers h7.

Dans cette section, nous allons tester notre application firewallofl pour évaluer son mécanisme d'automatisation et sa capacité à détecter de manière proactive le trafic non autorisé. Lorsqu'une connexion indésirable est détectée, le pare-feu génère automatiquement une règle correspondante. Cette règle est ensuite traduite en règles de flux et configurée automatiquement sur le commutateur D2.

Nous effectuerons un ping de la machine h7 vers h9. (Voir figure 48)

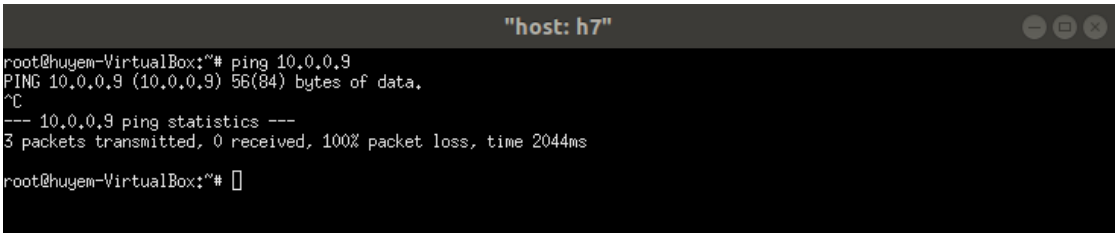

Figure 48 : Résultats du test de connectivité machine h7 vers h9.

Parallèlement, des logs seront générés pour enregistrer ces événements. (Voir figure 49)

Figure 49 : Journal des paquets bloqués par le pare-feu 2.

La figure 50 suivante présente la sortie du commutateur OpenVswitch D2 :

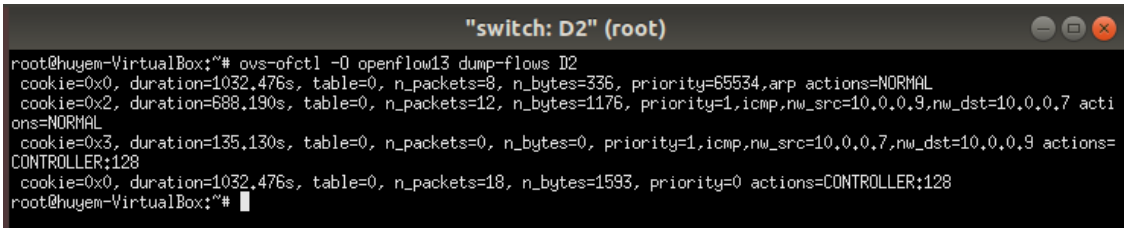

Figure 50 : Output Ovs D2.

Constat :

En déployant le filtrage au niveau de la couche de distribution, on parvient à bloquer le trafic malveillant avant qu'il ne puisse atteindre les couches plus profondes du réseau (couche Access). Cette approche réduit la charge de traitement au niveau distribution et prévient la propagation du trafic nocif à travers l'ensemble de l'infrastructure réseau.

4.4.4 Expérience 3 : Implémentation du Firewall Distribué au niveau couche Core :

Pour cette section expérimentale, nous avons déployé les serveurs suivants :

- Serveur WEB et serveur SMB : Les machines h1 et h4 sont dédiées à l'hébergement d'un serveur web et d'un serveur SMB respectivement. Les serveurs utilisent SimpleHTTPServer, une solution basée sur Python, et fonctionne sur le port 80.

- Serveur DHCP et DNS : Les machines h2 et h3 jouent le rôle de serveurs DHCP et DNS. Elles utilisent le protocole UDP et sont configurées pour écouter sur le port 53. L'utilitaire "nc" (netcat) est utilisé pour la configuration de ces serveurs.

Ces différentes configurations nous permettent de mettre en place des services essentiels au sein de notre réseau expérimental.

Le tableau suivant présente les règles de pare-feu appliquées au niveau de la couche Core :

Switch	Utilisateurs	Serveurs	Protocole	Port	Action	Rule Id
C1	h9	h1	TCP	80	Allow	1-2
C1	h9	h3	UDP	53	Allow	3-4
C1	h7	h2	UDP	53	Deny	5-6
C1	h7	h4	TCP	80	Deny	7-8

Tableau 5: Règles Pare-feu Couche Core.

La sortie ci-dessous donne un aperçu détaillé des politiques de filtrage et de contrôle du trafic appliquées par le pare-feu sur le Switch Core C1 :

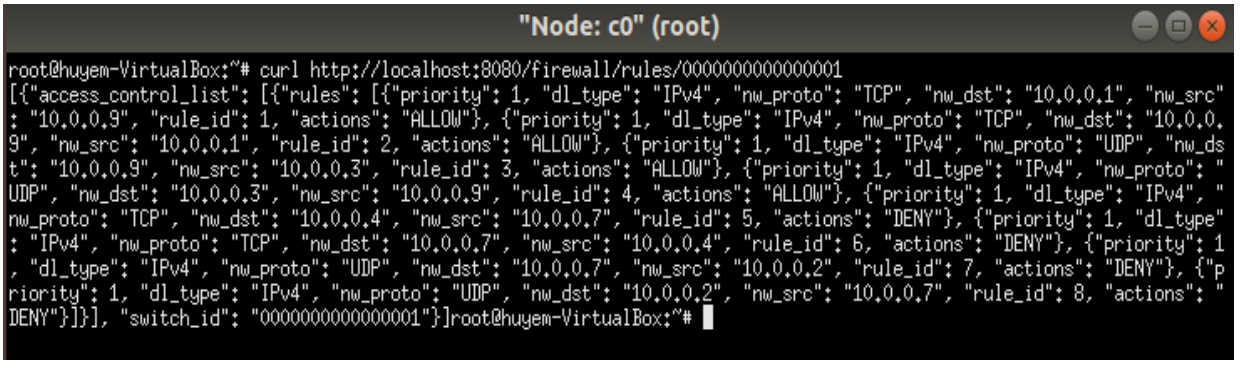

Figure 51 : Confirmation des règles pare-feu C1.

La sortie du commutateur OpenVswitch A4 est représentée dans l'illustration 52 :

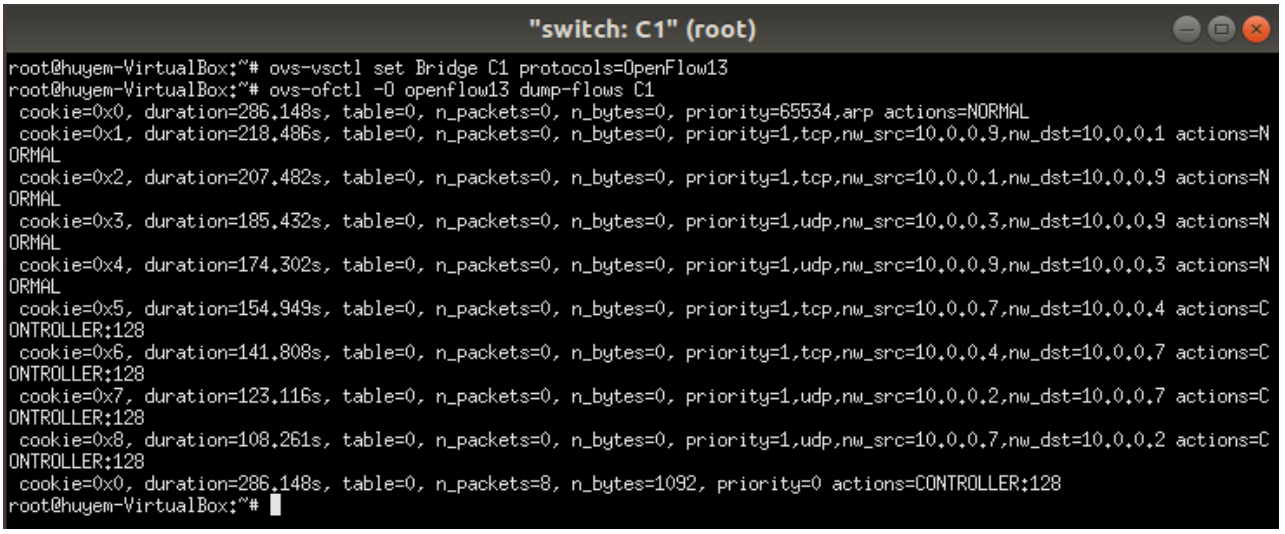

Figure 52 : Output Ovs C1.

Résultats :

Nous avons effectué un test de trafic UDP en commençant par mettre en place le serveur UDP sur le port 53 de l'hôte h3 en utilisant l'utilitaire netcat en mode serveur. "nc" et h9 en mode client. Nous avons envoyé une chaîne de caractères à notre serveur UDP et les résultats du test ont confirmé que les chaînes envoyées ont été correctement reçus par le serveur UDP, comme le montre la figure 53.

Figure 53 : Test de Connectivité UDP 1.

En utilisant la commande "wget" sur le client h9, nous avons réalisé un test de trafic web. Les résultats de ce test indiquent que la machine h9 parvient à accéder au serveur Web.

