الجمهورية الجـزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
-- تلمسان --جامعة أبي بـكر بلـقايـد
Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Télécommunications

Spécialité : Réseaux et Télécommunications

Par Ouanani Aymen Riad – Zaid Seyyid Ahmed

Sujet

Conception d'un routeur SDN Open-Flow pour la gestion intelligente des flux

Soutenu publiquement, le 22/06/2026, devant le jury composé de :

F.T. BENDIMERAD	Professeur Emirte	Université de Tlemcen	Président
S.M MERIAH	Professeur	Université de Tlemcen	Examinateur
F. DERRAZ	MCA	Université de Tlemcen	Encadrant
M. BOUSAHLA	MCB	Université de Tlemcen	Co-encadrant

Année universitaire : 2025–2026

Remerciements

Louange à Allah, le Tout-Puissant, qui nous a accordé la santé, la force et la persévérance nécessaires pour mener à bien nos études et accomplir ce travail. Que la paix et la bénédiction soient sur notre Prophète Mohammed (paix et salut sur lui).

Nous tenons à exprimer notre profonde gratitude à notre encadrant, **M. F. DERRAZ**, pour avoir accepté de diriger ce travail. Nous le remercions sincèrement pour sa disponibilité, ses précieux conseils, son accompagnement constant et ses orientations judicieuses qui ont grandement contribué à la réalisation de ce projet de fin d'études.

Nos remerciements les plus sincères s'adressent également à **M. M. BOUSAHLA,** notre co-encadreur, pour son soutien, sa disponibilité et ses remarques constructives. Ses conseils avisés et son aide précieuse ont été d'un grand apport tout au long de nos recherches.

Nous exprimons également notre reconnaissance à Mr F. T BENDIMERAD, président du jury, ainsi qu'a Mr S.M MERIAH examinateur pour l'honneur qu'ils nous ont fait en évaluant ce travail. Nous les remercions pour l'intérêt porté à notre projet ainsi que pour leurs remarques et recommandations enrichissantes.

Nous adressons enfin nos plus chaleureux remerciements à nos parents et à nos familles pour leur amour, leur soutien indéfectible, leurs encouragements constants et la confiance qu'ils nous ont toujours témoignée. Leur présence à nos côtés a été une source de motivation et de force tout au long de notre parcours.

À toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce travail, nous adressons nos sincères remerciements.

Merci infiniment à tous pour votre soutien et votre confiance.

Abstract

This thesis presents the design, deployment, and evaluation of a Software Defined Networking (SDN) architecture based on the OpenFlow protocol for the centralized management of network flows. The main objective is to study the contribution of SDN to improving the flexibility, supervision, and administration of network infrastructures compared with traditional architectures.

To achieve this objective, an experimental platform was implemented using Open vSwitch (OVS) switches and a POX controller hosted on the Intel Galileo platform. The different configuration steps made it possible to establish OpenFlow communications between the network devices and to implement a dynamic network flow management mechanism.

The architecture was validated through several experiments, including connectivity tests, analysis of OpenFlow Flow Tables, network traffic capture using Wireshark, and performance evaluation with iperf3. The obtained results confirmed the proper operation of the proposed architecture as well as the effectiveness of the controller in the centralized management of network flows.

The work carried out highlighted the advantages of SDN in terms of programmability, flexibility, global network visibility, and simplified administration. These results demonstrate the relevance of this approach for the development of next-generation networks.

Keywords : SDN, OpenFlow, POX, Open vSwitch, Intel Galileo, network flow management, SDN controller.

Resumé

Ce mémoire présente la conception, le déploiement et l'évaluation d'une architecture Software Defined Networking (SDN) basée sur le protocole OpenFlow pour la gestion centralisée des flux réseau. L'objectif principal est d'étudier l'apport du SDN dans l'amélioration de la flexibilité, de la supervision et de l'administration des infrastructures réseau par rapport aux architectures traditionnelles.

Pour atteindre cet objectif, une plateforme expérimentale a été mise en œuvre à l'aide de switches Open vSwitch (OVS) et d'un contrôleur POX hébergé sur la plateforme Intel Galileo. Les différentes étapes de configuration ont permis d'établir les communications OpenFlow entre les équipements et de mettre en œuvre un mécanisme de gestion dynamique des flux réseau.

La validation de l'architecture a été réalisée à travers plusieurs expérimentations, notamment des tests de connectivité, l'analyse des Flow Tables OpenFlow, la capture du trafic réseau à l'aide de Wireshark et l'évaluation des performances avec iperf3. Les résultats obtenus ont confirmé le bon fonctionnement de l'architecture proposée ainsi que l'efficacité du contrôleur dans la gestion centralisée des flux.

Les travaux réalisés ont permis de mettre en évidence les avantages du SDN en matière de programmabilité, de flexibilité, de visibilité globale du réseau et de simplification de l'administration. Ces résultats démontrent l'intérêt de cette approche pour le développement des réseaux de nouvelle génération.

Mots-clés : SDN, OpenFlow, POX, Open vSwitch, Intel Galileo, gestion des flux réseau, contrôleur SDN.

الملخص

تتناول هذه المذكرة تصميم وتنفيذ وتقييم بنية شبكية قائمة على مفهوم الشبكات المعرّفة برمجياً، وذلك بالاعتماد على بروتوكول التدفق المفتوح بهدف تحقيق إدارة مركزية وفعّالة لتدفقات الشبكة. ويتمثل الهدف الرئيسي من هذا العمل في دراسة مساهمة الشبكات المعرّفة برمجياً في تحسين مرونة الشبكة، وتسهيل إدارتها، وتعزيز عملية مراقبة حركة البيانات مقارنة بالشبكات التقليدية.

ولتحقيق هذا الهدف، تم إنشاء منصة تجريبية بالاعتماد على مبدلات افتراضية مفتوحة ومتحكم بوكس مستضاف على منصة إنتل غاليليو. وقد سمحت عمليات الإعداد والتهيئة بإنشاء اتصالات وفق بروتوكول التدفق المفتوح بين مختلف مكونات الشبكة، وتطبيق آلية ديناميكية لإدارة تدفقات البيانات.

تم التحقق من فعالية البنية المقترحة من خلال مجموعة من الاختبارات، شملت التحقق من الاتصال بين الأجهزة، وتحليل جداول التدفق، ومراقبة حركة المرور الشبكية باستخدام أداة وايرشارك، بالإضافة إلى تقييم الأداء باستعمال أداة آي بيرف ٣. وأظهرت النتائج المتحصل عليها نجاح البنية المقترحة في تحقيق إدارة مركزية لحركة المرور الشبكية وضمان استقرار الاتصالات بين مختلف المكونات.

أبرزت هذه الدراسة المزايا التي توفرها الشبكات المعرّفة برمجياً من حيث القابلية للبرمجة، والمرونة، والرؤية الشاملة للشبكة، وتبسيط عمليات الإدارة. وتؤكد النتائج أهمية هذه المقاربة في تطوير شبكات الجيل الجديد.

الكلمات المفتاحية : الشبكات المعرّفة برمجياً، بروتوكول التدفق المفتوح، متحكم بوكس، المبدلات الافتراضية المفتوحة، منصة إنتل غاليليو، إدارة تدفقات الشبكة، المتحكم الشبكي. ‘‘‘

Table des matières

Remerciements **i**

Abstract **ii**

Résume **iii**

الملخص **iv**

Liste des figures **ix**

Liste des tableaux **x**

Liste des abréviations **xi**

Introduction générale **1**

1 Concepts fondamentaux des réseaux IP **3**

1.1 Introduction 3

1.2 Fondements des réseaux IP et routage 3

1.2.1 Architecture des réseaux IP 4

1.2.2 Fonctionnement du routage 4

1.2.3 Protocoles de routage 5

1.2.4 Limites des mécanismes de routage 6

1.3 Qualité de service et gestion des flux réseau 6

1.3.1 Notion de QoS 6

1.3.2 Paramètres de performance 6

1.3.3 Techniques de gestion du trafic 7

1.3.4 Définition des flux réseau 7

1.3.5 Classification des flux 7

1.3.6 Importance de la gestion des flux 7
1.4 Limites des réseaux traditionnels 8
1.4.1 Manque de flexibilité 8
1.4.2 Complexité de gestion 8
1.4.3 Absence de vision globale 8
1.4.4 Difficulté d'optimisation 9
1.5 Transition vers le Software Defined Networking 9
1.5.1 Insuffisances des mécanismes réseau traditionnels 9
1.5.2 Évolution vers un modèle de contrôle global du réseau 9
1.5.3 Principe du Software Defined Networking (SDN) 9
1.6 Conclusion 10

2 Software Defined Networking et technologies associées 11
2.1 Introduction 11
2.2 Concept du Software Defined Networking 12
2.2.1 Définition du SDN 12
2.2.2 Principes fondamentaux 12
2.2.3 Objectifs du SDN 12
2.3 Architecture SDN 13
2.3.1 Plan de données 13
2.3.2 Plan de contrôle 13
2.3.3 Plan application 14
2.4 Protocole OpenFlow 14
2.4.1 Principe de fonctionnement 15
2.4.2 Structure des flow tables 15
2.4.3 Messages OpenFlow 15
2.5 Contrôleurs SDN 16
2.5.1 Rôle du contrôleur 16
2.5.2 Présentation de POX 17
2.5.3 Comparaison avec autres contrôleurs 17
2.6 Avantages et limites du SDN 18
2.6.1 Avantages du SDN 18
2.6.2 Limites du SDN 19
2.7 Conclusion 19

3 Conception, implémentation et validation de l'architecture SDN OpenFlow 20
3.1 Introduction 20
3.2 Présentation de la solution SDN et de l'environnement expérimental 21
3.2.1 Présentation générale de la solution 21
3.2.2 Infrastructure matérielle utilisée 21
3.2.2.1 Ordinateur hôte 21
3.2.2.2 Plateforme Intel Galileo 21
3.2.2.3 Machines virtuelles 22
3.2.3 Environnement logiciel et technologies utilisées 23
3.2.3.1 VMware Workstation 23
3.2.3.2 Ubuntu Linux 24
3.2.3.3 Open vSwitch (OVS) 24
3.2.3.4 Contrôleur POX 25
3.2.3.5 OpenFlow 25
3.2.3.6 Wireshark 25
3.2.3.7 iperf3 26
3.3 Conception et déploiement de l'architecture SDN 26
3.3.1 Architecture et topologie du réseau SDN 26
3.3.2 Configuration de l'infrastructure OpenFlow 27
3.3.2.1 Création des switches Open vSwitch 27
3.3.2.2 Configuration des interfaces réseau 28
3.3.2.3 Plan d'adressage IP et attribution des adresses réseau . . 28
3.3.3 Mise en œuvre du contrôleur SDN POX 29
3.3.3.1 Installation, configuration et lancement du contrôleur POX 29
3.3.3.2 Connexion des switches OpenFlow au contrôleur 29
3.4 Validation et tests de l'architecture SDN 30
3.4.1 Vérification de la connectivité réseau 30
3.4.2 Analyse des Flow Tables OpenFlow 31
3.4.3 Analyse du trafic réseau avec Wireshark 32
3.4.4 Évaluation des performances réseau avec iperf3 33
3.5 Analyse et interprétation des résultats 34
3.5.1 Analyse des performances obtenues 34
3.5.2 Apport du SDN dans la gestion des flux réseau 35

3.5.2.1 Centralisation du contrôle 35
3.5.2.2 Gestion dynamique des flux 36
3.5.2.3 Visibilité globale du réseau 36
3.5.2.4 Simplification de l'administration 36
3.5.2.5 Flexibilité et évolutivité 36
3.5.3 Comparaison avec les réseaux traditionnels 36
3.5.4 Perspectives d'amélioration 38
3.6 Conclusion 38

Conclusion générale **40**

Annexe **42**

Bibliographie **49**

Liste des figures

Figure I.1	Architecture générale d’un réseau IP
Figure I.2	Fondements des réseaux IP et du routage
Figure II.1	Architecture générale d’un réseau SDN basée sur OpenFlow
Figure II.2	Cycle de traitement d’un flux OpenFlow
Figure II.3	Architecture interne du contrôleur POX
Figure III.1	Plateforme Intel Galileo
Figure III.2	Machines virtuelles utilisées dans l’architecture SDN
Figure III.3	Interfaces du logiciel VMware
Figure III.4	Version d’Open vSwitch (OVS 3.3.4)
Figure III.5	Lancement du contrôleur SDN POX
Figure III.6	Interface graphique du logiciel Wireshark
Figure III.7	Topologie générale de l’architecture SDN OpenFlow déployée
Figure III.8	Configuration des interfaces réseau des switches OpenFlow
Figure III.9	Plan d’adressage IP
Figure III.10	Établissement des connexions OpenFlow entre les switches OVS et le contrôleur POX
Figure III.11	Vérification de la connectivité réseau dans l’architecture SDN
Figure III.12	Installation dynamique des règles OpenFlow par le contrôleur POX
Figure III.13	Analyse des règles OpenFlow dans les Flow Tables du switch Open vSwitch
Figure III.14	Analyse des échanges OpenFlow avec Wireshark
Figure III.15	Évaluation des performances réseau avec iperf3

Liste des tableaux

Tableau II.1 Comparaison entre les principaux contrôleurs SDN
Tableau III.1 Comparaison avec les réseaux traditionnels

Liste des abréviations

API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CapEx	Capital Expenditure
CLI	Command Line Interface
CoS	Class of Service
DPI	Deep Packet Inspection
DSCP	Differentiated Services Code Point
EGP	Exterior Gateway Protocol
ICMP	Internet Control Message Protocol
IGP	Interior Gateway Protocol
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
MAC	Media Access Control
NBI	Northbound Interface
ODL	OpenDaylight
OF	OpenFlow
ONOS	Open Network Operating System
OS	Operating System
OSPF	Open Shortest Path First
OVS	Open vSwitch
PFE	Projet de Fin d’Études
POX	Python-based OpenFlow Controller
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
RIP	Routing Information Protocol
Ryu	Contrôleur SDN développé en Python

SBI	Southbound Interface
SDN	Software Defined Networking
SPF	Shortest Path First
SPoF	Single Point of Failure
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
vSwitch	Virtual Switch
WAN	Wide Area Network
WFQ	Weighted Fair Queuing

Introduction générale

Les réseaux informatiques occupent une place essentielle dans les systèmes de communication modernes. Ils assurent l'échange de données entre utilisateurs, serveurs, objets connectés, applications industrielles et plateformes numériques. L'augmentation du trafic, la diversification des services et la généralisation de la virtualisation imposent aux infrastructures réseau des exigences de plus en plus fortes en matière de disponibilité, de sécurité, de qualité de service et de flexibilité.

Dans les architectures classiques, les fonctions de contrôle et de transfert sont généralement intégrées dans chaque équipement réseau. Cette organisation distribuée a permis le développement d'Internet et des réseaux IP actuels, mais elle présente plusieurs limites lorsque l'on souhaite administrer dynamiquement des environnements complexes. La configuration reste souvent manuelle, la vision de l'état global du réseau est limitée et l'optimisation des flux dépend de décisions locales prises par les équipements. Ces contraintes rendent difficile l'adaptation rapide du réseau aux variations du trafic et aux besoins des applications.

Le paradigme *Software Defined Networking* (SDN) propose une réponse à ces limitations en séparant le plan de contrôle du plan de données. Le contrôle du réseau est alors logiquement centralisé au niveau d'un contrôleur logiciel, tandis que les équipements de commutation assurent principalement le transfert des paquets selon les règles installées par ce contrôleur. Cette approche rend le réseau programmable, plus flexible et plus simple à superviser. Le protocole OpenFlow constitue l'un des mécanismes les plus utilisés pour assurer l'échange d'informations entre le contrôleur SDN et les commutateurs compatibles.

Le présent mémoire s'inscrit dans ce contexte et porte sur la conception d'un routeur SDN OpenFlow destiné à la gestion intelligente des flux réseau. La problématique étudiée peut être formulée comme suit : comment exploiter une architecture SDN basée sur OpenFlow afin de centraliser la décision réseau, d'installer dynamiquement des règles de transfert et d'améliorer la gestion des flux par rapport à une architecture traditionnelle ?

L'objectif principal de ce travail est de concevoir, déployer et valider une plateforme expérimentale SDN reposant sur Open vSwitch, le contrôleur POX et une plateforme Intel Galileo. L'étude vise à montrer comment les communications OpenFlow sont établies, comment les règles de flux sont générées et installées dans les tables OpenFlow, et comment les performances peuvent être observées à l'aide d'outils tels que Wireshark et iperf3.

