الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département : Génie électrique Et Informatique Industrielle

Mémoire de fin d'étude en vue de l'obtention du diplôme de Master en Télécommunications

Spécialité : Système de Télécommunications et Réseaux

- Thème –

Contribution à l'étude de la solution SDN

Réalisé par :

M. Bentegar Sid Ahmed Abdelillah

M. Halouane Alla Eddine

Les membres de Jury :

Mme. BENDOUDA Djamila M.C.B	Présidente
M. KHOUMERI El Hadi M.C.B	Promoteur
Mme. ABBAD Leila M.A.A	Examinateur
M. BOUCHACHI Islam M.C.B	Examinateur

Alger, le 26/06/2022

Année universitaire : 2021 –2022

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires pour mener ce travail à son terme.

Et en second lieu , on remercie nos parents pour leurs efforts pour rester confortable et heureux et pour obtenir une capacité suffisant de compléter ce travail,troisièmement on remercie nos frères de leurs encouragements toute l'année et tout le temps.

Et finalement on remercie notre promoteur M. KHOUMERI El Hadi pour leur disponibilité et on remercie aussi tous les professeurs de cette promotion et de tous les années passées pour nous donner toutes les informations dans nos domaines .

Table des matières

Remerciements . . . . . **I**

1 Introduction **1**

Introduction **1**
1.1 Présentation des chapitres 1

2 Historique **2**
2.1 Introduction 2
2.2 Open Signaling 2
2.3 Active Networking 2
2.4 DCAN 3
2.5 ForCES 3
2.5.1 Élément de transfert 3
2.5.2 Élément de contrôle 4
2.5.3 Élément de réseau : 4
2.5.4 Le protocole ForCES[1] : 4
2.6 4D Project 4
2.7 Netconf 4
2.8 Ethane 4
2.8.1 Le réseau doit être régi par des politiques de haut niveau 5
2.8.2 Le routage pour le réseau doit être au courant de ces politiques . . 5
2.8.3 Le réseau doit imposer la liaison entre les paquets et leur origine . 5
2.9 Conclusion 5

3 Software Defined Network (SDN) **7**

Software Defined Network (SDN) **7**
3.1 Introduction 7
3.2 Définition du SDN 7
3.3 Les avantages du SDN 7
3.3.1 Réduction de coûts 8
3.3.2 Extensibilité 8
3.3.3 Outil de gestion de réseau central 8
3.3.4 Isolation et contrôle du trafic 8
3.4 Limites des réseaux classiques 8
3.4.1 Complexité de gestion 8

[1] Forwarding and Control Element Separation

3.4.2 Difficulté à appliquer les politiques de réseau . . . 9
3.4.3 Problèmes d'évolutivité . . . 9
3.5 Application de SDN . . . 9
3.5.1 Équilibrage des charges pour les serveurs d'applications . . . 9
3.5.2 Mise à niveau des centres de données . . . 10
3.5.3 Simplicité la gestion de Réseaux de campus . . . 10
3.5.4 Sécurité des centres de données . . . 10
3.6 Architecture du SDN . . . 10
3.6.1 Plan de données . . . 11
3.6.2 Plan de contrôle . . . 11
3.6.3 Plan d'application . . . 11
3.7 Comparaison entre les réseaux traditionnels et le SDN . . . 12
3.8 Défis du SDN . . . 13
3.8.1 Evolutivité . . . 13
3.8.2 Tolérance aux pannes . . . 14
3.8.3 Sécurité . . . 14
3.8.4 Flexibilité . . . 17
3.9 Conclusion . . . 18

4 Application . . . 19
4.1 Introduction . . . 19
4.2 logiciels utilisés . . . 19
4.2.1 Mininet . . . 19
4.2.2 OpenDaylight . . . 19
4.3 L'implantation de l'application . . . 19
4.4 Conclusion . . . 25

Conclusion . . . 26

ملخص . . . 30

Résumé . . . 31

Abstract . . . 32

Table des figures

2.1 Communication sur un réseau Ethane [5] 5

3.1 L'architecture SDN [13] 11
3.2 Les attaques potentielles de l'architecture SDN 15
3.3 Externalisation vers un fournisseur de fonctionnalités dans le SDN 17