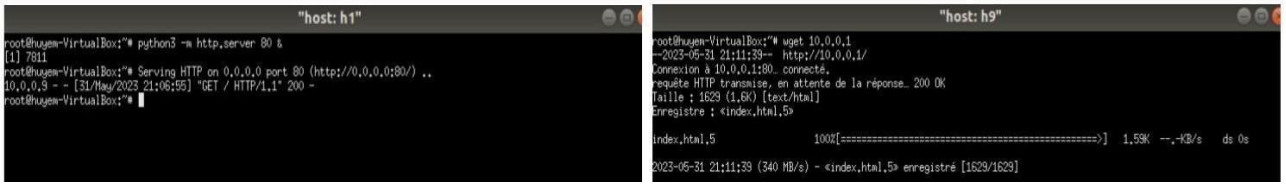

Figure 54 : Test de Connectivité TCP 1.

La figure suivante illustre test de connectivité entre client h7 et server SMB h4 :

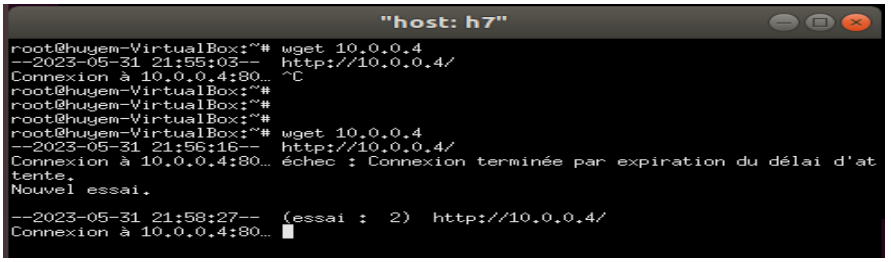

Figure 55 : Test de Connectivité TCP 2.

En temps réel, des journaux (logs) sont générés et affichés sur le terminal du contrôleur, comme illustré dans la figure 56 :

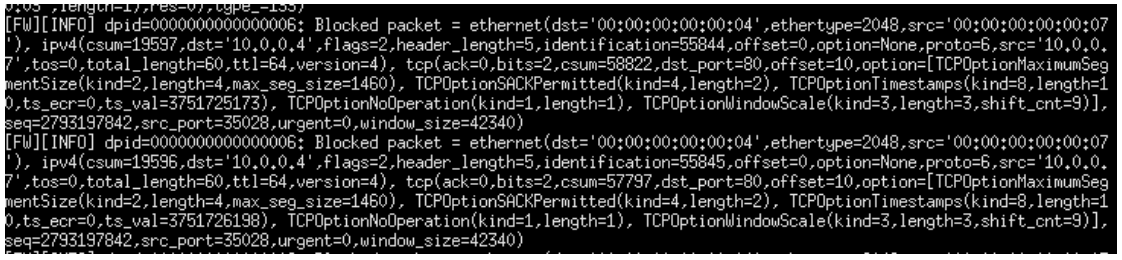

Figure 56 : Journal des paquets TCP bloqués par le pare-feu.

La figure 57 représente un test de connectivité entre le client h7 et le serveur h2. Cependant, les résultats du test ont confirmé que les chaînes envoyées ne sont pas reçues par le serveur DNS :

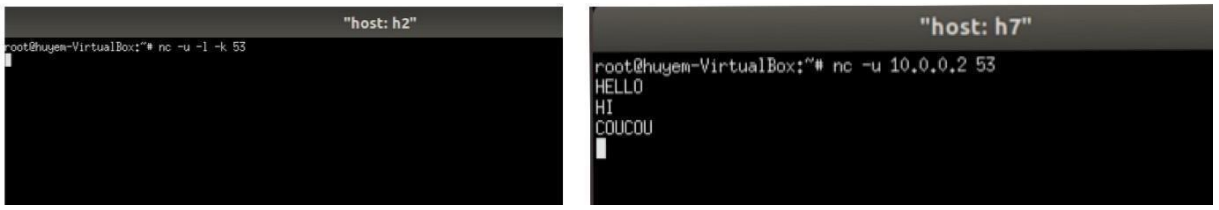

Figure 57 : Test de Connectivité UDP 2.

En outre, la figure 58 ci-dessous illustre que ces chaînes de caractères sont continuellement affichées dans le journal des événements bloqués.

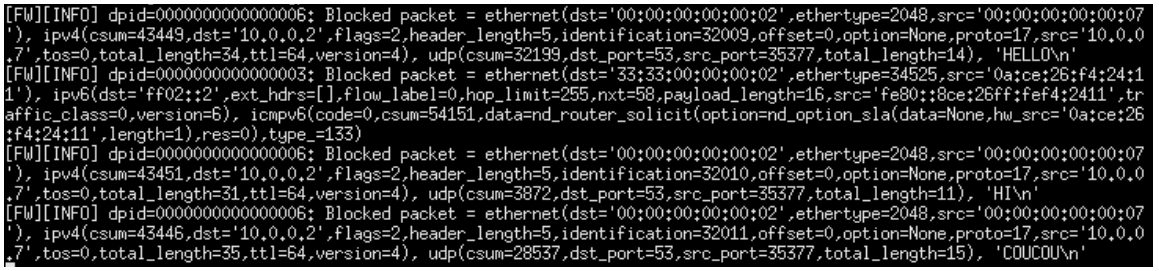

Figure 58 : Journal des paquets UDP bloqués par le pare-feu.

D'après les tests effectués au niveau core, il a été constaté que notre pare-feu fonctionne correctement et est capable de gérer les paquets TCP etUDP.

Constat :

En déployant des pare-feux distribués basés sur OpenFlow aux niveaux access et distribution, on décharge une partie du traitement des règles de pare-feu du switch core, ce qui lui permet de se concentrer sur la gestion du trafic sensible entre les serveurs et les utilisateurs. Ainsi, les pare-feux situés aux niveaux access et distribution prennent en charge une partie du traitement des règles de pare-feu, ce qui allège la charge sur le switch core.

L'architecture three-tier choisie pour implémenter cette solution permet de réduire les risques d'incohérence entre les règles de pare-feu. En spécifiant des règles spécifiques à chaque couche, on répartit de manière appropriée les règles de filtrage, évitant ainsi les répétitions des règles sur d'autres couches. Chaque couche est configurée de manière indépendante, ce qui facilite la mise à jour des règles spécifiques à cette couche sans affecter l'ensemble du service de pare-feu.

Enfin, l'architecture à trois niveaux offre une plus grande flexibilité et évolutivité. Chaque couche peut être mise à jour et améliorée de manière indépendante, sans affecter les autres couches. Cela facilite l'ajout de nouvelles fonctionnalités ou l'adaptation aux changements dans l'environnement réseau.

En conclusion, l'adoption d'une architecture à trois niveaux pour l'implémentation des pare-feux distribués basés sur OpenFlow présente de nombreux avantages, notamment une répartition efficace de la charge, une réduction des risques d'incohérence et une plus grande flexibilité. Cela contribue à améliorer les performances et la gestion du pare-feu dans l'environnement réseau.

4.5 Comparaison des résultats de performance pré et post-implémentation du pare-feu distribué OpenFlow :

Cette section présente une analyse comparative des performances de l'application en termes de temps de réponse (latence) avant et après l'implémentation du pare-feu distribué. L'objectif est d'évaluer l'impact de cette mise en place sur la latence du réseau.

Pour effectuer cette évaluation, la commande ping a été utilisée pour mesurer le temps de réponse entre deux nœuds du réseau h7 et h9. Les tests ont été réalisés avant l'implémentation du pare-feu distribué, puis répétés après sa mise en place.

La latence moyenne d'établissement des flux est définie comme le temps nécessaire à un paquet pour aller de la source à une destination et revenir.

Pour chaque test, le temps de réponse moyen en millisecondes (ms) est mesuré en utilisant la commande ping. Cette commande envoie des paquets ICMP echo=request à une adresse spécifiée, puis attend une réponse ICMP echo-reply.

Les résultats obtenus ont été présentés dans les figures 59 et 60, qui illustrent les données des tests réalisés avant et après l'implémentation du pare-feu distribué au respectivement :

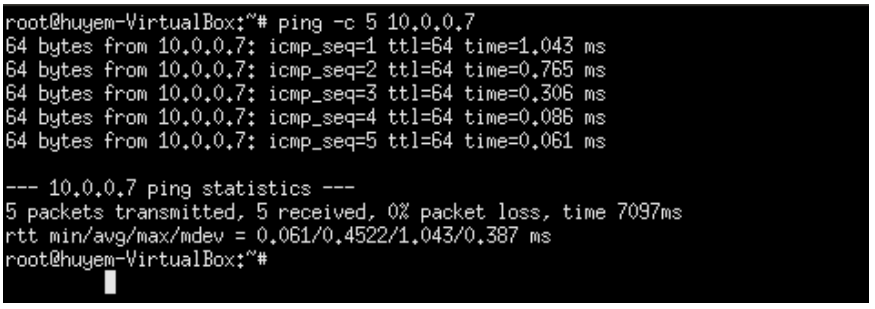

Figure 59 : Ping avant l'implémentation du pare-feu distribué.

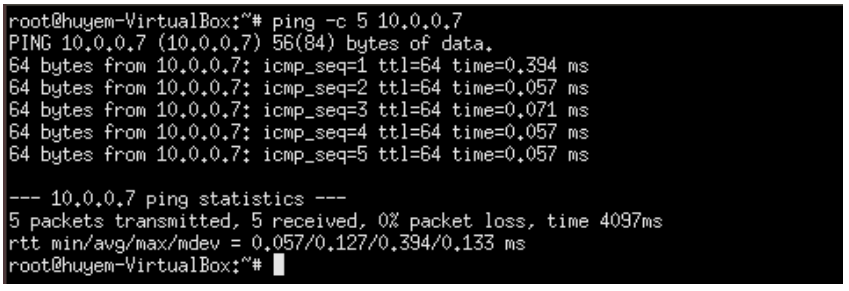

Figure 60 : Ping après l'implémentation du pare-feu distribué.