Ce mémoire est organisé en trois chapitres. Le premier chapitre présente les concepts fondamentaux des réseaux IP, le routage, la qualité de service et les limites des réseaux traditionnels. Le deuxième chapitre est consacré au Software Defined Networking, au protocole OpenFlow et aux contrôleurs SDN, avec un intérêt particulier pour POX. Le troisième chapitre décrit la conception, l'implémentation et la validation de l'architecture SDN proposée, puis analyse les résultats expérimentaux obtenus. Enfin, une conclusion générale synthétise les apports du travail et propose des perspectives d'amélioration.

Chapitre 1

Concepts fondamentaux des réseaux IP

1.1 Introduction

Les réseaux informatiques constituent un élément fondamental des systèmes d'information modernes en assurant l'échange de données entre les différents équipements connectés. Avec l'évolution des technologies numériques telles que le cloud computing, la virtualisation et l'Internet des objets, les infrastructures réseau sont confrontées à une augmentation importante du trafic ainsi qu'à des besoins croissants en performance et en flexibilité.

Les architectures réseau traditionnelles présentent cependant plusieurs limitations liées à la complexité de gestion, au manque de programmabilité et à la difficulté d'optimisation dynamique du trafic réseau. Face à ces contraintes, de nouvelles approches ont émergé, notamment le Software Defined Networking (SDN), qui introduit une gestion centralisée et programmable des infrastructures réseau.

Avant d'aborder les concepts du SDN, il est essentiel de comprendre les principes fondamentaux des réseaux IP, du routage ainsi que les mécanismes de gestion des flux réseau. Ce chapitre présente les notions de base relatives aux architectures IP et met en évidence les principales limites des réseaux traditionnels ayant conduit à l'émergence du paradigme SDN.

1.2 Fondements des réseaux IP et routage

Dans les réseaux modernes, l'architecture IP constitue la base de la communication entre les différents équipements connectés. Elle repose sur l'utilisation du protocole Internet (IP), qui permet l'identification unique de chaque équipement à travers une adresse IP.

L'architecture des réseaux IP est conçue de manière à assurer le transport des paquets de données entre une source et une destination, indépendamment du chemin emprunté. Cette approche repose sur un modèle distribué où chaque équipement réseau, notamment les routeurs, participe au processus d'acheminement des données.

Les réseaux IP sont caractérisés par leur capacité à interconnecter différents types de réseaux, ce qui permet la création d'infrastructures étendues comme Internet. Toutefois, cette architecture présente certaines limites, notamment en matière de contrôle et d'optimisation du trafic [1] .

1.2.1 Architecture des réseaux IP

L'architecture IP constitue le fondement des communications dans les réseaux modernes. Elle repose sur l'utilisation du protocole Internet (IP), qui permet d'identifier de manière unique chaque équipement connecté à l'aide d'une adresse IP.

Dans ce modèle, les données sont transmises sous forme de paquets indépendants, acheminés de la source vers la destination à travers différents équipements réseau. Cette architecture distribuée permet une grande flexibilité dans la transmission des données, mais elle limite également la capacité de contrôle global du réseau.

Grâce à cette organisation, les réseaux IP peuvent interconnecter des infrastructures hétérogènes à grande échelle, comme Internet. Toutefois, cette approche présente certaines contraintes en matière d'optimisation du trafic et de gestion des ressources réseau [1]. L'architecture générale d'un réseau IPest présentée dans la Figure 1.1

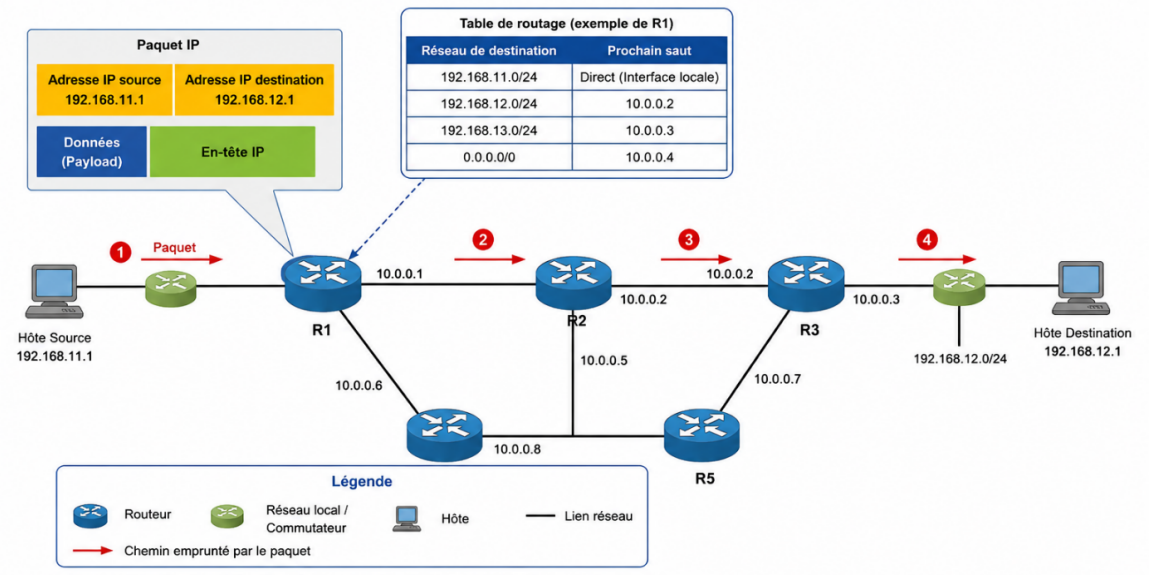


Fig. 1.1 : Architecture générale d'un réseau IP

1.2.2 Fonctionnement du routage

Le routage est un processus clé qui permet de déterminer le chemin suivi par les paquets dans un réseau. Chaque routeur est équipé d'une table de routage contenant les informations nécessaires pour décider du prochain saut.

Lorsqu'un paquet arrive à un routeur, celui-ci analyse l'adresse de destination et sélectionne le chemin approprié en fonction des entrées disponibles dans sa table. Ce mécanisme est appliqué de manière successive jusqu'à atteindre la destination finale.

Le routage peut être réalisé de manière statique ou dynamique. Dans le cas du routage dynamique, les routeurs échangent des informations afin d'adapter leurs décisions en fonction de l'état du réseau. Toutefois, ces décisions restent locales et ne permettent pas une gestion globale optimisée.

1.2.3 Protocoles de routage

Les protocoles de routage assurent la diffusion des informations nécessaires à la construction des tables de routage. Ils jouent un rôle fondamental dans le fonctionnement des réseaux de grande taille.

On distingue deux grandes catégories (Figure 1.2) :

- Les protocoles internes (IGP), tels que RIP et OSPF, utilisés à l'intérieur d'un réseau
- Les protocoles externes (EGP), comme BGP, utilisés pour l'interconnexion entre réseaux

Ces protocoles utilisent différents critères pour choisir le meilleur chemin, notamment la distance ou le coût. Cependant, leur fonctionnement reste basé sur une logique distribuée, ce qui limite leur capacité à réagir rapidement aux changements du réseau [1].

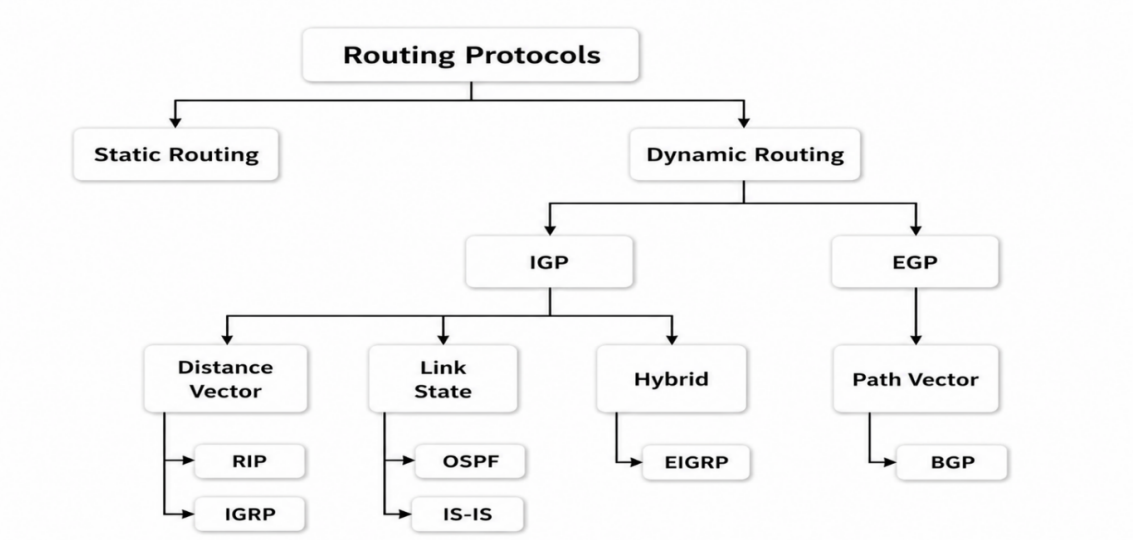


Fig. 1.2 : Fondements des réseaux IP et du routage

1.2.4 Limites des mécanismes de routage

L'architecture des réseaux traditionnels repose sur un plan de contrôle distribué où chaque nœud prend des décisions de routage de manière autonome et locale. Cette absence de vision globale limite considérablement la capacité du réseau à optimiser les flux de bout en bout [2]. En outre, la convergence des protocoles de routage classiques lors de changements topologiques est souvent lente, induisant des interruptions de service et une dégradation de la Qualité de Service (QoS). Enfin, la gestion rigide des ressources empêche une réaction dynamique face aux phénomènes de congestion. Ces lacunes structurelles justifient la transition vers le paradigme **SDN (Software-Defined Networking)**, qui prône la dissociation des plans de contrôle et de données pour une gestion centralisée et programmable [3]

1.3 Qualité de service et gestion des flux réseau

L'évolution massive des usages numériques et l'émergence d'applications gourmandes en ressources imposent aux infrastructures de transport une gestion bien plus fine que le simple acheminement au « meilleur effort » (*Best Effort*). Dans ce contexte, la compréhension des mécanismes de Qualité de Service (QoS) devient une nécessité fondamentale pour assurer la performance et la fiabilité des services critiques au sein des réseaux modernes.

1.3.1 Notion de QoS

La Qualité de Service (QoS) se définit comme la capacité d'une architecture réseau à proposer un traitement différencié et hiérarchisé selon les catégories de trafic [4]. Elle a pour objectif principal de garantir que les flux soumis à des contraintes strictes — notamment en termes de temps réel (latence, gigue) ou de haute disponibilité — bénéficient des ressources prioritaires nécessaires. Cette gestion permet de maintenir une expérience utilisateur (QoE) optimale, même en période de forte charge ou de congestion des liens [5]

1.3.2 Paramètres de performance

L'évaluation technique de la qualité d'un réseau s'appuie sur des indicateurs métriques précis qui traduisent son état de performance opérationnelle. La **bande passante** représente le débit binaire maximal effectif, une ressource critique souvent mise à l'épreuve par l'augmentation constante du volume de trafic [5]. Le **délai**, ou latence, mesure le temps de transit total d'un paquet de la source à la destination (One-Way Delay). La **gigue** (jitter) correspond à la fluctuation statistique de ce délai, impactant directement la fluidité des flux multimédias en temps réel [6]. Enfin, le **taux de perte de paquets** comptabilise les unités de données rejetées, un

phénomène généralement causé par la saturation des files d'attente (buffers) lors de congestions réseau [7].

1.3.3 Techniques de gestion du trafic

Pour réguler efficacement le transit des données, les équipements de routage déploient divers mécanismes de contrôle. Le **lissage (shaping)** permet de temporiser les paquets pour lisser les pics de trafic et maintenir un débit régulier, tandis que le **policeur (policing)** impose une limitation stricte en rejetant immédiatement tout excédent [7]. En complément, l'**ordonnancement (scheduling)** organise la priorité de sortie des paquets stockés en mémoire (files d'attente) afin de privilégier les flux les plus sensibles aux délais, en utilisant des algorithmes tels que le *Weighted Fair Queuing* (WFQ) [8].

1.3.4 Définition des flux réseau

Un flux réseau est défini comme une séquence de paquets partageant des caractéristiques d'identification identiques au cours d'une même session de communication. Dans les architectures IP standards, ce flux est caractérisé par son **quintuplet (5-tuple)**, qui regroupe les adresses IP source et destination, les numéros de ports correspondants ainsi que le protocole de transport utilisé [9]. Cette notion de flux est centrale, car elle constitue l'unité de base de manipulation dans les tables de flux (Flow Tables) du SDN.

1.3.5 Classification des flux

La classification constitue l'étape préliminaire permettant d'isoler les paquets selon leur nature applicative. Cette distinction peut s'opérer au niveau du transport, en utilisant les ports (TCP/UDP) pour identifier les services, ou au niveau applicatif (Deep Packet Inspection) pour une reconnaissance plus fine des besoins réels du trafic [5]. Une fois classé, le **marquage** (via les champs DSCP ou CoS) permet d'attribuer une étiquette de priorité au paquet, facilitant ainsi son traitement spécifique par l'ensemble des nœuds du réseau.

1.3.6 Importance de la gestion des flux

Une administration rigoureuse et granulaire des flux est aujourd'hui essentielle pour pallier l'absence de vision globale propre aux réseaux traditionnels. Elle permet non seulement d'optimiser l'utilisation des ressources existantes et de garantir le maintien des services pour les applications modernes, mais elle prépare également l'infrastructure à une gestion plus agile et dynamique. Cette évolution vers une maîtrise logicielle des flux constitue le fondement du passage vers le modèle **SDN**, où le contrôle n'est plus lié à l'équipement physique

mais à une logique programmatique centralisée [3].

1.4 Limites des réseaux traditionnels

L'architecture des réseaux classiques, bien qu'éprouvée, se heurte aujourd'hui à des défis structurels majeurs face à l'explosion du trafic et à la diversité des nouveaux services numériques. Cette section détaille les contraintes intrinsèques qui freinent l'évolution des infrastructures traditionnelles.

1.4.1 Manque de flexibilité

La principale limite des réseaux traditionnels réside dans le **couplage étroit** entre le plan de contrôle (intelligence) et le plan de données (transfert) au sein de chaque équipement. Cette intégration verticale rend le réseau rigide : toute modification d'une politique de routage nécessite une reconfiguration manuelle de chaque nœud, souvent propriétaire. Cette absence de programmabilité et le phénomène de **dépendance vis-à-vis des constructeurs (Vendor Lock-in)** empêchent le réseau de s'adapter en temps réel aux besoins changeants des applications modernes [3].

1.4.2 Complexité de gestion

Dans un modèle distribué, la gestion du réseau devient exponentiellement complexe à mesure que l'infrastructure s'étend. Les administrateurs doivent manipuler des interfaces de ligne de commande (CLI) disparates, ce qui augmente considérablement le risque d'erreurs de configuration humaine. Cette complexité opérationnelle, caractérisée par une gestion **"hop-by-hop"**, ralentit le déploiement de nouveaux services et alourdit la maintenance globale du système [10].

1.4.3 Absence de vision globale

Les décisions d'acheminement dans les réseaux classiques sont prises de manière autonome par chaque routeur en se basant sur une vue partielle de la topologie. Cette vision fragmentée empêche une orchestration efficace à l'échelle de l'infrastructure complète. Sans un **plan de contrôle centralisé** capable d'appréhender l'état global des liens (charge, latence, pannes), le réseau ne peut pas réagir de manière cohérente et optimale aux événements critiques [2].

1.4.4 Difficulté d'optimisation

Les mécanismes de routage traditionnels, s'appuyant majoritairement sur des algorithmes de type **Shortest Path First (SPF)**, ne permettent pas une exploitation optimale des ressources. Cette logique conduit fréquemment à la saturation de certains liens alors que d'autres restent sous-utilisés, créant des goulots d'étranglement. L'absence d'une intelligence centralisée rend l'ingénierie de trafic rigide et peu réactive face aux phénomènes de congestion dynamique [11].

1.5 Transition vers le Software Defined Networking

L'analyse des infrastructures classiques révèle un décalage croissant entre les capacités des équipements matériels et les exigences de flexibilité des services numériques actuels. Cette section marque le passage d'une gestion décentralisée vers un nouveau paradigme de contrôle logiciel.

1.5.1 Insuffisances des mécanismes réseau traditionnels

Les limites précédemment évoquées, telles que la rigidité du plan de contrôle et la complexité des configurations manuelles, constituent les principales insuffisances des réseaux traditionnels. L'impossibilité d'adapter dynamiquement les chemins de routage en fonction de l'état réel des flux entraîne une sous-optimisation chronique des ressources. Face à des environnements de plus en plus virtualisés, le modèle classique, trop statique, ne permet plus de garantir une Qualité de Service (QoS) constante, rendant nécessaire une rupture technologique [3].