4.1 Lancement de contrôleur 21
4.2 Lancement du la topologie 21
4.3 Visualisation du réseau 22
4.4 Test de Ping entre les hôtes et serveur 23
4.5 La connexion de l'administrateur au serveur a échoué au port 80. 24
4.6 le code de deuxième flux 25

Liste des tableaux

3.1 les caractéristique de sdn et de l'architecture de réseau traditionnelle [10] . 13

Liste des sigles et acronymes

SDN	*Softwate Defined Network*
DCAN	*Devolved Control ATM Networks*
ForCES	Forwarding and Control Element Separation
XML	Extensible Markup Language
API	*Application Programming Interface*
REST	*Representational State Transfer*
HTTP	*Hypertext Transfer Protocol*
ARP	*Address Resolution Protocol*
ICMP	*Internet Control Message Protocol*
OSPF	*Open Shortest Path First*
STP	*Spanning Tree Protocol*
DoS	*Denial-of-Service*
DDoS	*Distributed Denial-of-Service*
GNS3	*Graphical Network Simulator-3*
SVG	*Scalable Vector Graphics*
ONF	*Open Network Foundation*
OXM	*OpenFlow Extensible Match*
ATM	*Asynchronous Transfer Mode*
ACL	*Access-list*

Chapitre 1

Introduction

Dans les entreprises, le réseau est devenu une chose importante dans la vie professionnelle, car il facilite grandement la conduite de certains travaux grâce aux applications qui fonctionnent grâce à l'infrastructure réseau.
Cette structure se compose des équipements qui fonctionnent à leur tour pour communiquer et envoyer des paquets entre ordinateurs et serveurs, et comme nous le savons, les ordinateurs se développent de manière remarquable, il en va de même les infrastructures réseau pour traiter les énormes quantités de données générées par les ordinateurs.
Et quand on regarde notre réseau actuel , on s'aperçoit qu'il est encore d'une peu mauvaise qualité, qu'il manque de développement et qu'il est également sujet aux erreurs. C'est pourquoi il est apparu de la programmation réseau, notamment "Software-defined networking(SDN)", qui a su se faire remarquer dans le domaine des réseaux grâce à ses avantages et sa solution à un grand nombre de problèmes .
Notre objectif dans ce travail est de clarifier la solution SDN avec son historique, ses avantages, ses domaines d'utilisation et les défis auxquelles elle est confrontée.

1.1 Présentation des chapitres

Chapitre 1
Ce chapitre est considéré comme une introduction de notre mémoire de master.

Chapitre 2
Dans ce chapitre, nous allons parler de l'histoire du SDN et de ses prédécesseurs.

Chapitre 3
Dans ce chapitre, nous allons parler des aspects fondamentaux de la technologie SDN ainsi que leurs applications et défis.

Chapitre 4
Dans ce chapitre un va réaliser une implémentation d'ACL[1] dans un réseau SDN

[1] *Access-list*

Chapitre 2

Historique

2.1 Introduction

Dans ce chapitre nous discuterons des étapes de l'évolution du SDN depuis les années 1990. Il y a eu plusieurs projets pour atteindre les objectifs voulus, tels que rendre les réseaux programmables. On montrera les objectifs voulus, tels que rendre les réseaux programmables. On montrera les objectifs ainsi que les obstacles rencontrés.

2.2 Open Signaling

Une série d'ateliers appelés Open Signaling (OPENSIG) a débuté en 1995 dans le but de rendre les réseaux ATM[1] plus ouverts, programmables et évolutifs. L'objectif fondamental de ce projet est de fournir un accès aux commutateurs et aux routeurs via un ensemble d'interfaces réseau programmables ouvertes, ce qui permet le développement de nouveaux services et architectures de réseau programmables [4]. Le problème qu'ils ont eu avec ce projet était que les routeurs et les commutateurs étaient entièrement intégrés, c'est pourquoi ils ont trouvé difficile, voire presque impossible de mettre en œuvre ce projet . [4].