Les temps de réponses ont été représentées graphiquement pour les deux scénarios : avant et après l'implémentation du pare-feu OpenFlow sur le commutateur D2. Le graphique permet de visualiser la variation de la latence entre les deux situations. (Voir figure 61)

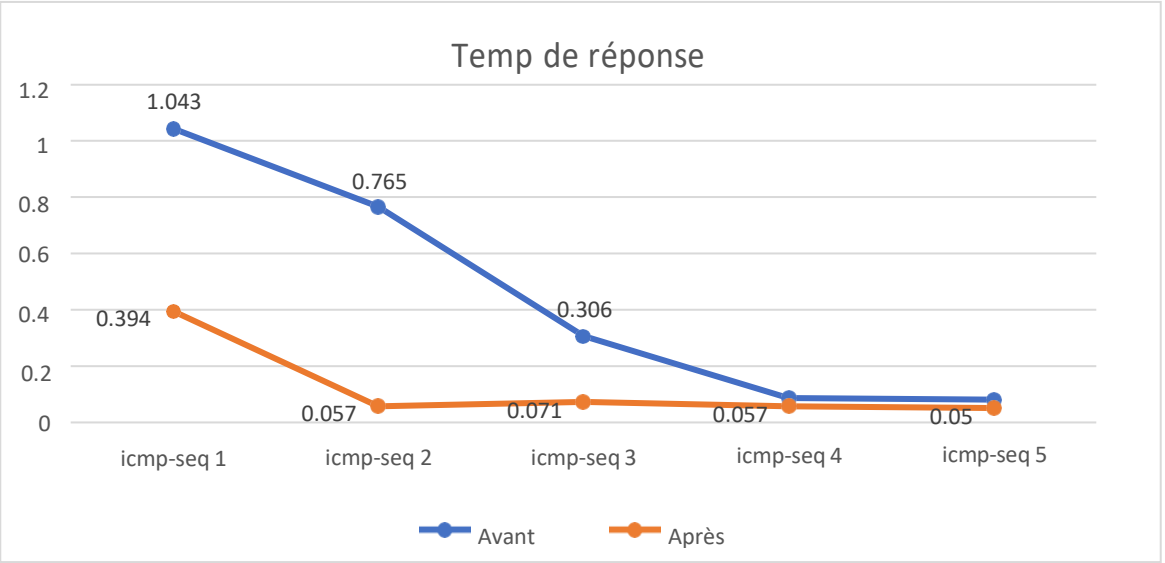

Figure 61 : temps de réponse avant et après l'implémentation du firewallofl sur D2.

Dans le premier cas où l'application firewall distribuée n'est pas mise en place, le temps de réponse mesuré lors de la première tentative de ping est de 1.043 ms.

En revanche, dans le cas où le pare-feu distribué est implémenté au niveau du commutateur de distribution D2, le temps de réponse est réduit à 0.394 ms. Cette différence s'explique par les raisons suivantes :

En l'absence du pare-feu distribué, lorsque le commutateur D2 reçoit un paquet, il ne dispose pas d'une entrée de flux correspondante préinstallée pour le traiter localement. Par conséquent, le commutateur doit envoyer une demande au contrôleur afin d'obtenir les instructions nécessaires pour le traitement du paquet. Cette communication entre le commutateur et le contrôleur introduit un délai supplémentaire, ce qui se traduit par un temps de réponse plus long de 1.043 ms.

Lors de la mise en place du pare-feu distribué sur le commutateur D2, les règles de filtrage sont automatiquement traduites et configurées sur ce commutateur spécifique. En conséquence, lorsque les paquets sont reçus par D2, ils peuvent être rapidement traités en appliquant les règles de filtrage correspondantes localement. Cela permet de réduire le temps de réponse, car il n'est pas nécessaire de transférer les paquets vers le contrôleur pour un traitement supplémentaire. Cette approche améliore l'efficacité du pare-feu en permettant une prise de décision locale.

Pour des informations plus prédiseuse sur les performances de réseau en termes de temps de réponse, ces statistiques de latences mettent en évidence les différentes mesures et facilite la compréhension des résultats avant et après l'intégration d'un pare-feu distribué basé open sur commutateurs de distribution D2, à savoir :

- rtt min : C'est la valeur minimale de la latence mesurée, indiquant le temps le plus court en millisecondes qu'il a fallu pour que les paquets atteignent la destination et reviennent.

- rtt avg : C'est la valeur moyenne de la latence mesurée, représentant la moyenne des temps de réponse de tous les paquets envoyés.
- rtt max : C'est la valeur maximale de la latence mesurée, indiquant le temps le plus long en millisecondes qu'il a fallu pour que les paquets atteignent la destination et reviennent.
- mdev : C'est l'écart moyen (deviation) de la latence mesurée par rapport à la valeur moyenne. Il donne une indication de la variation de la latence.

En représentant ces résultats, il est possible de visualiser clairement les différents arguments et comprendre l'impact de l'application sur la latence : Après / Avant (Voir figure 62)

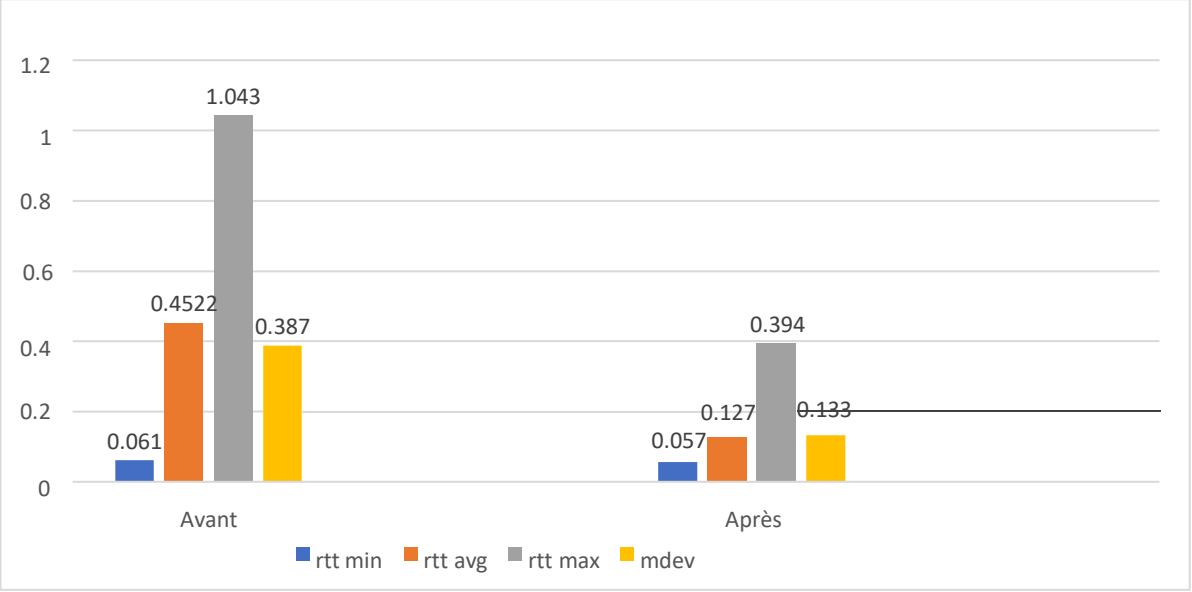

Figure 62 : Variation du temps de réponse (RTT) avant et après l'implémentation du pare-feu distribué

Discussion :

L'utilisation de ces statistiques de latence permet de mesurer et de comparer objectivement les performances du réseau avant et après l'intégration du pare-feu distribué.

En analysant les statistiques de latence avant et après l'intégration du pare-feu distribué, plusieurs observations peuvent être faites. Tout d'abord, une réduction significative du rtt min, rtt avg et rtt max est constatée après l'implémentation. Cela indique une amélioration des performances du réseau, avec des temps de réponse plus courts en général. Par exemple, le rtt min est passé de 0.061ms à 0.057 ms, le rtt avg est passé de 0.452 ms à 0.127 ms et le rtt max est passé de 1.043ms à 0.394 ms.

De plus, une diminution de la valeur mdev est également observée. Avant l'implémentation du pare-feu distribué, la mdev était de 0.397 ms, tandis qu'après l'implémentation, elle est passée à 0.133 ms. Cette réduction de la mdev suggère une plus grande stabilité et cohérence dans les mesures de latence, ce qui est bénéfique pour la performance globale du réseau.

Ces observations mettent en évidence les améliorations significatives apportées par l'intégration du pare-feu distribué en termes de temps de réponse et de stabilité du réseau. Ces résultats renforcent l'efficacité et l'optimisation des performances obtenues grâce à cette solution.

Malgré l'ajout d'une couche de sécurité supplémentaire avec le pare-feu, l'impact sur la latence a été maîtrisé.

Le tableau suivent permis de souligner les avantages de l'approche distribuée, tels que la flexibilité, l'évolutivité, les performances améliorées, la réduction de la latence et une meilleure fiabilité grâce à la répartition de la charge sur plusieurs couches de la topologie. Il a également mis en évidence certains aspects à prendre en compte, tels que la sécurité de l'appareil central et les coûts potentiels d'investissement en matériel dédié.