1.5.2 Évolution vers un modèle de contrôle global du réseau

Pour pallier ces manques, l'ingénierie réseau a évolué vers l'abstraction du matériel. L'idée centrale est de s'affranchir d'une vision locale par équipement pour adopter un modèle de contrôle global. En centralisant logiquement l'intelligence du réseau, il devient possible d'appréhender la topologie dans son ensemble. Cette évolution permet une meilleure réactivité face aux pannes et offre une plateforme unique pour orchestrer les politiques de sécurité et de gestion de trafic de manière cohérente sur toute l'infrastructure [10].

1.5.3 Principe du Software Defined Networking (SDN)

Le Software Defined Networking (SDN) concrétise cette évolution en imposant une séparation stricte entre le **plan de contrôle** et le **plan de données**. Le principe fondamental repose

sur la **programmabilité** : le réseau n'est plus une collection de boîtes fermées (”black boxes”), mais une ressource logicielle agile. L'apport majeur du SDN réside dans sa capacité à automatiser la gestion des flux via des protocoles ouverts comme **OpenFlow**. Cette approche transforme le réseau en une infrastructure dynamique, capable de s'auto-ajuster pour optimiser les performances [2].

1.6 Conclusion

Dans ce chapitre, nous avons présenté les principes fondamentaux des réseaux IP ainsi que les mécanismes de routage permettant l’acheminement des paquets à travers les infrastructures réseau. Nous avons également étudié les notions liées à la gestion des flux réseau et leur rôle dans l’optimisation des communications entre les différents équipements connectés.

L’analyse du fonctionnement des réseaux traditionnels a permis de mettre en évidence leur architecture distribuée ainsi que les mécanismes utilisés pour assurer le transfert des données. Toutefois, l’évolution des besoins en connectivité, l’augmentation du trafic réseau et la diversification des services numériques rendent la gestion de ces infrastructures de plus en plus complexe.

Ces constats soulignent la nécessité de disposer de solutions capables d’améliorer la supervision, la flexibilité et l’efficacité des réseaux modernes. Dans cette perspective, de nouvelles architectures ont été proposées afin de répondre à ces exigences et d’offrir une meilleure maîtrise des flux de communication.

Chapitre 2

Software Defined Networking et technologies associées

2.1 Introduction

L'évolution des réseaux informatiques et l'augmentation continue des besoins en connectivité ont mis en évidence certaines limites des architectures réseau traditionnelles. Bien que les réseaux IP aient permis le développement d'Internet et des infrastructures modernes, leur gestion devient de plus en plus complexe face à la croissance du trafic, à la diversité des applications et aux exigences accrues en matière de performance et de qualité de service.

Afin de répondre à ces nouveaux défis, le paradigme Software Defined Networking (SDN) a émergé comme une approche innovante visant à rendre les réseaux plus flexibles, programmables et faciles à administrer. Son principe fondamental repose sur la séparation du plan de contrôle et du plan de données, permettant ainsi de centraliser l'intelligence du réseau au niveau d'un contrôleur capable de superviser et de gérer dynamiquement les flux de communication.

L'objectif de ce chapitre est de présenter les concepts fondamentaux et les technologies qui constituent la base des architectures SDN. Nous aborderons tout d'abord les principes de fonctionnement du SDN et son architecture générale, avant d'étudier le protocole OpenFlow, qui assure la communication entre le contrôleur et les équipements réseau. Nous présenterons également les différents types de contrôleurs SDN ainsi que le contrôleur POX retenu dans le cadre de ce projet.

Cette étude permettra de comprendre les mécanismes qui rendent possible la gestion centralisée et dynamique des réseaux et constituera le fondement théorique nécessaire à la conception et à la mise en œuvre de l'architecture SDN développée dans ce mémoire.

2.2 Concept du Software Defined Networking

Le Software Defined Networking représente un changement de paradigme fondamental dans la conception et l'exploitation des réseaux. Plutôt que de considérer le réseau comme une accumulation d'équipements autonomes, le SDN propose de le traiter comme une ressource programmable et unifiée.

2.2.1 Définition du SDN

Le **Software-Defined Networking (SDN)** est une architecture réseau émergente qui repose sur la séparation physique et logique du **plan de contrôle** (Control Plane), responsable des décisions d'acheminement, et du **plan de données** (Data Plane), chargé du transfert effectif des paquets.

Contrairement aux réseaux conventionnels où chaque nœud intègre sa propre intelligence, le SDN centralise logiquement le contrôle au sein d'une entité logicielle appelée **« contrôleur SDN »**. Cette approche transforme les équipements réseau en dispositifs de transfert simplifiés, dont le comportement est dicté par des applications externes, permettant ainsi une gestion du trafic d'une souplesse sans précédent [3].

2.2.2 Principes fondamentaux

Le fonctionnement du SDN s'articule autour de trois principes piliers qui garantissent son efficacité :

1. **Séparation des plans :** Elle libère le matériel de la complexité des protocoles de routage distribués, simplifiant ainsi la conception des équipements de terrain (switches blancs ou ”bare-metal”).

2. **Centralisation logique :** Elle offre au contrôleur une vue omnisciente et en temps réel de la topologie complète. Cette abstraction permet de prendre des décisions globales plutôt que locales [2].

3. **Programmabilité via interfaces ouvertes :** L'usage d'API et de protocoles standardisés, tels qu'**OpenFlow** (Southbound Interface), permet une communication directe entre le contrôleur et les équipements. Cela rend le réseau entièrement automatisable et indépendant des solutions propriétaires des constructeurs [10].

2.2.3 Objectifs du SDN

L'implémentation d'une architecture SDN vise plusieurs objectifs stratégiques :

- **Agilité dynamique :** Permettre la modification des politiques réseau par simple mise à jour logicielle, répondant ainsi instantanément aux besoins des applications.
- **Optimisation des ressources (Traffic Engineering) :** Assurer une répartition intelligente de la charge sur l'ensemble des liens disponibles pour éviter les phénomènes de congestion.
- **Réduction des coûts (CapEx/OpEx) :** L'utilisation de matériels génériques moins coûteux (CapEx) et l'automatisation des tâches d'administration complexes (OpEx) réduisent significativement les dépenses globales.
- **Favorisation de l'innovation :** Offrir aux ingénieurs une plateforme flexible pour tester et déployer de nouveaux protocoles sans avoir à modifier le matériel physique [14].

2.3 Architecture SDN

L'architecture SDN repose sur une décomposition logique en trois couches distinctes, reliées par des interfaces de communication normalisées(Figure II.1). Cette organisation hiérarchique permet de séparer les fonctions de transmission physique de l'intelligence logicielle, offrant ainsi une modularité totale du réseau.

2.3.1 Plan de données

Le plan de données constitue la couche inférieure de l'architecture. Il est composé d'équipements réseau physiques ou virtuels (vSwitches) compatibles avec des protocoles de communication ouverts. Contrairement aux équipements traditionnels, ces dispositifs sont dépourvus d'intelligence propre : leur rôle est strictement limité au transfert (**Forwarding**), à la modification ou au rejet des paquets en fonction des règles inscrites dans leurs **tables de flux (Flow Tables)**. Cette simplification permet l'émergence du matériel ”bare-metal”, dont le comportement est entièrement dicté par la couche supérieure via l'interface sud (**Southbound Interface**), principalement le protocole **OpenFlow[2]**.

2.3.2 Plan de contrôle

Le plan de contrôle est le pilier central, agissant comme le système d'exploitation du réseau (Network OS). Il est matérialisé par un contrôleur SDN (tel que **POX**, Ryu ou ONOS) qui détient une vue omnisciente de la topologie. Le contrôleur maintient l'état global du réseau, traite les requêtes d'acheminement non répertoriées (Packet-In) et traduit les politiques de haut niveau en instructions de bas niveau (Flow-Mod). En centralisant logiquement le contrôle, il

garantit une cohérence absolue des politiques de sécurité et de gestion de trafic sur l'ensemble de l'infrastructure [10].

2.3.3 Plan application

Le plan application regroupe les services logiciels qui définissent la logique métier du réseau. Ces applications exploitent les abstractions fournies par le contrôleur pour implémenter des fonctions avancées : équilibrage de charge (Load Balancing), pare-feu dynamique ou routage intelligent basé sur la QoS. Grâce aux interfaces de programmation (**Northbound Interfaces**), souvent de type API REST, les développeurs peuvent concevoir des solutions réseau sur mesure sans contraintes matérielles. Cette couche transforme l'infrastructure en un service agile, capable d'évoluer par simple mise à jour logicielle [15]

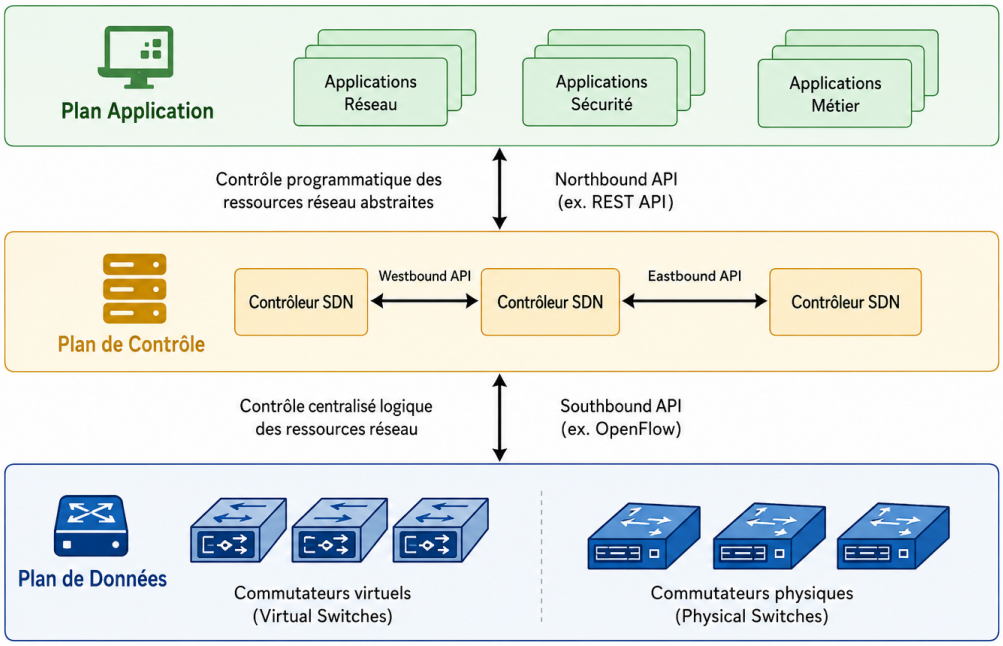


Fig. 2.1 : Architecture générale d'un réseau SDN basée sur OpenFlow

2.4 Protocole OpenFlow

Le protocole OpenFlow est historiquement le premier standard de communication de l'interface sud (*Southbound*) dans une architecture SDN. Il définit la manière dont un contrôleur peut interagir directement avec le plan de données pour modifier le comportement de transfert des commutateurs.

2.4.1 Principe de fonctionnement

Le fonctionnement d'OpenFlow repose sur une gestion du trafic par **flux**. Lorsqu'un paquet arrive sur un commutateur, celui-ci consulte ses **Flow Tables**. Deux scénarios sont possibles :

1. **Table Hit :** Une règle correspondante est trouvée ; le commutateur applique l'action associée (Forward, Drop, ou Modify).
2. **Table Miss :** Aucune règle ne correspond. Le commutateur encapsule le paquet (ou son en-tête) dans un message **Packet-In** envoyé au contrôleur. Ce dernier calcule la décision optimale et instruit le commutateur via un message **Flow-Mod** pour installer une nouvelle entrée.

Ce mécanisme permet de transformer un équipement de transfert passif en un nœud réseau intelligent et dynamiquement programmable [16].

2.4.2 Structure des flow tables

Chaque commutateur OpenFlow contient une ou plusieurs tables de flux, composées d'entrées de flux structurées de manière rigoureuse. Une entrée de flux s'articule autour de trois composants majeurs qui définissent le traitement du trafic. Tout d'abord, les champs de correspondance (**Match Fields**) permettent d'identifier le flux en examinant les informations des couches 2 à 4, telles que les adresses MAC, les adresses IP ou les numéros de ports. Ensuite, les compteurs (**Counters**) accumulent des statistiques précises sur le trafic, notamment le nombre de paquets et d'octets traités, ce qui s'avère crucial pour la surveillance et l'analyse des performances du réseau. Enfin, les instructions définissent l'action concrète à exécuter, comme l'envoi du paquet vers un port physique précis, la modification des en-têtes ou le passage vers une table suivante dans le cas d'un traitement multi-étapes [17].

2.4.3 Messages OpenFlow

La communication entre le contrôleur et le commutateur s'effectue à travers trois catégories de messages standardisés qui assurent la coordination du réseau. Les messages symétriques, tels que les ”**Hello**” ou les ”**Echo**”, sont utilisés principalement pour l'établissement de la session et le maintien de la connexion entre les entités. Les messages asynchrones sont quant à eux initiés par le commutateur pour signaler des événements au contrôleur, notamment via le message **Packet-In** pour un flux inconnu ou le message **Flow-Removed** lors de l'expiration d'une règle. Enfin, les messages de contrôle sont envoyés par le contrôleur pour administrer l'état du switch, le plus emblématique étant le message **Flow-Mod** qui permet d'ajouter, de modifier ou de supprimer dynamiquement des entrées dans les tables de flux. Cette gestion

par messages garantit une réactivité optimale de l'infrastructure face aux évolutions du trafic [2].

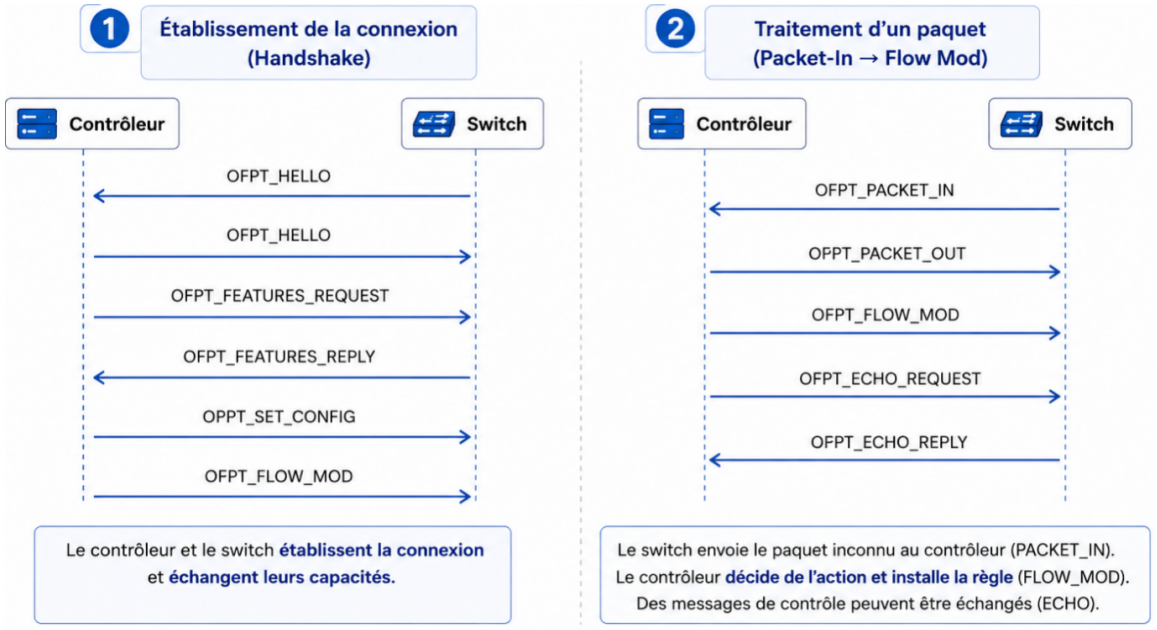


Fig. 2.2 : Cycle de traitement d'un flux OpenFlow

2.5 Contrôleurs SDN

Le contrôleur est l'élément de centralisation logique qui définit l'intelligence du réseau SDN. Il agit comme un système d'exploitation réseau, faisant le lien entre les exigences des applications et les capacités matérielles des commutateurs.

2.5.1 Rôle du contrôleur

Le contrôleur SDN occupe une position stratégique dans l'architecture en tant qu'intermédiaire entre les couches applicative et infrastructurelle. Son rôle principal est de maintenir une vue complète et cohérente de la topologie du réseau en collectant en temps réel les informations provenant des équipements de transfert. Il traduit les politiques de haut niveau, telles que les règles de sécurité ou de gestion de la Qualité de Service (QoS), en instructions de bas niveau (entrées de flux) qu'il déploie via le protocole OpenFlow. En gérant les conflits de règles et en orchestrant le routage global, le contrôleur garantit que le réseau fonctionne comme une entité unique et programmable plutôt que comme une somme d'équipements indépendants [10].