2.3 Active Networking

Il s'agit d'une technologie proposée dans les années 1990 pour construire des infrastructures de réseau basées sur des interfaces ouvertes et programmables, où les logiciels peuvent être dynamiquement déployés dans des nœuds de commutateurs ou routeurs [8]. Dans cette proposition, il existe deux méthodes de base[8] :

[1] *Asynchronous Transfer Mode*

Méthode de nœud programmable

Les programmes sont injectés dans des nœuds actifs programmables séparément des paquets de données réels traversant le réseau, l'utilisateur envoie le programme à un nœud du réseau (commutateur ou routeur), et lorsque les données arrivent au nœud, le programme sera stocké et exécuté .

Methode d'encapsulation

Le programme est intégré dans chaque paquet de données qui appelle la capsule, cette dernière contenant une partie du programme pouvant également contenir des données intégrées. Lorsqu'une telle capsule atteint le nœud actif, l'environnement d'exécution interprète le programme et prend des mesures en fonction de l'interprétation du programme.

L'inconvénient de cette technologie est le manque de sécurité [8], mais malgré ce défaut, cette technologie laissait espérer le développement de réseaux programmables similaires aux avantages de cette technologie .

2.4 DCAN

L'objectif du projet DCAN[2] est de concevoir et développer l'infrastructure nécessaire au contrôle et à la gestion évolutive des réseaux multiservices[7]. Le principe de DCAN est que les fonctions de contrôle/gestion des différents appareils qui composent un tel réseau (généralement des commutateurs ATM) doivent être extraites des appareils eux-mêmes et être déléguées à des postes de travail externes dédiés à cet effet[7].

2.5 ForCES

Le protocole ForCES définit un protocole standard et un mécanisme d'échange d'informations pour séparer les fonctions du plan de contrôle (telles que les protocoles de routage, les protocoles de signalisation) à partir du plan de transfert de données des routeurs IP et des dispositifs similaires[6].

Les éléments opérationnels de ForCES sont les suivants [9] :

2.5.1 Élément de transfert

L'élément de transfert est généralement implémenté dans le matériel et situé dans le réseau. Ce dernier est chargé de transmettre et de filtrer les règles qu'il reçoit du contrôleur .

[2] *Devolved Control ATM Networks*

2.5.2 Élément de contrôle

L'élément de contrôle est concerné par la coordination entre les différents dispositifs du réseau et par la communication des informations de transfert et de routage aux éléments de transfert .

2.5.3 Élément de réseau :

L'élément de réseau est le dispositif de réseau réel qui se compose d'un ou plusieurs éléments de transfert et d'un ou plusieurs d'élément de contrôle.

2.5.4 Le protocole ForCES :

Ce protocole est utilisé pour les communications entre les éléments de contrôle et les éléments de transfert dans un élément de réseau ForCES.

2.6 4D Project

Depuis le milieu des années 2000, le projet 4D a été lancé dans le but de prendre toutes les décisions de contrôle et de gestion de manière logiquement centralisée et séparée, contenant quatre plans : Plan de décision, Plan de diffusion, Plan de découverte, Plan de données[6].

2.7 Netconf

En 2006, le protocole NETCONF a été lancé comme une étape importante vers des systèmes de gestion de réseau basés sur XML[3]. Il s'agit d'un protocole de gestion qui utilise des API[4] pour définir des processus de gestion des équipements réseaux par laquelle les donnes de configuration peuvent être traitées et récupérées. Peut utiliser aussi ce protocole pour la gestion des pannes [24]. Bien que netconf soit une évolution de la mise en réseau mais pas considéré comme entièrement programmable car ce protocole ne sépare pas le plan de contrôle du plan de données[24].

2.8 Ethane

En 2006 ethane a été proposé dans le cadre de la création d'une solution basée sur des règles qui permet aux administrateurs de réseau de définir des règles relatives à l'accès des utilisateurs au niveau du réseau, ce qui inclut l'authentification et l'isolement des utilisateurs ce qui inclut l'authentification et l'isolement des utilisateurs comme illustré à la Figure 1.1 [5]. Ethane est construit autour de trois principes fondamentaux[5] :

[3]Extensible Markup Language

[4]*Application Programming Interface*

2.8.1 Le réseau doit être régi par des politiques de haut niveau

Ethane épouse l'idée que le réseau doit être régi par des politiques définies à des niveaux élevés plutôt que sur une base par appareil. Ces politiques devraient se situer au niveau des services et des utilisateurs, ainsi que des machines par les quelles les utilisateurs peuvent se connecter au réseau.