Cette comparaison aide à mettre en évidence les bénéfices et les considérations liés à l'implémentation d'un pare-feu distribué basé sur OpenFlow dans une architecture réseau : (Voir Tableau 5)

Aspect	Pare-feu distribué basé sur OpenFlow	Pare-feu traditionnel
Flexibilité	Permet des modifications rapides et flexibles des règles de filtrage à travers le contrôleur SDN	Requiert des configurations statiques et des mises à jour manuelles
Contrôle centralisé	Permet une gestion centralisée et un contrôle précis du trafic à partir du contrôleur SDN	La gestion et le contrôle sont répartis sur les dispositifs individuels du pare-feu
Évolutivité	Facilement étendu pour prendre en charge de nouveaux services ou applications	Nécessite des modifications et des mises à jour manuelles sur chaque appareil
Analyse du trafic en temps réel	Offrir des fonctionnalités d'analyse en temps réel grâce à l'accès aux informations du plan de données via le contrôleur SDN	Peut avoir des limitations en termes de capacités d'analyse en temps réel
Latence	Réduit la latence en appliquant les règles localement sur les commutateurs	La latence peut être plus élevée en raison du traitement centralisé
Fiabilité	Répartit la charge sur plusieurs commutateurs pour une meilleure résilience	Une défaillance de l'appareil central peut affecter tout le pare-feu

Sécurité	Fournit une sécurité renforcée en appliquant les règles de filtrage directement sur les commutateurs	Peut nécessiter des mécanismes supplémentaires pour assurer la sécurité de l'appareil central
Gestion et maintenance	Simplifie la gestion grâce à une interface centralisée pour la configuration des règles	Nécessite une configuration et une gestion manuelles sur chaque appareil
Intégration avec d'autres services et applications	Peut être intégré plus facilement avec d'autres services et applications basés sur SDN	Nécessiter des configurations spécifiques pour l'intégration avec d'autres services
Analyse intelligente du trafic	Prendre des décisions de filtrage basées sur des informations et des politiques spécifiques au réseau	Peut se baser sur des règles de filtrage statiques sans tenir compte du contexte du réseau
Coût	Réduire les coûts en évitant l'achat d'appareils spécifiques pour le pare-feu	Peut nécessiter des investissements plus importants en termes de matériel dédié

Tableau 6 : Aspects du pare-feu - OpenFlow vs Traditionnel.

En somme, le choix d'une architecture three-tier et le développement d'une application firewall distribuée basée sur OpenFlow offrent une approche efficace pour améliorer les performances, optimiser la gestion des règles de pare-feu et garantir la sécurité du réseau. Cette combinaison permet de bénéficier des avantages de la virtualisation du réseau et de l'automatisation de la gestion des règles, tout en assurant une adaptabilité et une évolutivité accrues pour répondre aux besoins changeants des environnements réseau.

4.6 Conclusion :

L'implémentation d'un pare-feu distribué basé sur le protocole OpenFlow 1.3 et l'architecture réseau SDN spécifique à l'organisme d'accueil offre une solution efficace pour renforcer la sécurité du réseau. L'intégration de l'application Firewall dans le contrôleur RYU permet une gestion centralisée du pare-feu et offre la flexibilité nécessaire pour appliquer des règles de pare-feu spécifiques aux flux de données. Les tests et l'analyse des performances ont confirmé l'efficacité de notre solution, en démontrant une latence minimale ajoutée au trafic réseau.

Cependant, il est important de noter que cette solution présente une limite potentielle en termes de sécurité. Étant donné que l'application Firewall dépend du contrôleur RYU, la sécurité du contrôleur devient cruciale. Si le contrôleur est compromis, cela pourrait affecter l'ensemble du système de pare-feu distribué.

Afin de garantir la sécurité globale de la solution, des mesures de sécurité supplémentaires doivent être mises en place pour protéger le contrôleur contre les attaques potentielles. Cela peut inclure des mesures telles que l'authentification et l'autorisation rigoureuses, la surveillance continue du contrôleur, l'utilisation de mécanismes de détection d'intrusion et la mise en œuvre de politiques de sécurité strictes.

En conclusion, bien que l'implémentation d'un pare-feu distribué basé sur OpenFlow 1.3 et l'architecture réseau SDN spécifique à l'organisme d'accueil offre des avantages significatifs en termes de flexibilité et de gestion centralisée, il est essentiel de mettre en place des mesures de sécurité solides pour protéger le contrôleur et garantir l'intégrité et la fiabilité de l'ensemble du système de pare-feu distribué.

Conclusion Générale

La réalisation de notre projet de fin d'études a été motivée par plusieurs facteurs importants dans le domaine des réseaux et de la sécurité. Tout d'abord, l'essor rapide du SDN a suscité un grand intérêt et a été considéré comme une évolution majeure de l'architecture des réseaux traditionnels. En comprenant l'importance croissante du SDN, nous étions motivés à explorer plus en détail cette technologie révolutionnaire et à comprendre son potentiel en matière de sécurité.

C'est dans ce contexte que nous avons envisagé d'étudier l'application du SDN dans le domaine de la sécurité, plus précisément en développant un pare-feu SDN.

Dans notre propre, nous avons tout d'abord introduit le concept du réseau SDN et l'avons comparé au réseau traditionnel. Nous avons mis en évidence les avantages du SDN tels que sa flexibilité, sa programmabilité et sa capacité à centraliser le contrôle du réseau.

Nous avons défini le concept du SD-FW, qui est un pare-feu spécifiquement conçu pour tirer parti des capacités du SDN. Contrairement aux pares-feux traditionnels, le SD-FW utilise un contrôleur SDN pour gérer et programmer les règles de pare-feu à travers le réseau. Cela permet une gestion centralisée des politiques de sécurité. Nous avons développé une application intégrée dans le contrôleur SDN, reproduisant les fonctionnalités d'un pare-feu traditionnel. Cette application, que nous avons appelée firewallofl, permet de gérer et de programmer les règles de pare-feu à travers le réseau SDN.

Pour réaliser cela, nous avons travaillé en étroite collaboration avec les protocoles et les fonctionnalités du SDN, notamment OpenFlow version 1.3, qui est un protocole largement utilisé pour la communication entre le contrôleur SDN et les commutateurs.

L'ensemble des objectifs définis pour ce projet a été pleinement réalisé, comprenant :

- Virtualisation du pare-feu jusqu'aux commutateurs d'accès : Nous avons réalisé la virtualisation du pare-feu, ce qui nous a permis d'étendre sa protection jusqu'aux Commutateurs d'accès du réseau, assurant ainsi une sécurité à tous les points d'accès.

- Amélioration de l'efficacité opérationnelle grâce à la centralisation et à l'automatisation : Nous avons mis en place des processus de gestion automatisé et centralisé des règles du pare-feu, ce qui a permis d'améliorer l'efficacité opérationnelle et de faciliter la configuration.

- Maîtrise de l'impact sur la latence : Malgré l'ajout d'une couche de sécurité, le traitement des paquets réseau reste rapide et efficace.

- Journalisation en temps réel des événements bloqués : Nous avons mis en place un mécanisme de journalisation en temps réel des événements bloqués par le pare-feu. Cela permet une analyse et surveillance proactive et une réponse rapide aux menaces.

Perspectives

Comme perspective, L'application Firewall combinée à une application de détection d'intrusion (IDS) et une application prévention d'intrusion (IPS) sur un contrôleur SDN, peut offrir une solution de sécurité avancée pour les réseaux SDN. Lorsque ces applications fonctionnent conjointement, elles peuvent bénéficier des logs générés par le pare-feu SDN pour renforcer la détection des intrusions et améliorer la réactivité face aux menaces.

Les logs générés par l'application sont précieux pour les applications IDS/IPS, car ils fournissent des informations détaillées sur les événements bloqués :

Les IDS peuvent utiliser les informations des logs pour identifier de nouvelles signatures ou règles de détection basées sur les schémas de trafic bloqués. Par exemple, si application Firewall bloque régulièrement des tentatives de connexion provenant d'une adresse IP spécifique, l'IDS peut créer une règle de détection spécifique pour cette adresse IP, renforçant ainsi la capacité de l'IDS à détecter et à alerter sur de futures tentatives d'intrusion provenant de cette adresse.

Les IPS peuvent utiliser les logs pour corréler les événements bloqués avec d'autres activités suspectes sur le réseau. Par exemple, si application Firewall bloque une tentative de connexion sur un port sensible, l'IPS peut analyser les logs des autres systèmes de sécurité pour vérifier si cette même adresse IP a été détectée par un système d'authentification comme ayant tenté des accès non autorisés. Cette corrélation d'événements permet à l'IPS de détecter les attaques plus sophistiquées qui pourraient impliquer plusieurs actions coordonnées.

En outre, l'efficacité des IDS/IPS dépend en grande partie de la qualité et de la mise à jour de leurs bases de données de cyberattaques. Les logs générés par l'application Firewall peuvent être considérés comme une forme de base de données. Ils contiennent des informations sur les événements bloqués et peuvent être utilisés pour identifier de nouvelles attaques ou schémas de trafic malveillants qui ne sont pas encore répertoriés dans les bases de données d'IDS/IPS.

Annexe A

MININET

Prérequis :

- Système d'exploitation : machine virtuelle Linux, comme Ubuntu sur VirtualBox.

Figure 63 : Ubuntu sur VirtualBox.

- Python : Installer Python 2.7.x ou Python 3.x sur le système.

- Outils de développement : Vérifier la présence des outils de développement requis, tels que make, gcc, make.