2.5.2 Présentation de POX

POX est un contrôleur SDN open-source développé en Python, utilisé principalement pour l'expérimentation et le développement des architectures Software Defined Networking basées sur le protocole OpenFlow. Il assure la gestion centralisée du réseau en permettant le contrôle des switches OpenFlow à travers le plan de contrôle SDN.

Le contrôleur POX repose sur une architecture modulaire et événementielle permettant de développer différents services réseau tels que le forwarding, le routage, la supervision du trafic et la gestion des flux réseau. Il traite dynamiquement les événements générés par les switches OpenFlow, notamment les messages *Packet-In*, *Packet-Out* et *Flow-Mod*.

Grâce à sa simplicité, sa flexibilité et sa compatibilité avec OpenFlow, POX constitue une solution adaptée aux environnements académiques et aux plateformes expérimentales dédiées à l'étude et à la compréhension des mécanismes SDN [18]

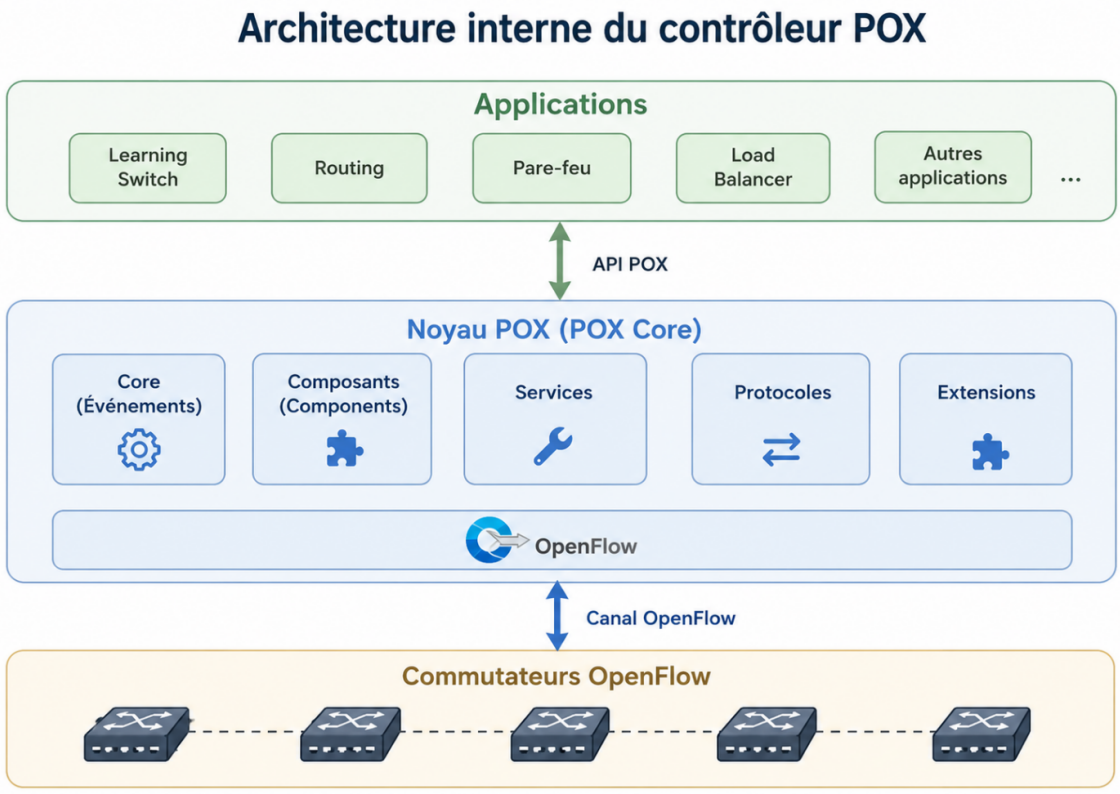


Fig. 2.3 : Architecture interne du contrôleur POX

2.5.3 Comparaison avec autres contrôleurs

Le choix d'un contrôleur dépend des objectifs de performance, de langage et d'utilisation. Contrairement à POX, qui est privilégié pour sa simplicité en Python, des contrôleurs comme **Floodlight** ou **Beacon** sont écrits en Java et offrent des performances plus élevées pour les environnements de production. À un niveau encore plus industriel, **OpenDaylight (ODL)**

et **ONOS** se distinguent par leur robustesse et leur capacité à gérer des réseaux de très grande taille avec une haute disponibilité. Cependant, ces plateformes présentent une courbe d'apprentissage beaucoup plus complexe. Alors que **Ryu** propose une alternative moderne en Python avec un support étendu des versions récentes d'OpenFlow, POX demeure une référence pour l'apprentissage et la démonstration de concepts fondamentaux de routage programmable **[19](**Tableau 2.1)

Tab. 2.1 : Comparaison entre les principaux contrôleurs SDN

Contrôleur	**Langage**	**Caractéristiques principales**	**Domaine d'utilisation**
POX	Python	Simple, léger et modulaire	Environnements académiques
Ryu	Python	Support avancé d'OpenFlow et grande extensibilité	Recherche et expérimentation
Floodlight	Java	Performances élevées et API REST	Réseaux d'entreprise
OpenDaylight	Java	Architecture modulaire et haute scalabilité	Environnements industriels
ONOS	Java	Haute disponibilité et gestion distribuée	Grands réseaux opérateurs

2.6 Avantages et limites du SDN

Le passage au paradigme SDN apporte des transformations profondes dans la gestion des infrastructures, mais comme toute technologie en évolution, il présente un ensemble de bénéfices et de défis qu'il convient d'analyser.

2.6.1 Avantages du SDN

L'adoption du SDN procure des gains significatifs en termes d'agilité et d'efficacité opérationnelle. Le premier avantage réside dans la **gestion centralisée**, qui permet aux administrateurs de configurer l'ensemble du réseau à partir d'un point unique, éliminant ainsi les erreurs liées aux configurations manuelles par équipement. Ensuite, la **programmabilité** offre une flexibilité sans précédent : les politiques réseau peuvent être modifiées dynamiquement par logiciel pour répondre instantanément aux besoins des applications. En outre, le SDN favorise l'**optimisation des ressources** grâce à une ingénierie de trafic globale, permettant une meilleure répartition de la charge et une réduction des congestions. Enfin, l'utilisation

d'interfaces ouvertes réduit la dépendance vis-à-vis des solutions propriétaires (**vendor lock-in**), ce qui stimule l'innovation tout en réduisant les coûts d'infrastructure [14].

2.6.2 Limites du SDN

Malgré ses promesses, le SDN soulève des problématiques, notamment en matière de sécurité et de passage à l'échelle. La centralisation du plan de contrôle crée un **"point de défaillance unique"** (Single Point of Failure) : si le contrôleur est compromis ou subit une panne, l'ensemble du réseau peut devenir instable. De plus, le contrôleur peut devenir un goulot d'étranglement si le nombre de demandes **Packet-In** provenant des commutateurs est trop élevé, limitant ainsi la **scalabilité** dans de très grands centres de données [15]. Il existe également des défis liés à l'interopérabilité entre les équipements SDN et les infrastructures héritées (legacy), ce qui rend la migration complexe. Enfin, la gestion d'un réseau programmable nécessite des compétences hybrides en réseaux et en développement logiciel, représentant un défi de formation pour les équipes techniques [10].

2.7 Conclusion

Ce deuxième chapitre a permis d'explorer les fondements théoriques et les composants technologiques du Software Defined Networking. En nous penchant sur la séparation des plans de contrôle et de données, nous avons compris comment le SDN transforme une infrastructure rigide en une plateforme logicielle agile et programmable. L'étude du protocole OpenFlow et du contrôleur POX a mis en évidence les mécanismes techniques permettant une interaction directe et granulaire avec les flux réseau.

Bien que le SDN présente des défis, notamment en matière de centralisation et de sécurité, ses avantages en termes d'automatisation et d'optimisation de la Qualité de Service (QoS) en font le candidat idéal pour notre projet. La compréhension de ces concepts est essentielle pour aborder la phase suivante de notre travail. Le chapitre suivant sera consacré à la conception et à la mise en œuvre de notre solution : un routeur SDN OpenFlow capable d'assurer une gestion intelligente des flux, en s'appuyant sur les outils et les principes présentés ici.

Chapitre 3

Conception, implémentation et validation de l'architecture SDN OpenFlow

3.1 Introduction

Après avoir présenté les concepts fondamentaux des réseaux IP ainsi que les principes du Software Defined Networking dans les chapitres précédents, ce chapitre est consacré à la conception, à l'implémentation et à la validation de l'architecture SDN réalisée dans le cadre de ce projet.

L'objectif principal de cette implémentation est de mettre en place une infrastructure réseau basée sur le protocole OpenFlow permettant une gestion centralisée et intelligente des flux réseau à l'aide d'un contrôleur SDN. Pour atteindre cet objectif, une architecture composée de plusieurs commutateurs Open vSwitch interconnectés a été déployée dans un environnement virtualisé supervisé par le contrôleur POX.

Ce chapitre présente dans un premier temps la solution proposée ainsi que l'environnement expérimental utilisé pour le déploiement du réseau SDN. Ensuite, les différentes étapes de conception et de configuration des équipements réseau seront détaillées, notamment la mise en œuvre des switches OpenFlow, la configuration du contrôleur SDN ainsi que l'automatisation des configurations réseau.

Enfin, plusieurs tests de validation et d'analyse des performances seront réalisés afin d'évaluer le fonctionnement de l'architecture implémentée, d'analyser les échanges OpenFlow ainsi que de démontrer l'apport du SDN dans la gestion dynamique et optimisée des flux réseau.

3.2 Présentation de la solution SDN et de l'environnement expérimental

3.2.1 Présentation générale de la solution

La solution proposée dans ce travail repose sur la mise en œuvre d'une architecture Software Defined Networking (SDN) basée sur le protocole OpenFlow afin d'assurer une gestion centralisée et dynamique des flux réseau. Cette architecture s'appuie sur la séparation du plan de contrôle et du plan de données, permettant une meilleure programmabilité et une supervision globale du réseau.

L'infrastructure déployée est composée de switches Open vSwitch interconnectés et pilotés par un contrôleur SDN POX chargé de gérer les communications réseau et d'installer dynamiquement les règles de forwarding dans les Flow Tables des switches OpenFlow.

La plateforme expérimentale a été réalisée dans un environnement virtualisé sous VMware Workstation exécutant des machines virtuelles Ubuntu Linux. Plusieurs outils tels que Wireshark et iperf3 ont été utilisés afin d'analyser le trafic réseau et d'évaluer les performances de l'architecture SDN implémentée.

Cette solution vise principalement à démontrer l'apport du SDN dans l'optimisation du trafic réseau, la gestion intelligente des flux ainsi que la simplification de l'administration des infrastructures réseau modernes.

3.2.2 Infrastructure matérielle utilisée

Cette partie présente les principaux équipements matériels utilisés pour la mise en œuvre et l'expérimentation de l'architecture SDN proposée. L'ensemble de ces composants a permis d'assurer le déploiement, la configuration et la validation de l'infrastructure réseau implémentée.

3.2.2.1 Ordinateur hôte

L'environnement expérimental de l'architecture SDN a été déployé sur un ordinateur portable HP EliteBook 645 G9 utilisé comme machine hôte pour l'exécution de la plateforme de virtualisation VMware Workstation et des différentes machines virtuelles Ubuntu Linux.

3.2.2.2 Plateforme Intel Galileo

Intel Galileo est une plateforme embarquée basée sur l'architecture Intel x86 permettant l'exécution d'applications Linux et de services réseau. Elle est largement utilisée dans les projets

de systèmes embarqués et les architectures réseau programmables.

Dans cette étude, elle a été utilisée pour héberger le contrôleur SDN POX assurant la gestion centralisée des switches OpenFlow. [20]

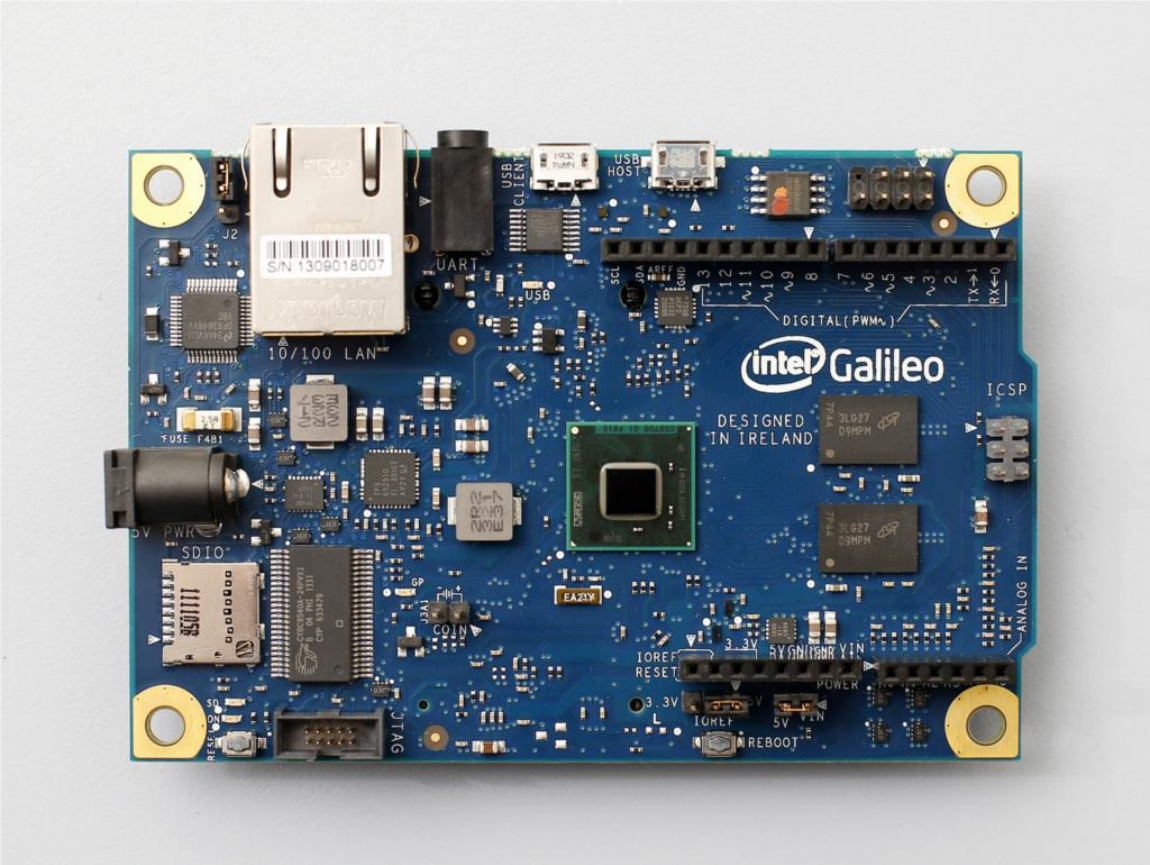


Fig. 3.1 : Plateforme Intel Galileo

3.2.2.3 Machines virtuelles

Plusieurs machines virtuelles Ubuntu ont été déployées afin d'implémenter les switches Open vSwitch de l'architecture SDN. Chaque machine virtuelle exécute un switch OpenFlow connecté au contrôleur POX et participe au transfert des paquets ainsi qu'aux échanges OpenFlow. Les hôtes ont été positionnés au niveau d'OVS1 et d'OVS2 afin de générer un trafic traversant l'ensemble de la topologie, tandis qu'OVS3 assure le rôle de switch intermédiaire. Cette organisation permet d'analyser le comportement du contrôleur, les échanges OpenFlow ainsi que l'installation dynamique des règles dans les Flow Tables.