2.8.2 Le routage pour le réseau doit être au courant de ces politiques

Les chemins empruntés par les paquets à travers le réseau doivent être dictés par les politiques de niveau supérieur décrites dans le point précédent plutôt que dans le cas des réseaux actuels, dans lesquels les chemins sont choisis en fonction de directives de niveau inférieur.

2.8.3 Le réseau doit imposer la liaison entre les paquets et leur origine

Si les décisions stratégiques reposent sur des concepts de plus haut niveau, tels que le concept d'utilisateur, les paquets circulant dans le réseau doivent pouvoir être retracés jusqu'à leur point d'origine (c'est-à-dire l'utilisateur ou la machine qui est la source réelle du paquet).

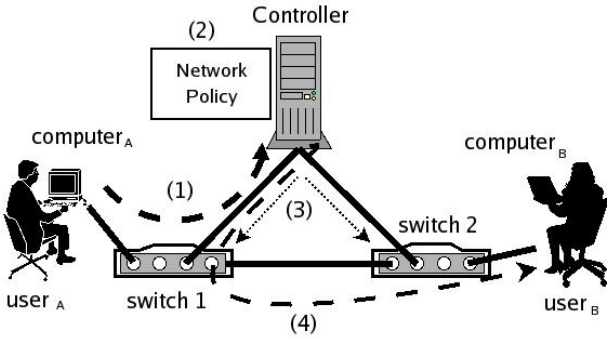


Fig. 2.1 : Communication sur un réseau Ethane [5]

2.9 Conclusion

Nous concluons dans ce chapitre que depuis les années 1990, les chercheurs en réseau ont envisagé d'optimiser le fonctionnement du réseau en séparant les plans d'équipement de l'infrastructure du réseau, mais ils ont dû faire face à plusieurs facteurs qui ont rendu cet objectif difficile à atteindre. Lorsque le projet d'éthane est apparu, Cela a donné beaucoup d'espoir pour atteindre l'objectif car l'éthane met en œuvre l'idée que les politiques sont implémentées à un niveau élevé, cela a affecté la séparation des plans d'infrastructures du

réseau de tel sorte que c'est le contrôleur qui a le droit de décider sur la transmission des paquets, en plus de la création des basses sur laquelle SDN s'appuyait.

Chapitre 3

Software Defined Network (SDN)

3.1 Introduction

SDN propose une nouvelle architecture, qui sépare les parties de contrôle du réseau et les fonctions d'acheminement, cette séparation est une rupture avec l'architecture traditionnelle. Grâce à cette séparation, nous pouvons isoler les tâches complexes de la tâche répétitive de transfert. La tâche complexe peut être automatisée et gérée séparément dans un contrôleur SDN centralisé ou un système d'exploitation de réseau. Dans ce chapitre, nous présentons les principaux aspects de SDN et les domaines de l'utilisation de cette technique et les limites des réseaux traditionnels, et la nouveauté que SDN a apportée par rapport au réseau conventionnel, les défis qui rencontrent le SDN .

3.2 Définition du SDN

Le Software-Defined Networking est une architecture de réseau qui sépare le plan de contrôle et le plan des données, et qui permettre la centralisation de toute procédure du plan de contrôle dans un contrôleur SDN, cela permet de simplifier les nœuds de réseaux et optimiser les performances du plan des donnes[20]. Cette architecture permet aux développeurs d'implémenter de nouveaux algorithmes à placer dans le contrôleur qui capable de gérer le réseau et de modifier sa fonctionnalité [20].

3.3 Les avantages du SDN

Les réseaux définis par logiciel vont changer la façon dont les ingénieurs et les concepteurs de réseaux construisent et exploitent leurs réseaux pour répondre aux besoins des entreprises. Avec le principe du SDN, les réseaux sont devenus des modèles standard, non propriétaires, et faciles à programmer et à gérer. Le SDN permettra aux entreprises et aux opérateurs de mieux contrôler leurs réseaux, de les adapter et de les optimiser afin de réduire le coût global d'exploitation du réseau. Certains des principaux avantages du SDN peuvent être résumés comme suit [19] :

3.3.1 Réduction de coûts

Le SDN prend en charge les modèles de réseau de la couche 1 à la couche 3 sans qu'il soit nécessaire d'acheter un équipement réseau coûteux. En d'autres termes, l'utilisation du SDN dans un environnement de production peut aider à réduire les coûts associés à l'achat de matériel coûteux[19].