Étapes d'installation :

Cette commande permet de télécharger la dernière version de Mininet à partir du dépôt GitHub officiel et de créer une copie locale du code source sur votre machine. Assurer d'avoir Git installer sur votre système avant d'exécuter cette commande.

```
hoyem@hoyem-VirtualBox:~$ git clone https://github.com/mininet/mininet.git
Clonage dans 'mininet'...
remote: Enumerating objects: 10339, done.
remote: Counting objects: 100% (185/185), done.
remote: Compressing objects: 100% (122/122), done.
Réception d'objets:  85% (8789/10339), 2.66 MiB | 122.00 KiB/s
```

Figure 64 : Clonage dans mininet.

Une fois le clonage terminé, accéder au répertoire "mininet" nouvellement créé pour poursuivre l'installation et l'utilisation de Mininet.

Une fois dans le répertoire "mininet", poursuivre l'installation de Mininet en exécutant le script d'installation. Utilisez la commande suivante pour exécuter le script en tant qu'administrateur :

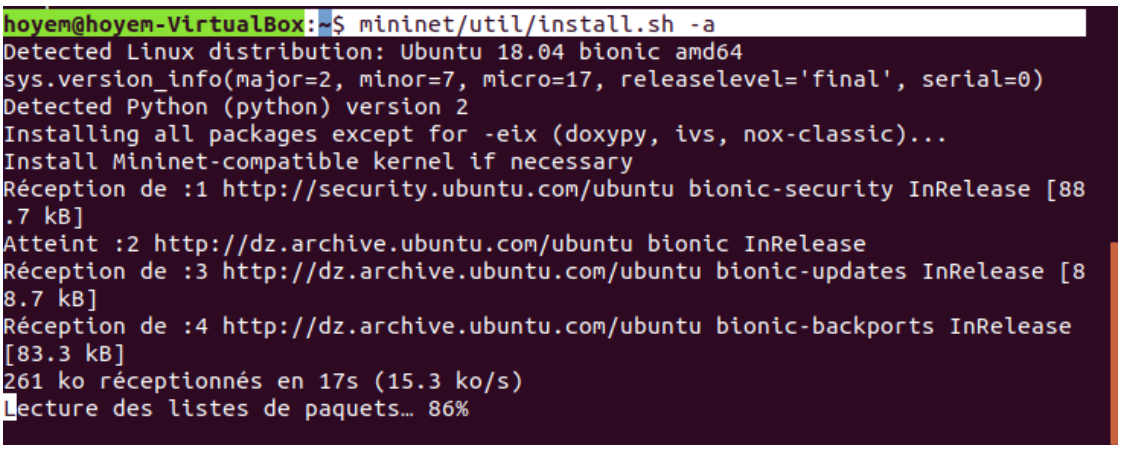

Figure 65 : Installation mininet.

Le processus d'installation de Mininet commencera. Suivez les instructions qui s'affichent à l'écran et acceptez les termes de la licence.

Une fois l'installation terminée, Mininet sera installé sur votre système.

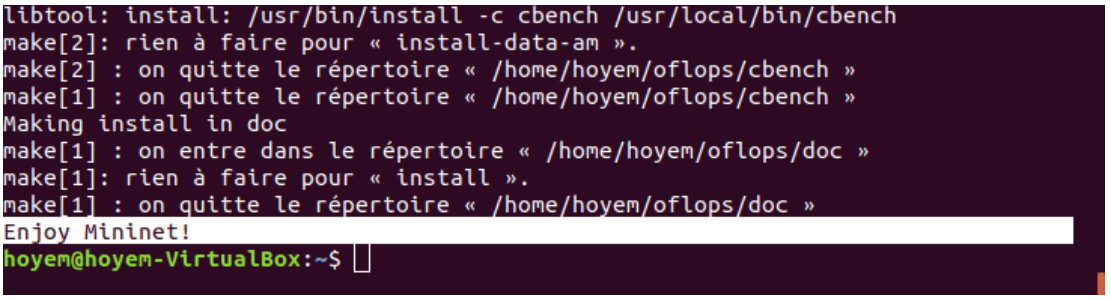

Figure 66 : Installation terminée mininet.

À ce stade, on peut explorer et créer des topologies Mininet en utilisant les fonctionnalités fournies par Mininet. Nous avons maintenant la possibilité de créer des réseaux virtuels à grande échelle en utilisant des hôtes simulés, des commutateurs et des contrôleurs SDN.

Annexe B

OpenVswitch

La commande wget http://www.openvswitch.org/releases/openvswitch-2.16.0.tar.gz est utilisée pour télécharger l'archive tar.gz d'Open vSwitch version 2.16.0 à partir du site officiel d'Open vSwitch

Figure 67 : Installation OVS.

La commande **sudo service ovs-switch status** est utilisée pour vérifier l'état du service Open Vswitch. En exécutant cette commande avec les privilèges d'administrateur (sudo), on obtient le statut actuel du service.

Si le service OpenVswitch est en cours d'exécution, un message indiquant que le service est actif et opérationnel est affiché. Sinon, le service n'est pas en cours d'exécution.

Cette commande est utile pour vérifier si le service OpenVswitch est démarré correctement et fonctionne comme prévu sur système.

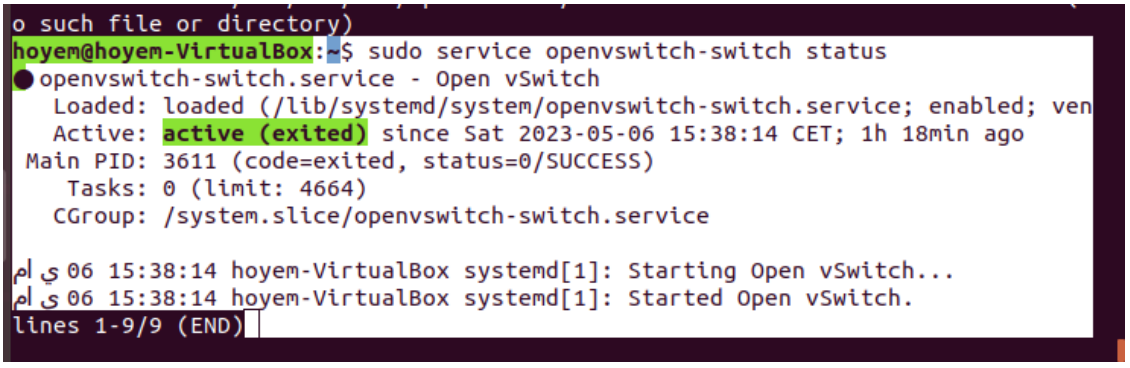

Figure 68 : Service OVS.

Annexe C

Contrôleur SDN

OpenDayLight :

Etape 1 :

Créer une machine virtuelle nommée "OpenDaylight" avec Ubuntu comme système d'exploitation. Télécharger la version la plus récente d'OpenDaylight, appelée Carbon, à partir du site www.opendaylight.org/downloads, qui est compatible avec Ubuntu 18.04.

Etape 2 :

Le contrôleur OpenDaylight SDN est un programme écrit sous le langage Java, une alternative pour installer l'environnement d'exécution pour le contrôleur SDN serait d'utiliser les commandes suivantes :

- apt-get install openjdk-8-jdk
- apt-get install openjdk-8-jre

Etape 3 :

Après avoir installé l'environnement d'exécution Java, exécuter les commandes suivantes dans le terminal Ubuntu :

1. Ouvrer le terminal.
2. Accéder au répertoire "ODL/bin" en utilisant la commande suivante :
    - cd ODL/bin
3. Lancer le contrôleur OpenDaylight en utilisant la commande suivante :
    - ./karaf -of13

Cela exécutera le script "karaf" avec l'option "-of13" pour lancer le contrôleur OpenDaylight en utilisant le protocole OpenFlow version 1.3. (Voir figure 69).

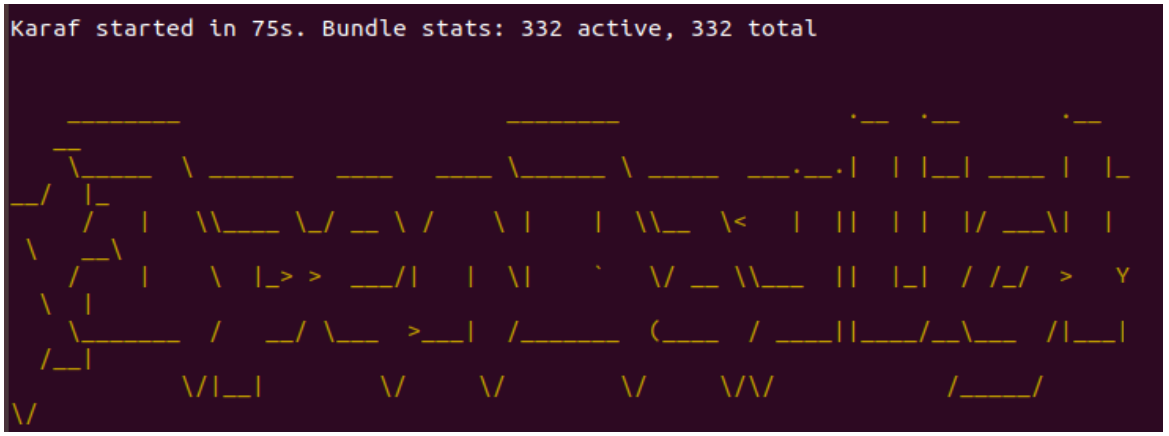

Figure 69 : Contrôleur ODL lancé.