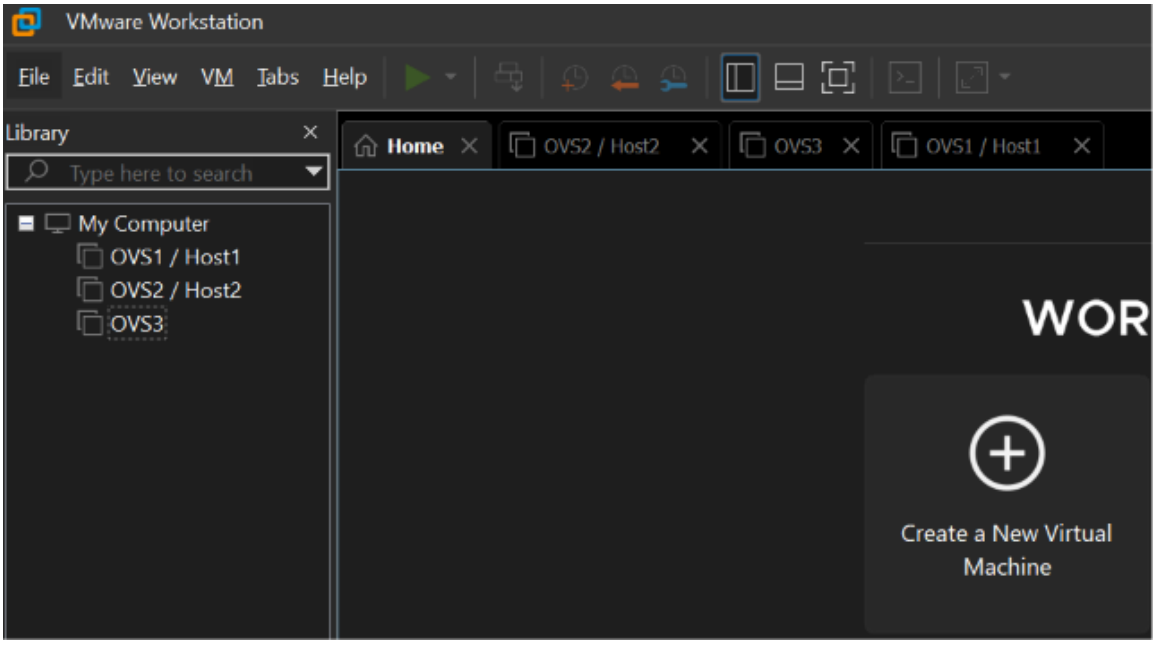


Fig. 3.2 : Machines virtuelles utilisées dans l'architecture SDN

3.2.3 Environnement logiciel et technologies utilisées

La mise en œuvre de l'architecture SDN proposée a nécessité l'utilisation de plusieurs outils logiciels et technologies permettant le déploiement, la configuration, la supervision et l'évaluation des performances du réseau expérimental.

3.2.3.1 VMware Workstation

VMware Workstation a été utilisé comme plateforme de virtualisation afin de créer et d'exécuter les différentes machines virtuelles composant l'environnement expérimental. Cette solution permet de simuler une infrastructure réseau complète dans un environnement flexible et isolé [20] Dans cette étude, le mode réseau Bridged a été utilisé afin de permettre la communication directe entre les machines virtuelles Ubuntu et la plateforme Intel Galileo hébergeant le contrôleur SDN POX. Cette configuration permet aux switches OpenFlow virtualisés d'accéder au réseau physique et d'établir les connexions OpenFlow avec le contrôleur SDN via le réseau local.

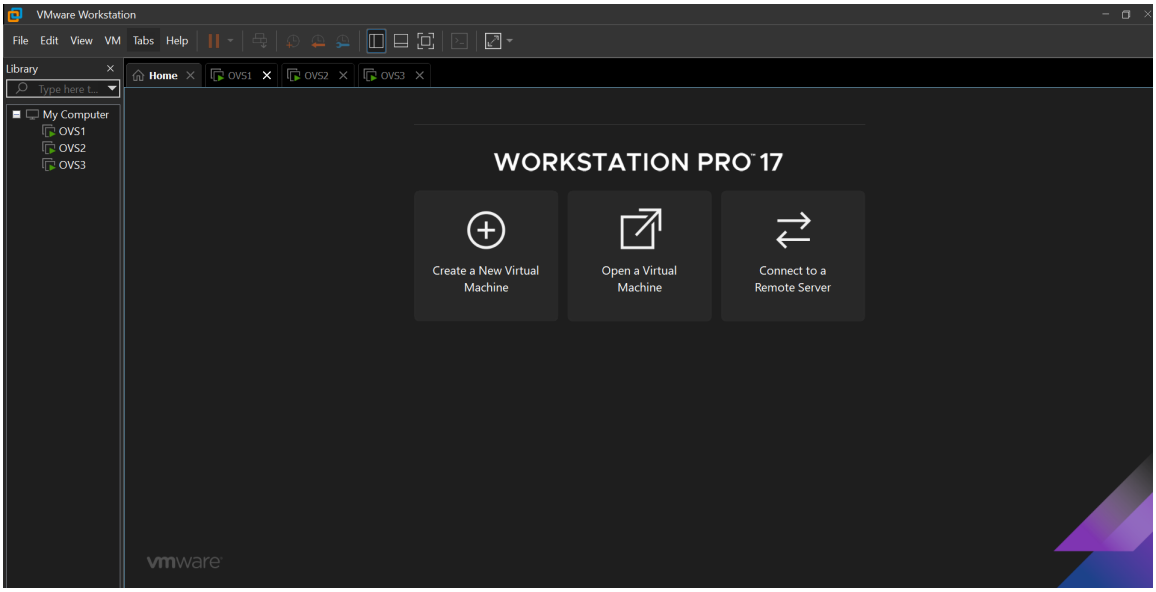


Fig. 3.3 : Interfaces de logiciel VMware

3.2.3.2 Ubuntu Linux

Ubuntu Linux a été utilisé comme système d'exploitation principal des machines virtuelles déployées dans l'architecture SDN. Grâce à sa stabilité et sa compatibilité avec les outils réseau open-source, il constitue une plateforme adaptée au déploiement des solutions SDN et OpenFlow[21]

3.2.3.3 Open vSwitch (OVS)

Open vSwitch (OVS) est un switch virtuel programmable utilisé pour implémenter les commutateurs OpenFlow dans l'architecture SDN. Il assure le forwarding des paquets réseau et permet l'intégration dynamique des règles OpenFlow installées par le contrôleur SDN **[22].**

```
zaid@zaid-VMware-Virtual-Platform:~$ sudo ovs-vsctl show
02c2aefb-1145-4055-9892-78b751c6d008
    Bridge br0
        Controller "tcp:192.168.1.34:6633"
        fail_mode: secure
        Port ens37
            Interface ens37
        Port br0
            Interface br0
                type: internal
    ovs_version: "3.3.4"
```

Fig. 3.4 : Version d'Open vSwitch

3.2.3.4 Contrôleur POX

Le contrôleur POX représente l'élément central du plan de contrôle dans l'architecture SDN implémentée. Développé en Python, il assure la gestion centralisée des switches OpenFlow ainsi que l'installation dynamique des règles de forwarding**[23].**

```
root@galileo:~/pox# ./pox.py log.level --DEBUG forwarding.l2_learning
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.1.0 (betta) going up...
DEBUG:core:Running on CPython (2.7.3/May 29 2015 21:31:34)
DEBUG:core:Platform is Linux-3.8.7-yocto-standard-i586-with-glibc2.0
INFO:core:POX 0.1.0 (betta) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
```

Fig. 3.5 : Lancement du contrôleur SDN POX

3.2.3.5 OpenFlow

OpenFlow est le protocole de communication utilisé entre les switches OpenFlow et le contrôleur SDN. Il permet l'échange des messages de contrôle nécessaires à la gestion dynamique des flux réseau **[17].**

3.2.3.6 Wireshark

Wireshark est un outil d'analyse réseau utilisé pour capturer et superviser le trafic circulant dans l'architecture SDN. Il permet notamment d'analyser les paquets ICMP, ARP ainsi que les messages OpenFlow **[24].**

```
root@galileo:~/pox# ./pox.py log.level --DEBUG forwarding.l2_learning
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.1.0 (betta) going up...
DEBUG:core:Running on CPython (2.7.3/May 29 2015 21:31:34)
DEBUG:core:Platform is Linux-3.8.7-yocto-standard-i586-with-glibc2.0
INFO:core:POX 0.1.0 (betta) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
```

Fig. 3.6 : Interface graphique du logiciel Wireshark

3.2.3.7 iperf3

Iperf3 est un outil de mesure des performances réseau utilisé pour évaluer le débit et la qualité des communications entre les équipements de l'architecture SDN **[25].**

3.3 Conception et déploiement de l'architecture SDN

3.3.1 Architecture et topologie du réseau SDN

L'architecture SDN mise en œuvre repose sur une organisation centralisée du contrôle réseau à travers un contrôleur POX hébergé sur la plateforme Intel Galileo. Cette architecture est composée de trois switches Open vSwitch (OVS1, OVS2 et OVS3) interconnectés et pilotés par le contrôleur via le protocole OpenFlow utilisant le port TCP 6633.

La topologie déployée est composée de trois switches Open vSwitch interconnectés le switch OVS3 joue le rôle de switch central et assure l'interconnexion entre OVS1 et OVS2 à travers deux segments réseau distincts. Le premier segment (LAN Segment 1) relie OVS1 à OVS3 via les interfaces ens37, tandis que le second segment (LAN Segment 2) assure la liaison entre OVS3 et OVS2 via les interfaces ens38 et ens37.

Deux machines virtuelles Ubuntu ont été utilisées comme hôtes de communication au sein de la topologie. Le premier hôte (Host1), configuré avec l'adresse IP **192.168.1.50/24**, est connecté au switch OVS1, tandis que le second hôte (Host2), configuré avec l'adresse IP **192.168.1.51/24**, est connecté au switch OVS2. Les switches Open vSwitch sont implémentés sous forme de bridges virtuels (**br0**) à l'intérieur des machines virtuelles, permettant ainsi le transfert des paquets selon les règles OpenFlow installées par le contrôleur. Le switch OVS3 joue le rôle de nœud intermédiaire assurant le transit du trafic entre les deux extrémités du réseau.

Les trois switches Open vSwitch (OVS1, OVS2 et OVS3) sont connectés au contrôleur POX à travers leurs interfaces **ens33**, reliées au réseau de contrôle. Le contrôleur, dont l'adresse IP est **192.168.1.34/24**, communique avec les switches via le protocole OpenFlow sur le port **TCP 6633**. Cette connexion permet l'échange des messages OpenFlow nécessaires à la supervision de l'infrastructure et à l'installation dynamique des règles de forwarding dans les

Flow Tables des switches.

Cette architecture permet de mettre en évidence le principe fondamental du SDN basé sur la séparation du plan de contrôle et du plan de données. Elle offre également un environnement adapté pour analyser les échanges OpenFlow, observer l'installation dynamique des règles de forwarding et évaluer les performances de la gestion centralisée des flux réseau

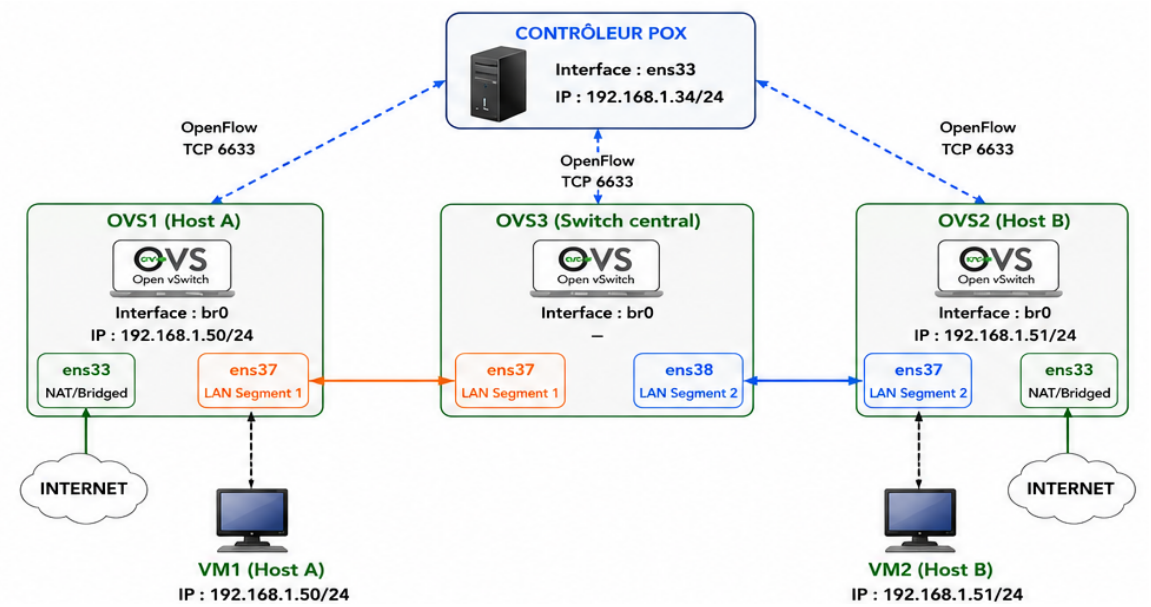


Fig. 3.7 : Topologie générale de l'architecture SDN OpenFlow déployée

3.3.2 Configuration de l'infrastructure OpenFlow

Cette étape consiste à configurer l'infrastructure OpenFlow utilisée dans l'architecture SDN déployée. Les opérations réalisées concernent principalement la création des switches Open vSwitch, la configuration des interfaces réseau, l'établissement des communications OpenFlow ainsi que l'attribution des adresses IP aux différents équipements réseau.

3.3.2.1 Création des switches Open vSwitch

Les switches Open vSwitch ont été créés afin d'assurer le forwarding des paquets dans le plan de données de l'architecture SDN. Trois switches OpenFlow nommés OVS1, OVS2 et OVS3 ont été déployés dans l'environnement expérimental.

La création des bridges Open vSwitch a été réalisée à l'aide des commandesshelle suivantes :

sudo ovs-vsctl add-br br0

Cette commande permet de créer un bridge OpenFlow virtuel nommé br0 utilisé comme switch principal dans chaque machine virtuelle OVS.

3.3.2.2 Configuration des interfaces réseau

Les interfaces réseau virtuelles ont été configurées afin d'assurer les communications entre les switches OpenFlow ainsi qu'entre les hôtes réseau et le contrôleur SDN.

L'ajout des interfaces réseau au bridge Open vSwitch a été réalisé avec les commandes shel-lesuivantes :

sudo ovs-vsctladd-port br0 ens37

sudo ovs-vsctladd-port br0 ens38

Les interfaces ont ensuite été activées à l'aide de la commandeshelle :

sudo ip link set ens37 up

sudo ip link set ens38 up

```
zaid@zaid-VMware-Virtual-Platform:~$ sudo ovs-vsctl show
02c2aefb-1145-4055-9892-78b751c6d008
    Bridge br0
        Controller "tcp:192.168.1.34:6633"
            is_connected: true
        Port br0
            Interface br0
                type: internal
        Port ens38
            Interface ens38
        Port ens37
            Interface ens37
    ovs_version: "3.3.4"
zaid@zaid-VMware-Virtual-Platform:~$
```

Fig. 3.8 : Configuration des interfaces réseau des switches OpenFlow

3.3.2.3 Plan d'adressage IP et attribution des adresses réseau

Un plan d'adressage IP a été mis en place afin d'assurer la communication entre les différents équipements de l'architecture SDN. Chaque équipement réseau dispose d'une adresse IP appartenant au réseau expérimental utilisé dans cette étude.

L'attribution des adresses IP a été réalisée à l'aide des commandes suivantes :

sudo ip addr add 192.168.1.50/24 dev br0

sudo ip addr add 192.168.1.51/24 dev br0

Cette configuration permet d'assurer la connectivité réseau entre les différents hôtes et switches OpenFlow. Capture des adresses IP configuréesavec :**ipaddr**

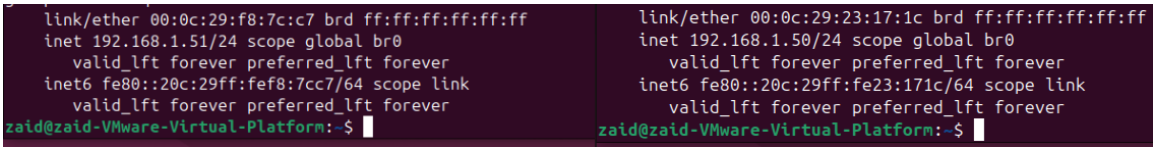


Fig. 3.9 : Plan d'adressage IP

3.3.3 Mise en œuvre du contrôleur SDN POX

3.3.3.1 Installation, configuration et lancement du contrôleur POX

Le contrôleur SDN POX a été installé et exécuté sur la plateforme embarquée Intel Galileo utilisée comme plan de contrôle dans l'architecture SDN déployée. Cette plateforme assure la gestion centralisée des switches OpenFlow et supervise les communications réseau à travers le protocole OpenFlow.

Après l'installation des dépendances nécessaires sous Linux, le contrôleur POX a été configuré puis lancé avec le module forwarding.l2_learning permettant l'apprentissage automatique des adresses MAC ainsi que l'installation dynamique des règles de forwarding dans les switches Open vSwitch.

Le lancement du contrôleur a été réalisé à l'aide de la commandeshelle suivante :

./pox.py log.level--DEBUG forwarding.l2_learning

Cette configuration permet au contrôleur POX d'intercepter les messages OpenFlow de type *Packet-In*, d'analyser les flux réseau et d'installer dynamiquement les règles OpenFlow nécessaires au forwarding des paquets dans l'infrastructure SDN.