3.3.2 Extensibilité

En utilisant le SDN, il est facile d'utiliser API's pour étendre les capacités de SDN en développant des applications pour contrôler le comportement du trafic réseau[19].

3.3.3 Outil de gestion de réseau central

Les administrateurs réseau ont souvent du mal à gérer la configuration des routeurs physiques, et lorsque plusieurs routeurs physiques doivent être gérés, cela peut rapidement devenir long et fastidieux. SDN simplifie la gestion des routeurs physiques en fournissant des API de gestion dans la console de contrôleur SDN[19].

3.3.4 Isolation et contrôle du trafic

Les administrateurs de centres de données peuvent bénéficier d'un contrôle réseau centralisé à l'aide de contrôleurs SDN, tandis que SDN fournit plusieurs mécanismes d'isolation, tels que la configuration d'ACL et de pare-feu sur des réseaux de machines virtuelles au niveau de la carte. Les règles de trafic peuvent également être définies à l'aide de la console, assurant un contrôle complet du trafic réseau[19].

3.4 Limites des réseaux classiques

Les architectures de réseau traditionnelles existantes ne peuvent pas répondre aux besoins actuels des utilisateurs finaux, des fournisseurs de services et des entreprises ; certaines de leurs limites sont [16] :

3.4.1 Complexité de gestion

Les technologies de réseau informatique précédentes étaient basées sur un ensemble de protocoles de routage conçus pour connecter les hôtes de manière fiable sur de longues distances, à des vitesses élevées et selon différentes conceptions de réseau. Afin de répondre aux exigences de l'industrie telles que la haute disponibilité, la sécurité et la connectivité étendue, au cours des dernières années.Mais cette approche de conception conduit à l'un des problèmes majeurs auxquels sont confrontés les administrateurs réseau aujourd'hui, la complexité de la gestion du réseau. Par exemple, ajouter ou modifier localisation d'un équipement sur un réseau est devenu un lourd travail pour les administrateurs de réseau,

qui doivent reconfigurer plusieurs parties du réseau, comme les listes d'accès, les VLAN, les protocoles de routage et les topologies de réseau.

En plus de ce qui précède, il faut tenir compte de la compatibilité des fournisseurs d'équipements et des versions de logiciels avant d'apporter toute modification au réseau. Par conséquent, les administrateurs de réseau gardent leur réseau plutôt statique, afin d'éviter ou de minimiser les interruptions de service qui peuvent être causées par tout changement.

3.4.2 Difficulté à appliquer les politiques de réseau

Pour maintenir la politique réseau de l'entreprise, les administrateurs réseau doivent configurer des centaines de routeurs, de commutateurs et de mécanismes. Dans un domaine virtualisé, chaque fois qu'un service informatique ajoute une machine virtuelle au réseau prend des heures ou des jours, pendant lesquels les administrateurs réseau doivent configurer et ajuster les ACL pour l'ensemble du réseau. Avec la complexité des réseaux actuels, il est difficile pour les administrateurs réseau de maintenir un cadre cohérent pour les droits d'accès, la qualité de service et la sécurité.

3.4.3 Problèmes d'évolutivité

En ce moment, les demandes pour un centre de données augmentent, et en même temps le réseau doit croître à la même vitesse. Mais en réalité, les réseaux sont devenus extrêmement complexes en raison de l'ajout de centaines ou voire des milliers de routeurs, de commutateurs et même de pare-feu qui doivent être gérables et configurables.

Les administrateurs réseau ont toujours beaucoup compté sur l'augmentation de la bande passante pour faire évoluer les réseaux d'entreprise, mais avec la virtualisation des centres de données, les modèles de trafic sont devenus très dynamiques, ce qui rend difficile la prévision des modèles de trafic.