Étape 4 :

Sur un deuxième terminal dans Ubuntu, exécuter les commandes suivantes :

- sudo apt-get install nmap

Cela installera l'outil de scanner de sécurité Nmap, qui est utilisé pour découvrir les hôtes et les services sur un réseau informatique et créer une "carte" du réseau.

- nmap localhost

Cela exécutera Nmap sur l'hôte local pour analyser les services et les ports ouverts.

Étape 5 :

Dans le premier terminal où les fonctionnalités Karaf ont été lancées, installer les fonctionnalités OpenDaylight à l'aide des commandes suivantes :

- feature:install odl-l2switch-switch-ui

Cela installera la fonctionnalité ODL L2Switch Switch UI, qui permet de gérer les commutateurs réseau virtuels dans OpenDaylight.

- feature:install odl-dlux-all

Cela installera toutes les fonctionnalités DLUX (Data Layer User Experience) d'OpenDaylight. DLUX est une interface utilisateur basée sur le web pour OpenDaylight, qui offre des fonctionnalités supplémentaires pour la gestion et la visualisation du réseau

Etape 5 :

Pour accéder à l'interface OpenDaylight (ODL), Dans la barre d'adresse du navigateur, saisire l'URL suivante : http://<adresse_IP_ODL>:8181/index.html.

Remplacer "<adresse_IP_ODL>" par l'adresse IP du serveur OpenDaylight.

Cela devrait charger l'interface OpenDaylight dans navigateur, permettant de gérer et de visualiser le réseau via l'interface utilisateur graphique (GUI) fournie par OpenDaylight. Assurer que le port 8181 est accessible. (Voir figure 70,71)

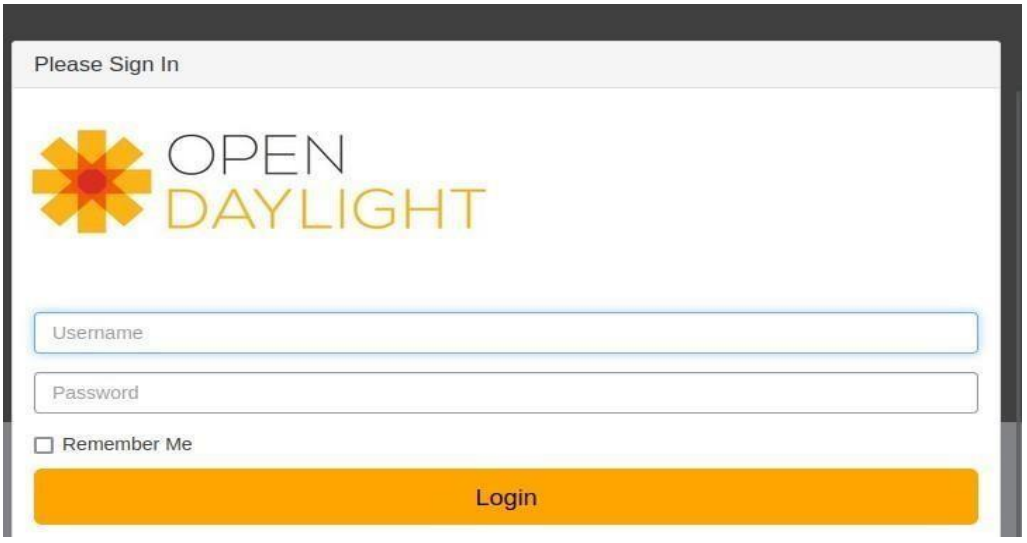

Figure 70 : Interface web représentant l'accueil du contrôleur OpenDaylight.

Utiliser les informations d'identification par défaut suivantes pour vous connecter :

- Nom d'utilisateur : admin
- Mot de passe : admin

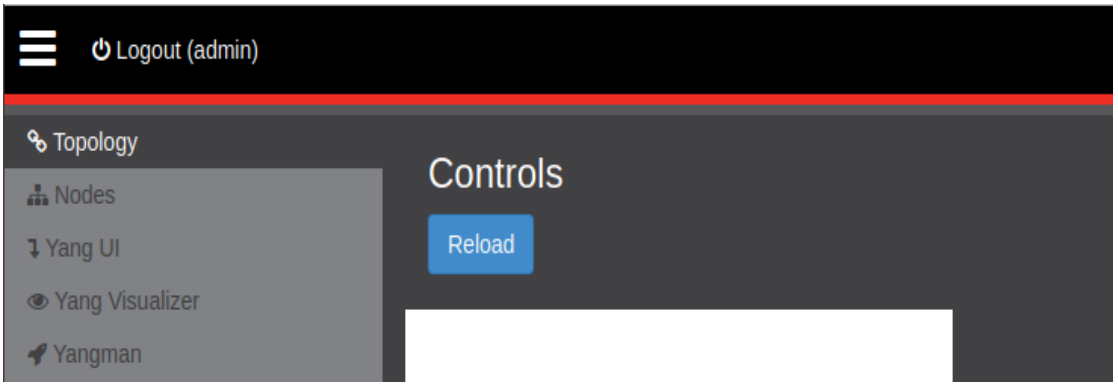

Figure 71 : Connexion à l'interface web du contrôleur Opendaylight.

RYU :

Pour créer manuellement l'environnement Ryu-Book :

Installer les dépendances nécessaires en exécutant la commande suivante dans le terminal :

- sudo apt-get install git python-dev python-setuptools python-pip

Cloner le dépôt Ryu en exécutant la commande suivante :

- git clone https://github.com/osrg/ryu.git

Accéder au répertoire Ryu :

- cd ryu

Installer Ryu et ses dépendances en exécutant la commande suivante :

- sudo pip install .

Après avoir installé Ryu, lancer le contrôleur en utilisant la commande suivante :

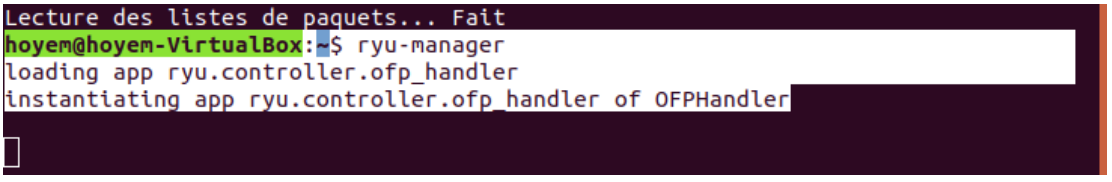

Figure 71 : RYU en cours d'exécution.

Une fois le contrôleur lancé, il sera prêt à gérer les commutateurs SDN et à exécuter les fonctionnalités et les applications correspondantes.

Annexe D

Présentation de l'organisme d'accueil :

La nouvelle structure organisationnelle de SONATRACH est conçue pour répondre aux changements internes et externes qui nécessitent une adaptation du schéma d'organisation et du mode de gestion de l'entreprise. Ces changements visent à relever les défis mentionnés dans le plan à moyen terme de SONATRACH, tels que l'augmentation de la production et des réserves dans le secteur amont, ainsi que la réalisation de projets de raffinage et de pétrochimie dans le secteur aval.

L'adaptation de l'organisation et la modernisation du mode de gestion sont des priorités pour la Direction Générale de SONATRACH. Cela vise à répondre à la demande croissante du marché national et à soutenir la position de SONATRACH sur les marchés pétroliers et gaziers.