3.3.3.2 Connexion des switches OpenFlow au contrôleur

Après le lancement du contrôleur SDN POX sur la plateforme Intel Galileo, les switches Open vSwitch ont été configurés afin d'établir une connexion OpenFlow avec le contrôleur. Cette communication permet au contrôleur de superviser le réseau et de gérer dynamiquement les flux réseau.

La configuration des connexions OpenFlow a été réalisée à l'aide de la commande suivante :

sudo ovs-vsctl set-controller br0 tcp :192.168.1.34:6633

Cette commandeshelle permet d'associer le bridge Open vSwitch**br0** au contrôleur POX via le protocole OpenFlow utilisant le port TCP 6633.

Une fois la connexion établie, les switches OpenFlow transmettent les messages *Packet-In* au contrôleur lorsqu'aucune règle correspondante n'est présente dans les Flow Tables. Le contrôleur analyse alors les informations du trafic réseau puis installe dynamiquement les règles OpenFlow nécessaires au forwarding des paquets.

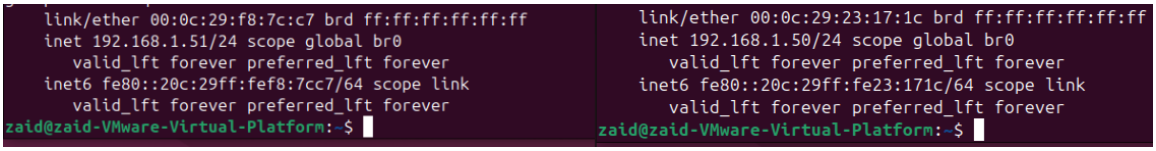


Fig. 3.10 : Établissement des connexions OpenFlow entre les switches OVS et le contrôleur POX

3.4 Validation et tests de l'architecture SDN

Cette étape a pour objectif de vérifier le bon fonctionnement de l'architecture SDN déployée ainsi que les échanges OpenFlow entre les différents équipements réseau. Plusieurs tests ont été réalisés afin de valider la connectivité réseau, l'installation des règles OpenFlow et le fonctionnement du contrôleur SDN POX.

3.4.1 Vérification de la connectivité réseau

Afin de vérifier le bon fonctionnement des communications réseau au sein de l'architecture SDN, des tests de connectivité ont été réalisés entre les différents équipements connectés aux switches OpenFlow.

La commande ping a été utilisée afin de tester l'échange des paquets ICMP entre les hôtes réseau et de confirmer la circulation correcte du trafic dans l'infrastructure SDN.

Le test suivant a été exécuté :

ping 192.168.1.51

Les résultats obtenus montrent la réception correcte des réponses ICMP avec un taux de perte nul, confirmant ainsi le bon fonctionnement des communications réseau et des connexions OpenFlow au sein de l'architecture SDN.

```
zaid@zaid-VMware-Virtual-Platform:~$ ping 192.168.1.51
PING 192.168.1.51 (192.168.1.51) 56(84) bytes of data.
64 bytes from 192.168.1.51: icmp_seq=1 ttl=64 time=303 ms
64 bytes from 192.168.1.51: icmp_seq=2 ttl=64 time=60.4 ms
64 bytes from 192.168.1.51: icmp_seq=3 ttl=64 time=2.31 ms
64 bytes from 192.168.1.51: icmp_seq=4 ttl=64 time=1.38 ms
64 bytes from 192.168.1.51: icmp_seq=5 ttl=64 time=1.18 ms
64 bytes from 192.168.1.51: icmp_seq=6 ttl=64 time=1.47 ms
64 bytes from 192.168.1.51: icmp_seq=7 ttl=64 time=1.51 ms
64 bytes from 192.168.1.51: icmp_seq=8 ttl=64 time=1.55 ms
64 bytes from 192.168.1.51: icmp_seq=9 ttl=64 time=1.32 ms
64 bytes from 192.168.1.51: icmp_seq=10 ttl=64 time=1.22 ms
64 bytes from 192.168.1.51: icmp_seq=11 ttl=64 time=1.87 ms
64 bytes from 192.168.1.51: icmp_seq=12 ttl=64 time=1.33 ms
64 bytes from 192.168.1.51: icmp_seq=13 ttl=64 time=1.04 ms
64 bytes from 192.168.1.51: icmp_seq=14 ttl=64 time=1.49 ms
64 bytes from 192.168.1.51: icmp_seq=15 ttl=64 time=1.47 ms
^C
--- 192.168.1.51 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14027ms
rtt min/avg/max/mdev = 1.037/25.477/302.627/75.509 ms
zaid@zaid-VMware-Virtual-Platform:~$
```

Fig. 3.11 : Vérification de la connectivité réseau dans l'architecture SDN

3.4.2 Analyse des Flow Tables OpenFlow

Afin d'analyser le fonctionnement du protocole OpenFlow, les Flow Tables des switches Open vSwitch ont été examinées après l'établissement des communications réseau.

La consultation des règles OpenFlow a été réalisée avec la commandeshelle suivante :

sudoovs-ofctl dump-flows br0

Les résultats affichés dans les Flow Tables montrent les règles dynamiquement installées par le contrôleur SDN POX après les échanges réseau entre les équipements. Ces règles contiennent notamment les adresses MAC source et destination, les paquets ICMP et ARP ainsi que les actions de forwarding appliquées par les switches OpenFlow.

En parallèle, les messages affichés par le contrôleur POX montrent l'installation dynamique des flux OpenFlow suite aux messages *Packet-In* envoyés par les switches Open vSwitch.

L'analyse des Flow Tables et des messages du contrôleur confirme le bon fonctionnement des échanges OpenFlow ainsi que la gestion dynamique des flux réseau dans l'architecture SDN déployée.

- Capture du terminal POX affichant l'installation dynamique des règles OpenFlow.

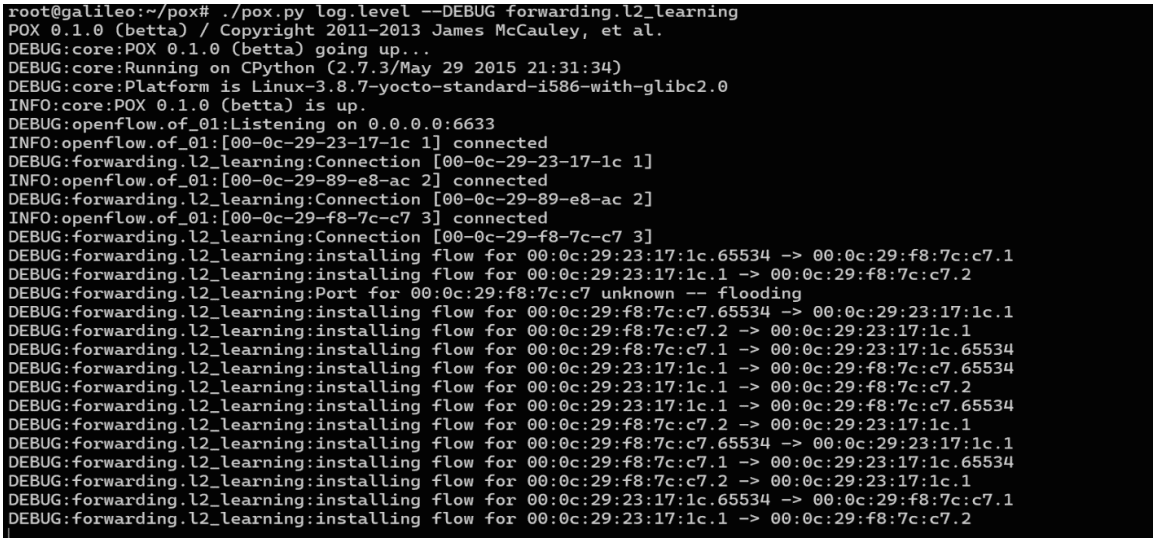


Fig. 3.12 : Installation dynamique des règles OpenFlow par le contrôleur POX

- Capture du résultat de la commandeshelle**ovs-ofctl dump-flows br0**.

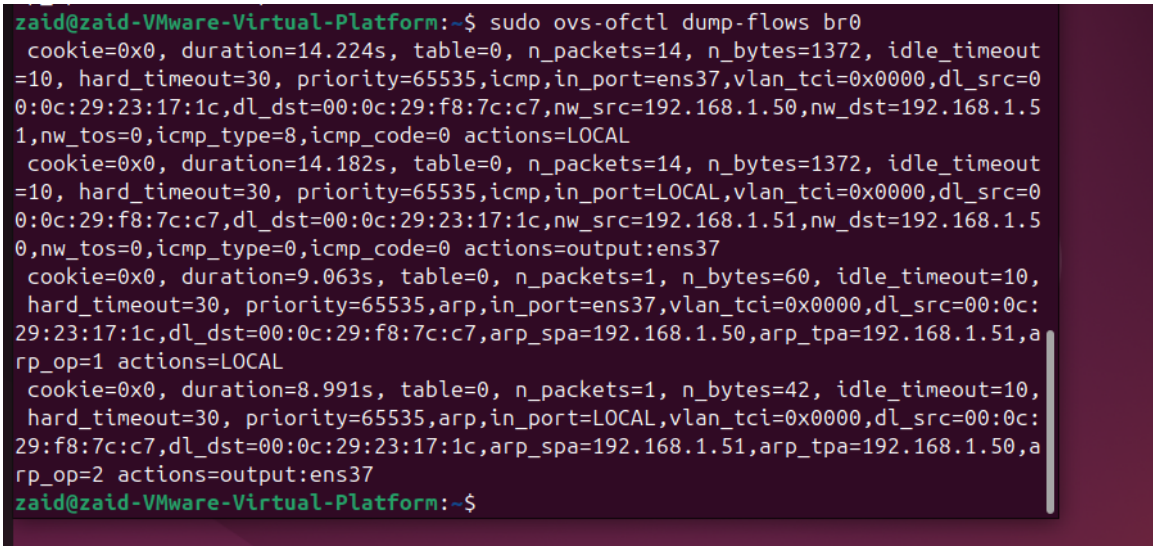


Fig. 3.13 : Analyse des règles OpenFlow dans les Flow Tables du switch Open vSwitch

3.4.3 Analyse du trafic réseau avec Wireshark

Afin d'analyser les échanges réseau au sein de l'architecture SDN déployée, l'outil Wireshark a été utilisé pour capturer et superviser les paquets circulant entre les switches Open vSwitch et le contrôleur SDN POX.

Les captures réseau ont été réalisées pendant les tests de communication afin d'observer les échanges OpenFlow ainsi que les différents paquets réseau circulant dans l'infrastructure SDN.

L'analyse des captures a permis d'identifier plusieurs messages OpenFlow, notamment les messages OFPT_PACKET_IN et OFPT_PACKET_OUT échangés entre les switches OpenFlow et le contrôleur POX via le port TCP 6633.

Les filtres Wireshark suivants ont été utilisés afin d'analyser le trafic réseau :

tcp.port == 6633

Ce filtre permet d'afficher les échanges OpenFlow entre le plan de contrôle et le plan de données.

Les résultats obtenus confirment le bon fonctionnement des communications OpenFlow ainsi que la gestion dynamique des flux réseau par le contrôleur SDN POX.

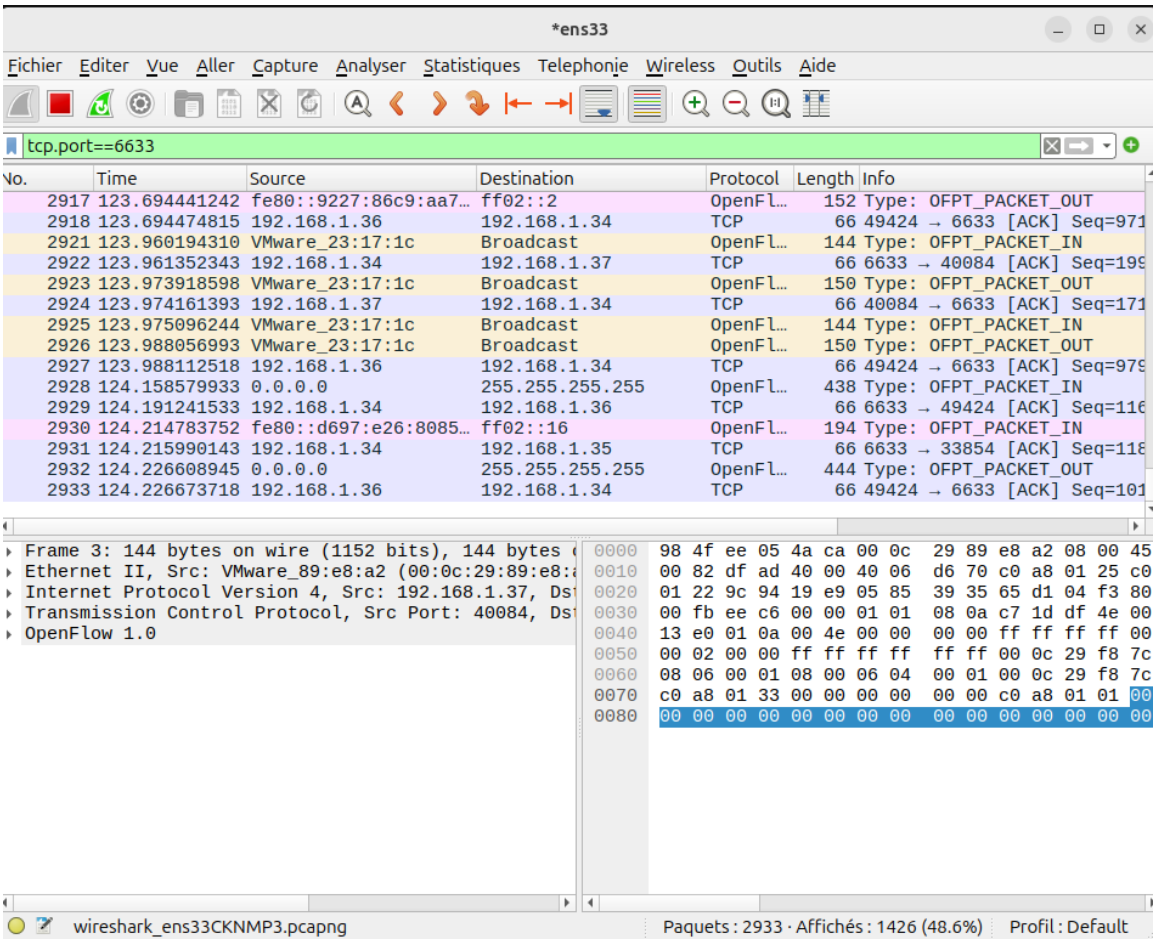


Fig. 3.14 : Analyse des échanges OpenFlow avec Wireshark

3.4.4 Évaluation des performances réseau avec iperf3

Afin d'évaluer les performances de l'architecture SDN déployée, des tests de débit réseau ont été réalisés à l'aide de l'outil iperf3. Cet outil permet de mesurer la bande passante ainsi que les performances des communications entre les équipements connectés aux switches OpenFlow.

Le mode serveur a été lancé sur l'un des équipements réseau avec la commande suivante :

iperf3 -s

Le test client a ensuite été exécuté depuis une autre machine avec :

iperf3 -c 192.168.1.51

Les résultats obtenus permettent d'évaluer le débit des communications réseau ainsi que la stabilité des échanges de données au sein de l'infrastructure SDN.

L'utilisation d'iperf3 a permis de confirmer le bon fonctionnement des communications réseau et les performances satisfaisantes de l'architecture OpenFlow déployée.

```
zaid@zaid-VMware-Virtual-Platform:~$ iperf3 -c 192.168.1.51
Connecting to host 192.168.1.51, port 5201
[  5] local 192.168.1.50 port 37576 connected to 192.168.1.51 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  28.0 MBytes   234 Mbits/sec    2   1.29 MBytes
[  5]   1.00-2.00   sec  0.00 Bytes  0.00 bits/sec    0   1.29 MBytes
[  5]   2.00-3.00   sec  1.38 MBytes  11.5 Mbits/sec    0   1.29 MBytes
[  5]   3.00-4.00   sec  0.00 Bytes  0.00 bits/sec    0   1.29 MBytes
[  5]   4.00-5.00   sec  46.9 MBytes   393 Mbits/sec   89   1.41 KBytes
[  5]   5.00-6.00   sec  0.00 Bytes  0.00 bits/sec    1   1.46 MBytes
[  5]   6.00-7.00   sec  0.00 Bytes  0.00 bits/sec    0   1.46 MBytes
[  5]   7.00-8.00   sec  0.00 Bytes  0.00 bits/sec    1   1.47 MBytes
[  5]   8.00-9.00   sec  0.00 Bytes  0.00 bits/sec    0   1.47 MBytes
[  5]   9.00-10.00  sec  50.5 MBytes   423 Mbits/sec   73   1.41 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec   127 MBytes   106 Mbits/sec  166             sender
[  5]   0.00-16.41  sec   127 MBytes  64.8 Mbits/sec                  receiver

iperf Done.
zaid@zaid-VMware-Virtual-Platform:~$
```

Fig. 3.15 : Évaluation des performances réseau avec iperf3

3.5 Analyse et interprétation des résultats

Cette section est consacrée à l'analyse des résultats obtenus lors des différentes expérimentations réalisées sur l'architecture SDN mise en œuvre. Les tests effectués ont permis d'évaluer le fonctionnement des communications réseau, le comportement du protocole OpenFlow ainsi que l'efficacité du contrôleur POX dans la gestion des flux. L'interprétation des résultats obtenus permet également de mettre en évidence les avantages de l'approche SDN en termes de centralisation du contrôle, de flexibilité et d'optimisation de la gestion du trafic réseau.