Les grands opérateurs, comme Amazon, Ebay et Facebook, sont confrontés à des problèmes d'évolutivité encore plus difficiles. Il est impossible de configurer manuellement de tels réseaux à grande échelle.

3.5 Application de SDN

3.5.1 Équilibrage des charges pour les serveurs d'applications

La fonction d'équilibrage de charge entre plusieurs serveurs est mise en œuvre par des équipements dédiés installés dans le réseau. Malgré l'utilisation du SDN, les commutateurs OpenFlow peuvent gérer automatiquement cette fonction et distribuer le trafic vers différents serveurs. Mais il n'est pas très évolutif ; il est donc possible d'écrire une application qui fonctionne au-dessus du contrôleur et qui peut fournir une application d'équilibrage de charge évolutive et efficace [23].

3.5.2 Mise à niveau des centres de données

Les centres de données sont une composante interne importante de nombreuses grandes entreprises qui disposent d'un grand nombre de centres de données pour gérer une demande massive et répondre rapidement. Ces centres de données sont extrêmement coûteux et compliqués à entretenir et à gérer. SDN et OpenFlow permettent aux entreprises de réduire les coûts d'installation et de configuration du centre de données, car la partie du réseau qui transporte les données peut être gérée à partir d'un contrôleur [12].

3.5.3 Simplicité la gestion de Réseaux de campus

Les réseaux de campus peuvent être difficiles à gérer, notamment en raison du besoin constant d'unifier les réseaux Wi-Fi et Ethernet. Les contrôleurs SDN peuvent profiter aux réseaux de campus en fournissant une gestion et une automatisation centralisées, une sécurité renforcée et une qualité de service au niveau des applications sur le réseau [17].

3.5.4 Sécurité des centres de données

Les entreprises utilisent des pares-feux traditionnels pour protéger leurs centres de données, d'autre part, les SDN peuvent fournir une protection efficace et simplifier la gestion des pare-feu grâce à un contrôle et une automatisation centralisée, ce qui permet également aux administrateurs de visualiser, de modifier et de surveiller l'activité réseaux pour réduire les risques [17].

3.6 Architecture du SDN

Une représentation typique d'une architecture SDN se compose de trois couches : la couche d'application, la couche de contrôle et la couche de données comme le montre figure 3.1 . Ces couches communiquent via des interfaces de programmation d'application (API) nord et sud .