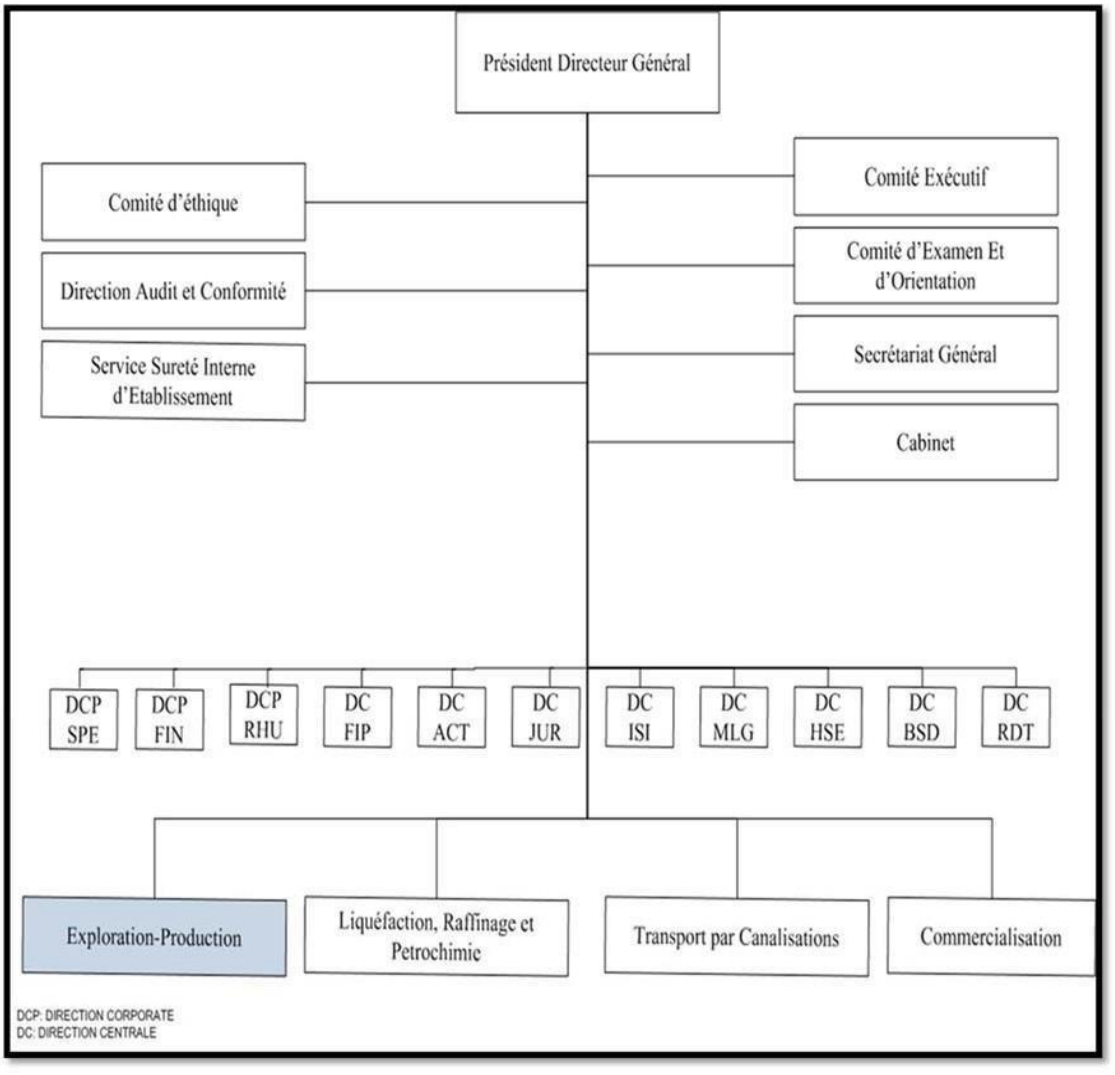

Figure 72 : Organigramme de la Sonatrach.

Présentation de la division production

La division production (SH/DP) de SONATRACH est structurée en un siège situé à Alger, ainsi que dix (10) régions réparties dans le sud du pays. Son objectif principal est de développer l'exploitation des gisements d'hydrocarbures et d'optimiser la production en mettant en œuvre des méthodes efficaces de récupération et de conservation des réserves.

Au sein de la division production, il y a la responsabilité de gérer l'exploitation de 1925 puits producteurs d'huile, 400 puits producteurs de gaz naturel, 173 puits injecteurs d'eau et 200 puits injecteurs de gaz.

En 1991, la production brute de SONATRACH, incluant tous les produits, a atteint 160,4 millions de tonnes équivalent pétrole (TEP), dont 54,6 TEP de gaz ont été réinjectés dans les gisements. Les exportations d'hydrocarbures comprenaient une part significative de gaz naturel liquéfié (GNL), qui représentait 27,71% du total des exportations. La production de GNL a atteint 31,9 millions de mètres cubes de gaz liquides.

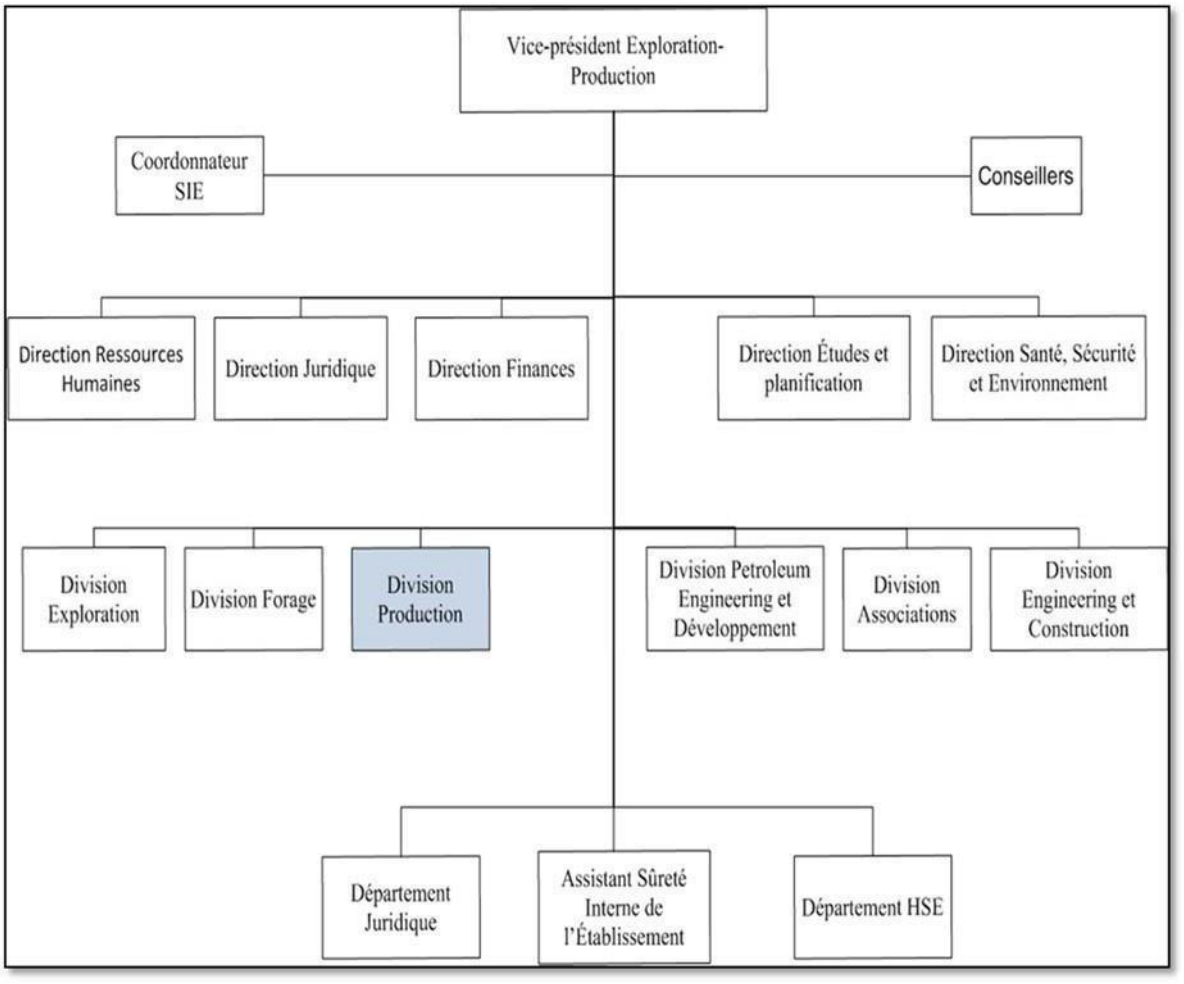

Figure 73: Organigramme de l'activité Amont.

La division production emploi actuellement plus de 16.000 agent répartis entre le siège et lesrégions et structures de : Sept (07) directions :

- Direction Opérations.
- Direction Réalisation.
- Direction Finance et Comptabilité.
- Direction Gestion du Personnel.
- Direction Approvisionnement et Transport.
- Direction Moyens Généraux.
- Direction Informatique

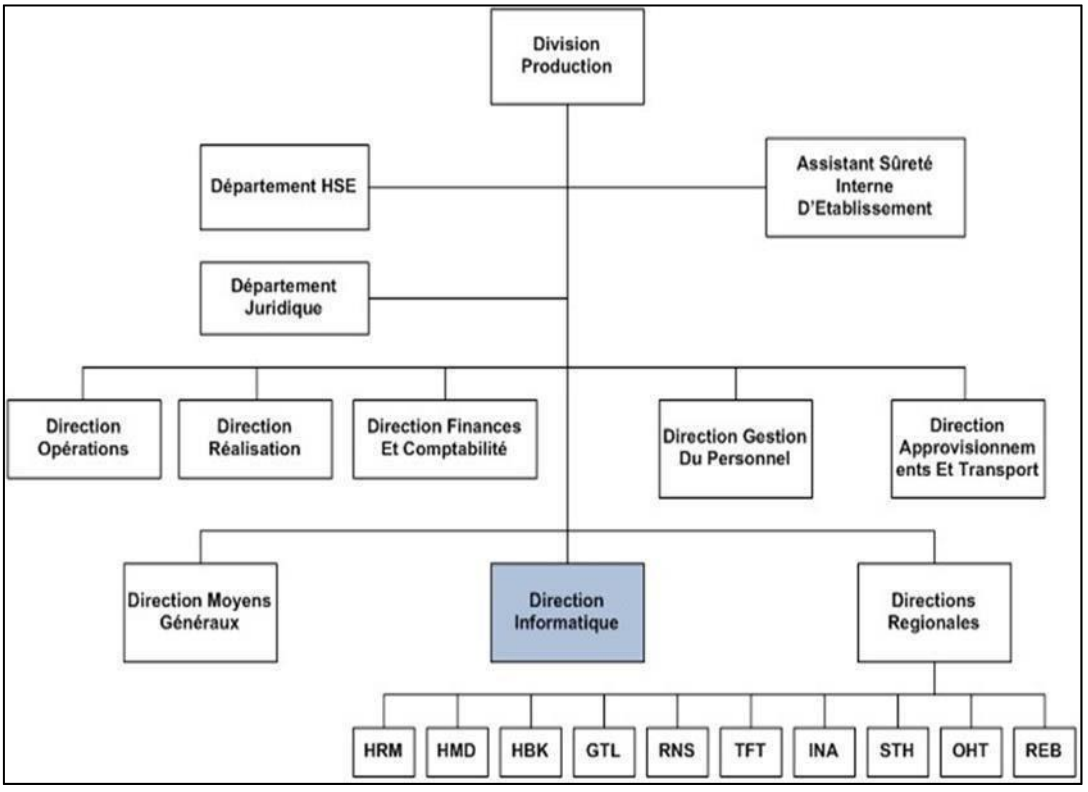

Figure 74 : Organigramme de la Division Production.