3.5.1 Analyse des performances obtenues

Les résultats obtenus à travers les différents tests réalisés confirment le bon fonctionnement de l'architecture SDN déployée. Les essais de connectivité effectués à l'aide de la commande

ping ont montré une communication stable entre les différents équipements du réseau, avec un taux de perte nul, ce qui valide la configuration correcte des interfaces réseau ainsi que le bon acheminement des paquets au sein de l'infrastructure.

L'analyse des Flow Tables OpenFlow a permis de vérifier l'installation dynamique des règles de transfert par le contrôleur POX. En effet, lorsqu'un paquet ne correspond à aucune entrée existante dans la table de flux, le switch Open vSwitch transmet un message ***Packet-In*** au contrôleur. Celui-ci analyse les informations reçues puis installe automatiquement les règles appropriées, permettant ainsi d'optimiser les traitements ultérieurs et de réduire les échanges de signalisation.

Les captures réalisées avec Wireshark ont également confirmé le bon fonctionnement des communications OpenFlow entre le contrôleur et les switches. L'observation des messages ***OFPT_PACKET_IN***et ***OFPT_PACKET_OUT***met en évidence l'interaction permanente entre le plan de contrôle et le plan de données, caractéristique fondamentale des architectures SDN.

Par ailleurs, les mesures effectuées à l'aide de l'outil ***iperf3*** ont montré un débit réseau satisfaisant ainsi qu'une transmission stable des données entre les différents nœuds de l'architecture. Ces résultats démontrent que la solution mise en œuvre est capable de transporter efficacement le trafic tout en maintenant de bonnes performances de communication.

Dans l'ensemble, les résultats obtenus valident l'architecture proposée et démontrent l'efficacité du modèle SDN basé sur OpenFlow pour la gestion centralisée et dynamique des flux réseau. Cette approche offre une meilleure visibilité sur le trafic, simplifie l'administration du réseau et facilite l'adaptation aux besoins évolutifs des infrastructures modernes.

3.5.2 Apport du SDN dans la gestion des flux réseau

Les expérimentations réalisées ont permis de mettre en évidence plusieurs avantages de l'approche SDN dans la gestion des flux réseau. Grâce à l'utilisation du contrôleur POX et du protocole OpenFlow, l'architecture déployée offre une gestion centralisée, flexible et dynamique des communications. Les principaux apports observés sont présentés dans les sous-sections suivantes.

3.5.2.1 Centralisation du contrôle

L'architecture mise en œuvre repose sur un modèle de contrôle centralisé assuré par le contrôleur POX exécuté sur la plateforme Intel Galileo. Contrairement aux architectures traditionnelles où chaque équipement réseau prend localement ses décisions de transfert, le contrôleur centralise l'intelligence du réseau et supervise l'ensemble des switches Open vSwitch (OVS1, OVS2 et OVS3). Cette approche permet une gestion cohérente des flux réseau et une application uniforme des politiques de contrôle au sein de la topologie déployée.

3.5.2.2 Gestion dynamique des flux

Les expérimentations réalisées ont montré que les règles OpenFlow sont installées dynamiquement par le contrôleur POX en fonction du trafic observé. Lorsqu'un paquet ne correspond à aucune entrée présente dans la Flow Table d'un switch, celui-ci génère un message *Packet-In* à destination du contrôleur. Après analyse des informations reçues, le contrôleur crée et installe les règles de forwarding appropriées à travers des messages *Flow-Mod*. Ce mécanisme réduit les traitements ultérieurs et améliore l'efficacité de l'acheminement des paquets dans le réseau.

3.5.2.3 Visibilité globale du réseau

Grâce à sa position centrale, le contrôleur dispose d'une vue globale de l'ensemble de la topologie SDN. Cette visibilité lui permet de superviser les communications entre les différents équipements, d'identifier les flux actifs et de suivre l'état du réseau en temps réel. Les captures Wireshark ainsi que l'analyse des Flow Tables ont confirmé la capacité du contrôleur à collecter les informations nécessaires pour prendre des décisions adaptées aux conditions du trafic.

3.5.2.4 Simplification de l'administration

L'utilisation du contrôleur POX a permis de simplifier considérablement l'administration de l'infrastructure réseau. Les opérations de configuration et de gestion des flux sont réalisées de manière centralisée sans nécessiter une intervention individuelle sur chaque switch Open vSwitch. Cette approche réduit les risques d'erreurs de configuration, facilite les opérations de maintenance et améliore la gestion globale du réseau.

3.5.2.5 Flexibilité et évolutivité

L'architecture déployée a démontré une bonne capacité d'adaptation aux évolutions du réseau. Grâce à l'utilisation d'OpenFlow, l'ajout de nouveaux switches ou de nouveaux hôtes peut être réalisé sans modifier profondément les mécanismes de contrôle existants. La programmabilitédu réseau offerte par le SDN facilite également l'intégration de nouvelles politiques de routage, de sécurité ou de qualité de service, ce qui constitue un avantage important pour les infrastructures réseau modernes.

3.5.3 Comparaison avec les réseaux traditionnels

Afin de mettre en évidence les avantages de l'architecture SDN proposée, une comparaison a été réalisée avec les réseaux traditionnels. Cette comparaison repose sur les résultats obtenus

lors des différentes expérimentations menées dans ce travail ainsi que sur les caractéristiques de chaque approche.

Tab. 3.1 : Comparaison avec les réseaux traditionnels

Critère	**Réseaux traditionnels**	**Architecture SDN proposée**
Plan de contrôle	Distribué dans chaque équipement réseau	Centralisé au niveau du contrôleur POX
Gestion des flux	Basée sur des règles statiques	Dynamique via le protocole OpenFlow
Configuration	Manuelle et locale	Centralisée et programmable
Supervision du réseau	Limitée à chaque équipement	Vision globale du réseau
Administration	Plus complexe	Simplifiée grâce au contrôleur
Évolutivité	Limitée	Flexible et facilement extensible
Adaptation aux changements	Lente	Rapide et dynamique
Gestion du trafic	Peu flexible	Optimisée par le contrôleur SDN

L'analyse du tableau3.1 comparatif montre que la topologie SDN mise en œuvre dans ce travail offre une gestion plus souple des communications réseau que les architectures traditionnelles. Grâce au contrôleur POX, les switches Open vSwitch ne prennent pas eux-mêmes les décisions de transfert, mais appliquent les règles OpenFlow installées dynamiquement par le contrôleur.

Les tests réalisés ont confirmé cette capacité à gérer les flux de manière centralisée. En effet, l'analyse des Flow Tables a montré l'installation automatique des règles de forwarding, tandis que les captures Wireshark ont mis en évidence les échanges OpenFlow entre le contrôleur et les switches. Cette approche permet une meilleure maîtrise du trafic réseau et simplifie les opérations de gestion et de supervision.

À l'inverse, dans un réseau traditionnel, les équipements fonctionnent de manière plus autonome et nécessitent généralement des configurations individuelles. Ainsi, la solution SDN proposée se distingue par une administration plus simple, une meilleure visibilité sur le réseau et une plus grande flexibilité dans la gestion des flux.

3.5.4 Perspectives d'amélioration

Les résultats obtenus ont permis de valider le fonctionnement de l'architecture SDN basée sur OpenFlow et de démontrer l'efficacité du contrôleur POX dans la gestion centralisée des flux réseau. Toutefois, plusieurs améliorations techniques peuvent être envisagées afin d'augmenter les performances, la flexibilité et les capacités de la solution proposée.

Dans cette étude, le contrôleur POX a été retenu en raison de sa légèreté et de sa faible consommation de ressources, ce qui le rend compatible avec la plateforme embarquée Intel Galileo utilisée pour l'hébergement du plan de contrôle. Cependant, l'utilisation d'une plateforme matérielle plus performante permettrait d'intégrer des contrôleurs SDN plus évolués tels que Ryu, ONOS ou OpenDaylight. Ces contrôleurs offrent des fonctionnalités avancées de supervision, de gestion distribuée, d'orchestration des services réseau et de traitement des statistiques OpenFlow.

Une autre amélioration consisterait à exploiter les informations collectées par les switches Open vSwitch à travers les Flow Tables et les statistiques OpenFlow. Ces données pourraient être utilisées pour développer des mécanismes de gestion dynamique du trafic capables d'adapter automatiquement les décisions de forwarding en fonction de la charge du réseau, du volume des flux ou de l'état des liens.

L'intégration de mécanismes de Qualité de Service (QoS) représente également une perspective importante. En associant les fonctionnalités OpenFlow aux politiques de QoS, il serait possible de classifier les flux réseau, de réserver des ressources pour certaines applications critiques et d'optimiser l'utilisation de la bande passante disponible.

Par ailleurs, l'ajout de fonctions avancées de sécurité basées sur le SDN, telles que le contrôle d'accès dynamique, le filtrage intelligent des paquets ou la détection d'anomalies réseau, permettrait d'améliorer la robustesse et la sécurité globale de l'infrastructure.

Enfin, l'extension de la topologie expérimentale à un plus grand nombre de switches Open vSwitch et d'hôtes réseau constituerait une étape importante pour évaluer la scalabilité de l'architecture. Une telle évolution permettrait d'étudier le comportement du contrôleur face à un volume plus important de messages OpenFlow et d'analyser les performances de la solution dans des conditions plus proches des environnements réels.

3.6 Conclusion

Dans ce chapitre, nous avons présenté la conception, le déploiement et la validation d'une architecture SDN basée sur le protocole OpenFlow. L'environnement expérimental mis en place repose sur trois switches Open vSwitch (OVS1, OVS2 et OVS3) pilotés par un contrôleur POX hébergé sur la plateforme Intel Galileo.

Les différentes étapes de configuration et de déploiement ont permis de mettre en œuvre une infrastructure SDN fonctionnelle capable d'assurer une gestion centralisée des flux réseau. Les expérimentations réalisées ont confirmé le bon fonctionnement des communications entre les équipements ainsi que l'établissement correct des échanges OpenFlow entre les switches et le contrôleur.

Les résultats obtenus à travers les tests de connectivité, l'analyse des Flow Tables, les captures réseau réalisées avec Wireshark et les mesures de performances effectuées avec iperf3 ont permis de valider l'efficacité de la solution proposée. Ils ont également mis en évidence les principaux avantages du SDN, notamment la centralisation du contrôle, l'installation dynamique des règles OpenFlow, la visibilité globale du réseau et la flexibilité dans la gestion des flux.

Enfin, l'étude réalisée a montré que l'intégration des technologies SDN constitue une approche pertinente pour répondre aux besoins croissants des réseaux modernes en matière de supervision, d'automatisation et d'optimisation du trafic. Les perspectives d'amélioration proposées ouvrent la voie à des architectures plus évolutives, plus intelligentes et capables de s'adapter à des environnements réseau de plus grande envergure.

Conclusion générale

L'objectif principal de ce travail était de concevoir et de mettre en œuvre une architecture SDN basée sur le protocole OpenFlow afin d'assurer une gestion centralisée et intelligente des flux réseau. Pour atteindre cet objectif, une infrastructure expérimentale a été déployée à l'aide de switches Open vSwitch, d'un contrôleur POX hébergé sur la plateforme Intel Galileo ainsi que de plusieurs outils de supervision et d'analyse réseau.

Dans un premier temps, une étude théorique a permis de présenter les concepts fondamentaux des réseaux définis par logiciel (SDN), le protocole OpenFlow ainsi que les principales technologies utilisées dans ce domaine. Cette étude a mis en évidence les limites des architectures réseau traditionnelles et les avantages apportés par la séparation du plan de contrôle et du plan de données.

Par la suite, l'architecture proposée a été conçue et déployée dans un environnement expérimental composé de trois switches Open vSwitch interconnectés et pilotés par un contrôleur centralisé. Les différentes étapes de configuration ont permis d'établir les communications OpenFlow entre les équipements et de mettre en œuvre un mécanisme de gestion dynamique des flux réseau.

Les tests réalisés ont confirmé le bon fonctionnement de l'infrastructure déployée. Lesvérifications de connectivité, l'analyse des Flow Tables OpenFlow, les captures réseau effectuées avec Wireshark ainsi que les mesures de performances réalisées à l'aide d'iperf3 ont permis de valider l'efficacité de la solution proposée. Les résultats obtenus ont montré que le contrôleur POX est capable d'installer dynamiquement les règles de forwarding et d'assurer une gestion centralisée des communications réseau.

L'analyse des résultats a également mis en évidence plusieurs avantages de l'approche SDN, notamment la centralisation du contrôle, la flexibilité de gestion, la visibilité globale sur le réseau et la possibilité d'adapter dynamiquement le comportement des équipements en fonction du trafic observé. Ces caractéristiques constituent des atouts importants pour lesinfrastructures réseau modernes nécessitant davantage d'automatisation, de supervision etd'optimisation.

Enfin, plusieurs perspectives d'amélioration ont été identifiées, notamment l'utilisation de contrôleurs SDN plus avancés, l'intégration de mécanismes de Qualité de Service

(QoS),l’exploitation des statistiques OpenFlow pour l’optimisation dynamique du trafic ainsi quel’extension de la topologie expérimentale à des environnements de plus grande taille.

En conclusion, les travaux réalisés ont permis de démontrer l’intérêt et la faisabilité del’approche SDN pour la gestion intelligente des flux réseau. Les résultats obtenus confirmentque cette technologie représente une solution prometteuse pour le développement des réseaux de nouvelle génération, capables d’offrir davantage de flexibilité, de contrôle et d’efficacité dans la gestion des communications.

Apports et perspectives complémentaires

Les résultats obtenus constituent une base expérimentale cohérente pour l’étude des réseaux programmables. Pour renforcer ce travail, il serait pertinent d’étendre la topologie à un nombre plus important de commutateurs, d’intégrer des politiques explicites de qualité de service, de comparer plusieurs contrôleurs SDN modernes et d’exploiter les statistiques OpenFlow pour automatiser davantage les décisions de routage. Ces perspectives permettraient de rapprocher la plateforme d’un environnement réseau opérationnel plus réaliste.