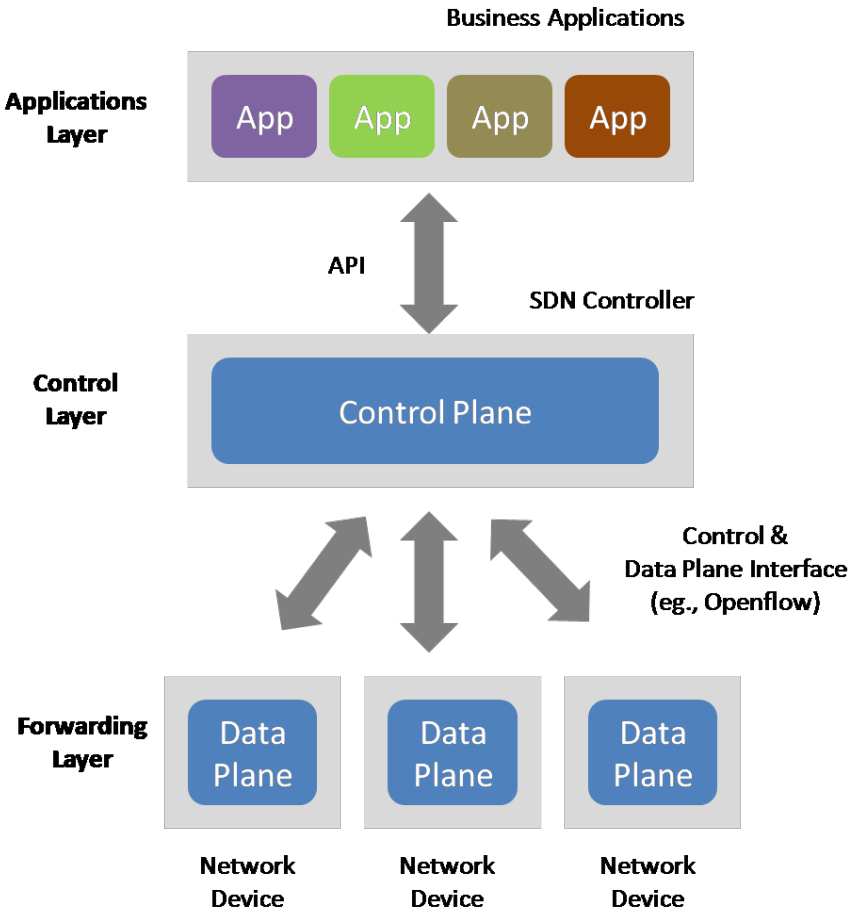


FIG. 3.1 : L'architecture SDN [13] .

3.6.1 Plan de données

Est constitué d'un ensemble de plusieurs équipements de réseau (switch, router), chacun d'entre eux contenant un ensemble de ressources de transfert ou de traitement du trafic des paquets [2].

3.6.2 Plan de contrôle

Le plan de contrôle est le cerveau des réseaux qui contiennent un ensemble de contrôleurs SDN, chacun représentant un contrôle central qui gère les politiques et les flux de trafic sur l'ensemble du réseau [2].

3.6.3 Plan d'application

Il s'agit d'applications qui s'exécutent au-dessus du réseau, telles que les équilibreurs de charge, les pares-feux et la surveillance et la gestion du réseau. Les outils de communication de base entre les couches mentionnées sont fournis via des interfaces de programmation d'application (API)[2].

Northbound API :

Dans l'infrastructure de réseaux, le plan de contrôle est une structure de gestion qui vérifie et gère le comportement du réseau.

Southbound API :

Facilite un contrôle efficace du réseau et permet aux contrôleurs SDN d'apporter des modifications en fonction des besoins et des exigences en temps réel. Celui-ci est chargé d'établir la communication entre le plan de contrôle et le plan de données [21].

3.7 Comparaison entre les réseaux traditionnels et le SDN

Les réseaux traditionnels sont implémentés dans le plan de contrôle du paradigme distribué pour des protocoles comme OSPF[1] ARP[2] STP[3], Ces équipements sont connectés mais il n'y a pas une machine qui gère les réseaux [11].

- La différence fondamentale est que les réseaux conventionnels sont basés sur le matériel alors que le SDN est généralement basé sur le logiciel[11]. En plus de ça , ce dernier est polyvalent ce qui permette aux administrateurs de mieux contrôler et de manipuler facilement les ressources à distance dans le plan de contrôle[11].
- Les réseaux conventionnels utilisent le principe de combiner le plan de contrôle et les données dans une unité physique, lorsque sa capacité est partagée, la charge de trafic et la charge sur le CPU et la RAM augmentent., pour le SDN les plans peuvent être facilement surveillés et gérés par le contrôleur et permettre au réseau de mieux se configurer avec une charge de trafic moindre, en séparant ces processus et en disposant d'un serveur dédié[11].
- SDN permet aux responsables informatiques de fournir des services d'infrastructure physique et une bande passante supplémentaire sans investissement. Les réseaux traditionnels nécessitent un nouveau matériel pour étendre les capacités du réseau[11].
- Les principales différences entre l'architecture de réseau traditionnelle et l'architecture SDN sont présentées dans le tableau 3.1.

[1] *Open Shortest Path First*
[2] *Address Resolution Protocol*
[3] *Spanning Tree Protocol*

Caractéristiques	SDN	Réseau Traditionnel
Contrôle du réseau centralisé	✓	×
Programmabilité	✓	×
Flexibilité du réseau	✓	×
Réseau de contrôle complexe	×	✓
Amélioration des performances	✓	×
Configuration of Error-Prone	×	✓
Gestion améliorée	✓	×
Efficacité de la configuration	✓	×
Facile à utiliser et à mettre en œuvre	✓	×

TAB. 3.1 : les caractéristique de sdn et de l'architecture de réseau traditionnelle [10]

3.8 Défis du SDN

Bien que l'architecture SDN aide à résoudre de nombreux problèmes auxquels sont confrontées les infrastructures réseau actuelles, elle est encore en cours de développement[11]. Comme expliqué dans le paragraphe précédent, il existe de nombreux avantages, plusieurs applications et plusieurs domaines qui peuvent y être implémentés, mais il existe également divers défis qui nécessitent une grande attention. Ces défis seront expliqués dans cette section.