Direction Informatique :

La direction informatique de SONATRACH est composée d'une équipe hautement qualifiée, comprenant des post-gradués, des ingénieurs, des licenciés et des techniciens. Elle dispose d'un matériel informatique considérable pour répondre aux besoins de l'entreprise. Sa mission principale est de définir la politique de développement informatique de la division production, en accord avec la stratégie et la politique de SONATRACH. Elle étudie et évalue les besoins en systèmes d'information, élabore et met en œuvre les plans de développement des technologies de l'information, de la communication et de la gestion des connaissances. Elle administre le réseau informatique, standardise les normes d'acquisition d'équipements

Informatiques, et élabore le plan informatique de la division. De plus, elle fournit des services de traitement de l'information aux structures de la division production, et gère une base de données technique, économique et financière.

La direction informatique se compose de quatre départements : maintenance, développement, système et exploitation, ainsi que réseau et télécommunication. Ces départements travaillent en étroite collaboration pour assurer le bon fonctionnement des systèmes informatiques et répondre aux besoins scientifiques et techniques de la division production.

Au cours de notre visite des différents départements de la Direction Informatique, nous avons eu l'opportunité de nous familiariser avec l'environnement de travail, de comprendre l'organisation de la Division Production et de nous intégrer au personnel de la Direction Informatique. Il est important de souligner le positionnement de la Division Production parmi les départements de l'entreprise et les responsabilités qui lui sont attribuées.

Le département réseau et télécommunication joue un rôle clé au sein de la Division Production et se voit assigner diverses missions, notamment :

- Intervention sur les équipements de télécommunication tels que les appareils téléphoniques, les autocommutateurs et les fibres optiques.
- Administration du réseau étendu (WAN) qui relie les différentes activités de SONATRACH.
- Mise en place de nouvelles fonctionnalités telles que les réseaux privés virtuels (VPNs), l'ingénierie de trafic et la gestion de la qualité de service.
- Évaluation de la charge du réseau.
- Protection du réseau local (LAN) et du réseau étendu (WAN).
- Installation et maintenance des réseaux locaux.
- Mise en place et administration de l'inter-réseau reliant les différents sites de la Division Production.
- Intervention sur tous types de câblages réseau et commutation.
- Intégration des micro-ordinateurs des employés au réseau local de l'entreprise.
- Mise en place d'un réseau reliant les différents sites de la Division Production.
- Fourniture d'un accès à Internet à tous les utilisateurs de l'outil informatique.
- Établissement des cahiers des charges pour l'extension et l'amélioration du réseau existant.

Bibliographie :

[1] S. Ben Chahed, « Mise en œuvre des aspects de gestion des réseaux définis par logiciels (réseaux SDN) », masters, École Polytechnique de Montréal, 2015.

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

[3] 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.

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

[5] 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

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

[7] Kim, H., Feamster, N., & Vempala, S. (2013). SDN vs. traditional networking: a performance evaluation. Proceedings of the 9th ACM International on Conference on emerging Networking Experiments and Technologies.

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

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

[10] K. Pentikousis, Ed., "Software-Defined Networking (SDN): Anatomy of a Network Revolution", Wiley, 2015.

[11] I. Choukri, Mohammed Ouzzif, Khalid Bouragba. Software Defined Networking (SDN): Etat de L'art. Colloque sur les Objets et systèmes Connectés, Ecole Supérieure de Technologie de Casablanca (Maroc), Institut Universitaire de Technologie d'AixMarseille (France), Jun 2019.

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

[13] 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).

[14] "OpenFlow Tutorial", http://archive.openflow.org, 2014.

[15] Part of LF Networking (LFN). "opendaylight documentation". site oficiel. [online]. avaliable at : https://www.opendaylight.org, 20

[16] 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.

[17] 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/.

[18] "floodlight documentation officiel. [online] at: https://floodlight.atlassian.net/

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

[20] Hosted by The Linux Foundation. "onos documentation". site ociel. [online]: https://wiki.onosproject.org, December 5, 2014.

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

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

[23] "openmul sdn platform," site ociel. [online]. avaliable at : http://www.openmul. org/openmul-controller.html, 2011. [34] A.Cox Z.Cai and E.T.S.Ng.

[24] "maestro : A system for scalable open-ow control,". [online]. avaliable at : http://www.cs.rice.edu/{~}eugeneng/papers/ TR10-11.pdf, 2011.

[25] 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).

[26] « Introduction to open vswitch », http://www.keepingitclassless.net, Oct. 2013.

[27] Openflowtechnlogy http://h17007.www.hp.com/ch/fr/solutions/technology/openflow.

[28] « Introduction to open vswitch », http://www.keepingitclassless.net, Oct. 2013.

[29] K. Idoudi, « Implémentation d'un plan de contrôle unifié pour les réseaux multicouches IP/DWDM », thèse master, Université de Québec à Montréal, Mai 2014

[30] "Cisco Software-Defined Access: Simplify Access to Services and Applications" de cisco press

[31] "Data Center Networking Demystified: An Introduction to Data Center NetworkingTechnologies, Design, and Best Practices" de Arup Chakrabarti.

[32] "SD-WAN: An Expert Guide to Planning, Designing, and Deploying a Software DefinedWAN" de Dave Greenfield et Michael Wood.

[33] Foundation, O.N.: Openflow switch specification version 1.3.1. Tech. rep., Open Networking Foundation (September 2012).

[34] Mamushiane, L., Lysko, A., & Dlamini, S. (2018, April). A comparative evaluation ofthe performance of popular SDN controllers. In 2018 Wireless Days (WD) (pp. 54-59). IEEE.

[35] Cas d'utilisation, Visibilité et contrôle : https://www.opendaylight.org/use-cases-and-users/by-function/visibility-and-control

[36] http://mininet.org, last visited at January 15, 2022.

[37] Qualité de production, commutateur virtuel ouvert multicouche https://www.openvswitch.org/

[38] Open vSwitch: http://www.linusnova.com/2013/04/open-vswitch

[39] What is Open vSwitch: https://northboundnetworks.com/blogs/sdn/what-is-open-vswitch

[40] Sakellaropoulou, D. «A Qualitative Study of SDN Controllers», thèse de doctorat de l'université de Athens, Septembre 2017.

[41] Jaber Jawad Al-Asfoor, M. (2017). A Simulation of a Networked Video Monitoring System Using NS2. Journal of Al-Qadisiyah for Computer Science and Mathematics, 9(1), 117-131

[42] Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016, April). SDN controllers: A comparative study. In 2016 18th mediterranean

electrotechnical conference (MELECON) (pp. 1-6). IEEE.

[43] https://github.com/aorogat/CBench, last visited at January 29, 2022.

[44] SDN Controllers: Benchmarking & Performance Evaluation (12 /02/2019), IEEE

[45] Introduction to SDN (Software Defined Networking) : https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-sdn-software-defined-networking

[46] Implemanting Software-define-network (SDN) based firewall : https://www.opensourceforu.com/2016/07/implementing-a-software-defined-network-sdn-based-firewall/

[47] Zerkane, S. (2018). Security Analysis and Access Control Enforcement through Software Defined Networks [Thèse de doctorat, BCOM, site de Rennes]. Lab-STICC – CNRS UMR 6285.

[48] Dixit, V. H., Kyung, S., Zhao, Z., Doupé, A., Shoshitaishvili, Y., & Ahn, G. (Year). Challenges and Preparedness of SDN-based Firewalls. Arizona State University, Tempe, AZ, USA

[49] Thuy Vinh Tran, Heejune Ahn A Network Topologyaware Selectively Distributed Firewall Control in SDN Department of Electrical and Information Engineering Seoul National University of Science and Technology Seoul, République de Corée

[50] Phatak, A., Kadikar, R.U., Amutha, B. (2018). "Performance Analysis of Firewall Based on SDN and OpenFlow." In Proceedings of the 2018 International Conference on Communication and Signal Processing (ICCSP), Computer Science.

[51] Yash Bajajet al(2018) 'Pare-feu basé sur le concept de Sdn',Journal international de la recherche avancée actuelle,07(2), p.

[52] Avid, Tariq, Tehseen Riaz et Asad Rasheed, "Un pare-feu de couche 2 pour un réseau défini par logiciel", dansInformation Assurance and Cyber Security (CIACS), Conférence 2014 sur, p. 39-42. IEEE, 2014.

[53] Kaur, Karamjeet, Krishan Kumar, Japinder Singh et Navtej Singh Ghumman, "Pare-feu programmable utilisant un réseau défini par logiciel", dansL'informatique pour le développement mondial durable.

[54] Pena, Justin Gregory V. et William Emmanuel Yu, "Développement d'un pare-feu distribué à l'aide de la technologie de mise en réseau définie par logiciel", 4e Conférence internationale de l'IEEE sur les sciences et technologies de l'information, p. 449-452. IEEE.

[55] Wang, al. "SDN-Based Firewall Using Machine Learning for Real-Time Network Attack Detection and Prevention."p 25-34

[56] Open Network Foundation, « openFlow Switch Specification », version 1.4, Octobre 2013.

[57] Protocole_openflow : [https://www.supinfo.com/articles/single/1010-protocole-openflow]

[58] 'Open Networking Foundation (ONF) : https://www.opennetworking.org/.

[59] Ferreira, T. & Cloarec, V « le projet OpenFlow », Telecome Lille,2010.