Annexe

Annexe A – Scripts et commandes de configuration

Scripts et commandes de configuration

Création et configuration des switches Open vSwitch :

OVS 1

```
# Création bridge
sudo ovs-vsctl add-br br0
# Ports
sudo ovs-vsctl add-port br0 ens38
# Controller
sudo ovs-vsctl set-controller br0 tcp :192.168.1.34:6633
# IP (Host A)
sudo ip addr flush dev ens33
sudo ip addr add 192.168.1.50/24 dev br0
# Up interfaces
sudo ip link set br0 up
sudo ip link set ens38 up
```

OVS 2

```
# Création bridge
sudo ovs-vsctl add-br br0
# Ports
sudo ovs-vsctl add-port br0 ens37
# Controller
sudo ovs-vsctl set-controller br0 tcp :192.168.1.34:6633
# IP (Host B)
sudo ip addr flush dev ens33
```

sudo ip addr add 192.168.1.51/24 dev br0

Up interfaces

sudo ip link set br0 up

sudo ip link set ens38 up

OVS 3

Création bridge

sudo ovs-vsctl add-br br0

Ports

sudo ovs-vsctl add-port br0 ens37

sudo ovs-vsctl add-port br0 ens38

Controller

sudo ovs-vsctl set-controller br0 tcp :192.168.1.34:6633

Up interfaces

sudo ip link set br0 up

sudo ip link set ens37 up

sudo ip link set ens38 up

Lancement du contrôleur POX

ssh root@192.168.1.34

cd ~/pox

./pox.py log.level --DEBUG forwarding.l2_learning

ANNEXE B

Etapes d’installation des outils de travail

1. Installation VMware Worstation Pro 17

Télécharger le depuis ce site

https ://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html Après installation voici l’interface de logiciel

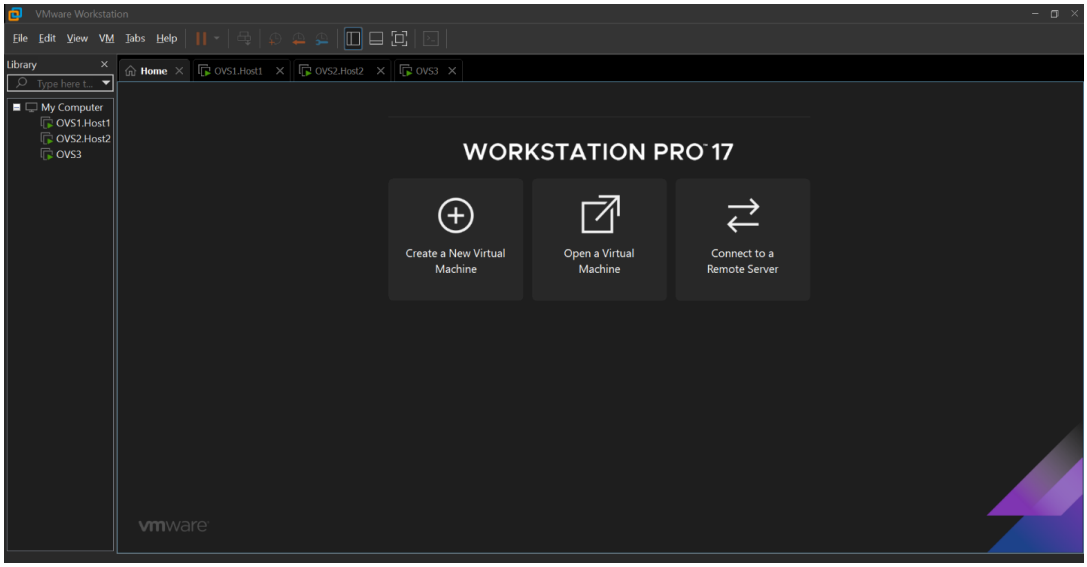


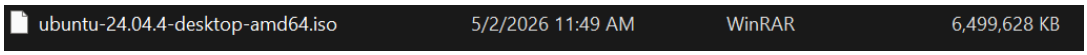
Figure.B.1 : Interface VMware Workstation pro17

2. Installation machine virtuelle Ubuntu

L'image de la VM Ubuntu est disponible dans ce site pour téléchargement

https ://releases.ubuntu.com/16.04/

La version utilisée est la suivante



Allez sur VMware Workstation pro et ajouter une nouvelle machine virtuelle



Figure.B.2 : Etape d'installation de VM Ubuntu

Sélectionnez "Linux" comme type d'OS et "Ubuntu-64bit" comme version de l'OS. Configurez la quantité de mémoire vive et d'espace de stockage allouée à la machine virtuelle Et Sélectionnez l'emplacement du fichier image ISO téléchargé à l'étape 1 comme disque d'installation.

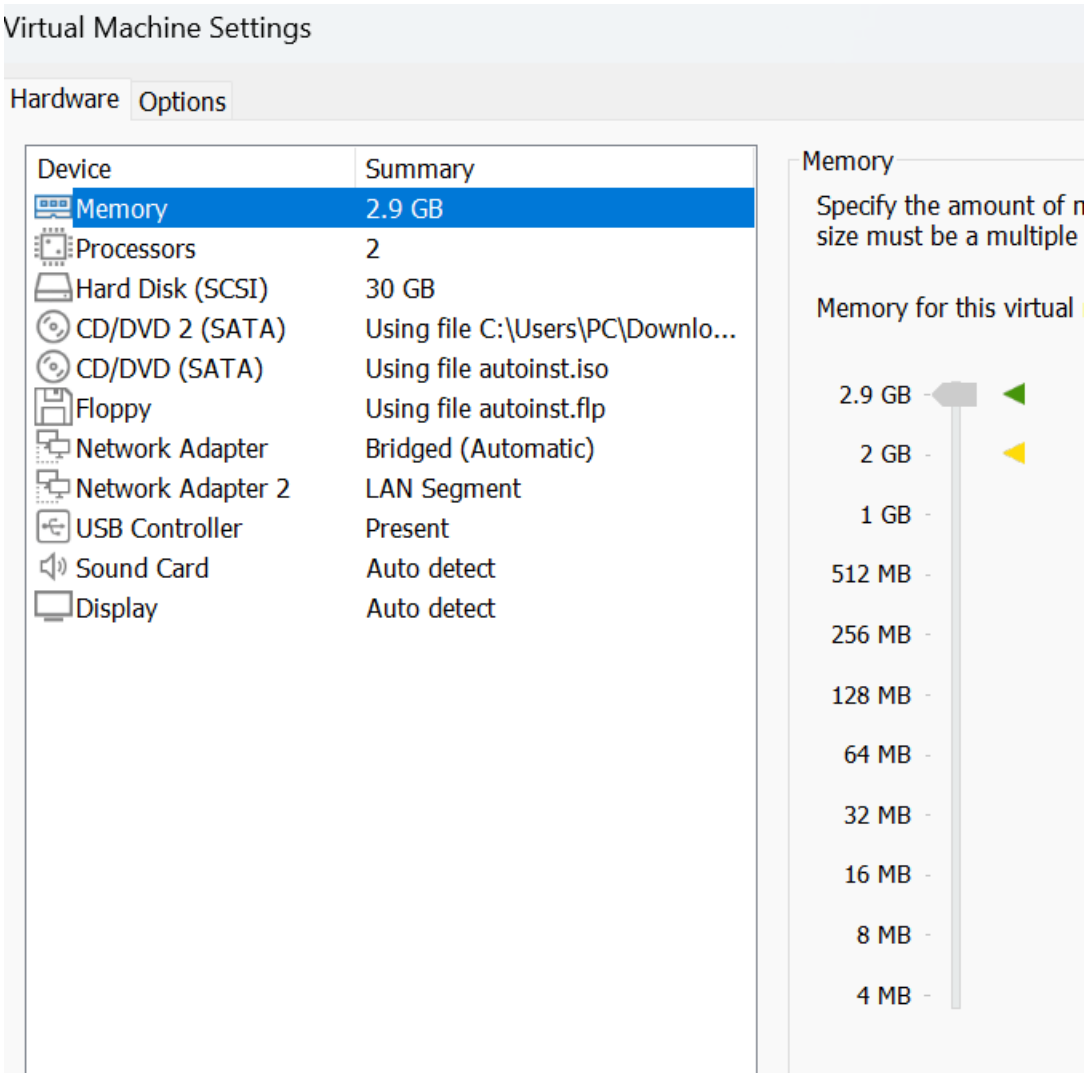


Figure.B.3 : Les paramètres à configuré dans les VM Ubuntu

Ensuite allumé la machine en cliquons sur Power on thisvirtual machine et commencé l'installation Voici l'interface de VM Ubuntu utilisé (Figure.B.4) :

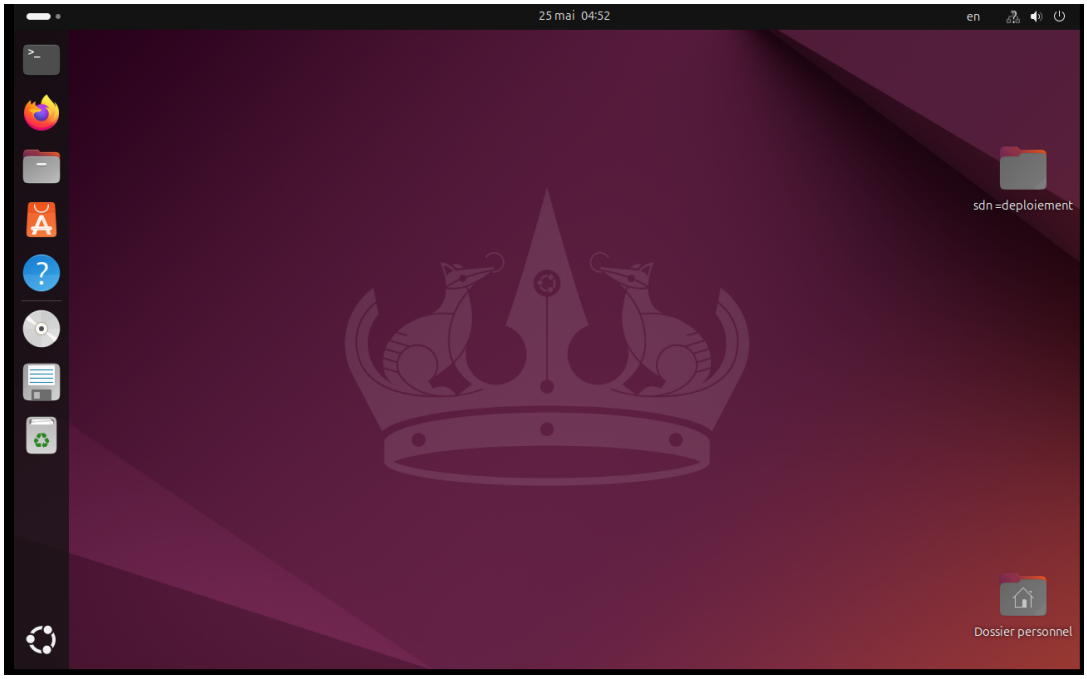


Figure.B.4 : Interface machine virtuel Ubuntu

Installation Wireshark

Dans les OVSwitch va faire les étapes suivantes

Entrer dans le terminal la commande suivante

Mettez à jour les informations du package en exécutant la commande suivante :

```
File  Edit  View  Search  Terminal  Help
floodlight@floodlight:~$ sudo apt-get update
Err http://security.ubuntu.com trusty-security InRelease

Err http://security.ubuntu.com trusty-security Release.gpg
  Could not resolve 'security.ubuntu.com'
Err http://extras.ubuntu.com trusty InRelease

Err http://extras.ubuntu.com trusty Release.gpg
  Could not resolve 'extras.ubuntu.com'
Err http://us.archive.ubuntu.com trusty InRelease

Err http://us.archive.ubuntu.com trusty-updates InRelease

Err http://us.archive.ubuntu.com trusty-backports InRelease

Err http://us.archive.ubuntu.com trusty Release.gpg
  Could not resolve 'us.archive.ubuntu.com'
Err http://us.archive.ubuntu.com trusty-updates Release.gpg
  Could not resolve 'us.archive.ubuntu.com'
Err http://us.archive.ubuntu.com trusty-backports Release.gpg
  Could not resolve 'us.archive.ubuntu.com'
Reading package lists... 69%
```

Figure.B.5 : Etape d’installation de Wireshark

Installez Wireshark en exécutant la commande suivante :

Sudoapt-getinstallwireshark

Lorsque l'installation est terminé vous pouvez lancer Wireshark en exécutant la commande suivante :**Sudowireshark**

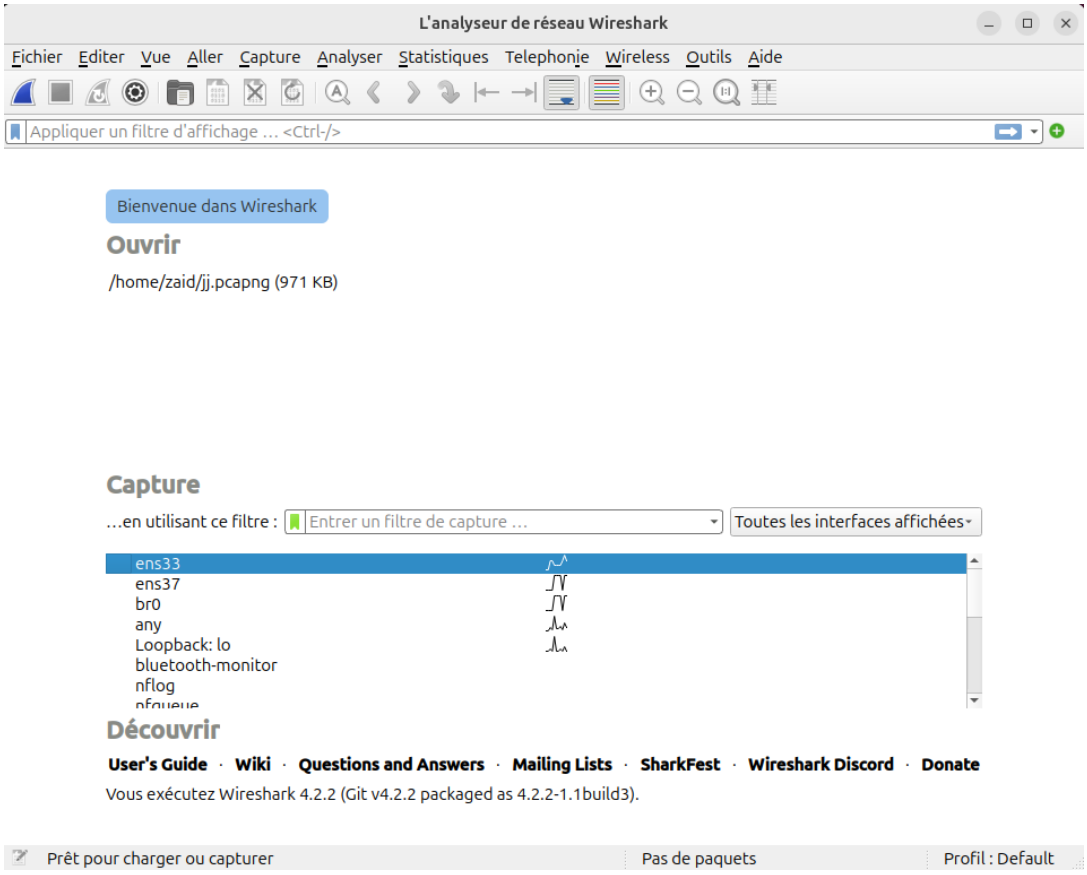


Figure.B.6 :Interface de Wireshark

Installation Iperf

Télécharger dans VM Ubuntu iperf depuis le lien suivant :**https ://iperf.fr/iperfdownload.php#ubuntu**

Ouvrez le terminal sur votre machine et tapez les commandes suivantes :

Sudo apt-get update

Sudo apt-get install iperf

Bibliographie

[1] Kurose, J. F. et Ross, K. W. *Computer Networking : A Top-Down Approach*. Pearson, 2017.

[2] McKeown, N. et al. *OpenFlow : Enabling Innovation in Campus Networks*. ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

[3] Kreutz, D. et al. *Software-Defined Networking : A Comprehensive Survey*. Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[4] ITU-T. *Definitions of Terms Related to Quality of Service*. Recommendation E.800, International Telecommunication Union, 2008.

[5] Tanenbaum, A. S. et Wetherall, D. J. *Computer Networks*, 5e éd. Boston, MA : Pearson Education, 2011.

[6] Schulzrinne, H. et al. *RTP : A Transport Protocol for Real-Time Applications*. RFC 3550, 2003.

[7] Stallings, W. *Data and Computer Communications*, 10e éd. Pearson, 2013.

[8] Keshav, S. *An Engineering Approach to Computer Networking*. Addison-Wesley, 1997.

[9] Postel, J. *Internet Protocol*. RFC 791, 1981.

[10] Nunes, B. A. A. et al. *A Survey of Software-Defined Networking : Past, Present, and Future of Programmable Networks*. IEEE Communications Surveys & Tutorials, 2014.

[11] Sobrinho, J. L. *Algebra and Algorithms for QoS Path Selection*. IEEE/ACM Transactions on Networking, 2002.

[12] Armbrust, M. et al. *A View of Cloud Computing*. Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[13] Qin, Z. *Software-defined Networking for Smart Grid Communications : A Comprehensive Survey*. IEEE Communications Surveys & Tutorials, 2014.

[14] Open Networking Foundation. *Software-Defined Networking : The New Norm for Networks*. White Paper, 2012.

[15] Sezer, S. et al. *Are We Ready for SDN ? Implementation Challenges for Software-Defined Networks*. IEEE Communications Magazine, vol. 51, no. 7, pp. 36–43, 2013.

[16] Lantz, B., Heller, B. et McKeown, N. *A Network in a Laptop : Rapid Prototyping for Software-Defined Networks*. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010.

[17] Open Networking Foundation. *OpenFlow Switch Specification*, Version 1.0.0, 2009.

[18] McCauley, J. *POX : A Python-based OpenFlow Controller*, 2012.

[19] Rowshanrad, S. et al. *A Survey on SDN Controllers : A Comparative Study*. 8th International Conference on Information Technology, 2016.

[20] VMware. *VMware Workstation Documentation*. VMware Inc.

[21] Canonical Ltd. *Ubuntu Documentation*.

[22] Open vSwitch Project. *Open vSwitch Documentation*.

[23] NOXRepo Project. *POX SDN Controller Documentation*.

[24] Wireshark Foundation. *Wireshark User Guide*.

[25] ESnet. *iperf3 Documentation*.