3.8.1 Evolutivité

Le contrôleur central peut gérer 30 000 requêtes par seconde[3]. Mais cela ne veut pas dire que ce sera suffisant, surtout si le flux arrive à un rythme rapide et qu'il y a des changements rapides dans le réseau[1]. Ces demandes peuvent submerger l'unité centrale de contrôle et donc réduire la réponse du contrôleur, ce qui entraîne un mauvais travail dans les nœuds du réseau et donc une congestion se produit dans le transfert des paquets, et en plus si le réseau s'étend en augmentant les commutateurs, il peut cesser de fonctionner à partir de nombreuses demandes[1].

Pour résoudre ce problème il existe deux solutions, la première consiste à repartir la logique de contrôle entre le contrôleur et les commutateurs, cependant, une telle solution nécessite des modifications dans la conception des commutateurs OpenFlow et va à l'encontre des principes de base du paradigme SDN[1].

Une autre possibilité que la communauté des chercheurs considère comme une solution efficace consiste à façonner le plan de contrôle de manière à atténuer les problèmes d'évolutivité et de fiabilité, notamment grâce à l'utilisation de contrôleurs SDN physiquement distribués[1].

3.8.2 Tolérance aux pannes

Comme mentionné précédemment, SDN se compose d'un plan de contrôle et d'un plan de données séparées. Grâce à cette isolation, cette technique permet à un contrôleur de prendre des décisions de routage et permet aux administrateurs réseau de configurer correctement un équilibrage de charge intelligent et correct[18].

Depuis que le niveau de contrôle fonctionne comme le cerveau du réseau, donc l'erreur dans le niveau de contrôle, qui est généralement l'unité de contrôle, affecte le travail de l'ensemble du réseau[18]. De plus, une erreur de niveau de données peut survenir à cause de problèmes avec les commutateurs ou les liens entre elles. Donc une défaillance[18] au niveau des données ou au niveau des données entraîne l'indisponibilité du réseau[18].

Pour résoudre ce problème, la redondance doit utiliser des composants supplémentaires dans le système pour obtenir un réseau tolérant aux pannes afin que le système ne soit pas affecté pendant le fonctionnement. Il existe deux types de redondances[18].

Redondance matérielle

Cette solution doit être conçue comme une architecture SDN basse sur la tolérante aux pannes, selon [18] la majorité des chercheurs ont imaginé une solution qui consiste à connecter chaque commutateur à plusieurs contrôleurs afin qu'un seul contrôleur joue le rôle de maître et les autres contrôleurs prennent le contrôle en cas d'erreur. Mais l'inconvénient est qu'il s'agit d'une redondance qui utilise du matériel supplémentaire et nécessite donc des ressources réseau supplémentaires, ce qui augmente le coût du réseau[18].

Redondance logicielle

La redondance logicielle utilise plusieurs versions d'un logiciel dans le système et suppose que différentes versions du logiciel ne tomberont pas en panne au même instant[18]. La tolérance aux pannes peut être obtenue en utilisant la redondance logicielle dans le réseau, La manière la plus courante pour obtenir la redondance logicielle consistée à utiliser les répliques de différentes applications et programmes de contrôleur[18]. Dans cette approche, si une réplique rencontre une erreur, une autre réplique peut prendre le relais[18].

3.8.3 Sécurité

Lorsque nous parlons de réseaux, y compris SDN, la sécurité est une caractéristique très importante afin d'assurer la convivialité, la sécurité et la protection de tous les éléments et informations [11]. Tout appareil d'un réseau devra être sécurisé et défendu, en même temps l'architecture doit être capable de détecter, de résoudre et d'exposer le problème. La conception de SDN pose un problème de sécurité car la séparation entre niveaux des données et le contrôle permet des failles de sécurité et des problèmes de sécurité des données. La figure 3.2 montre les attaques potentielles de l'architecture SDN [11].

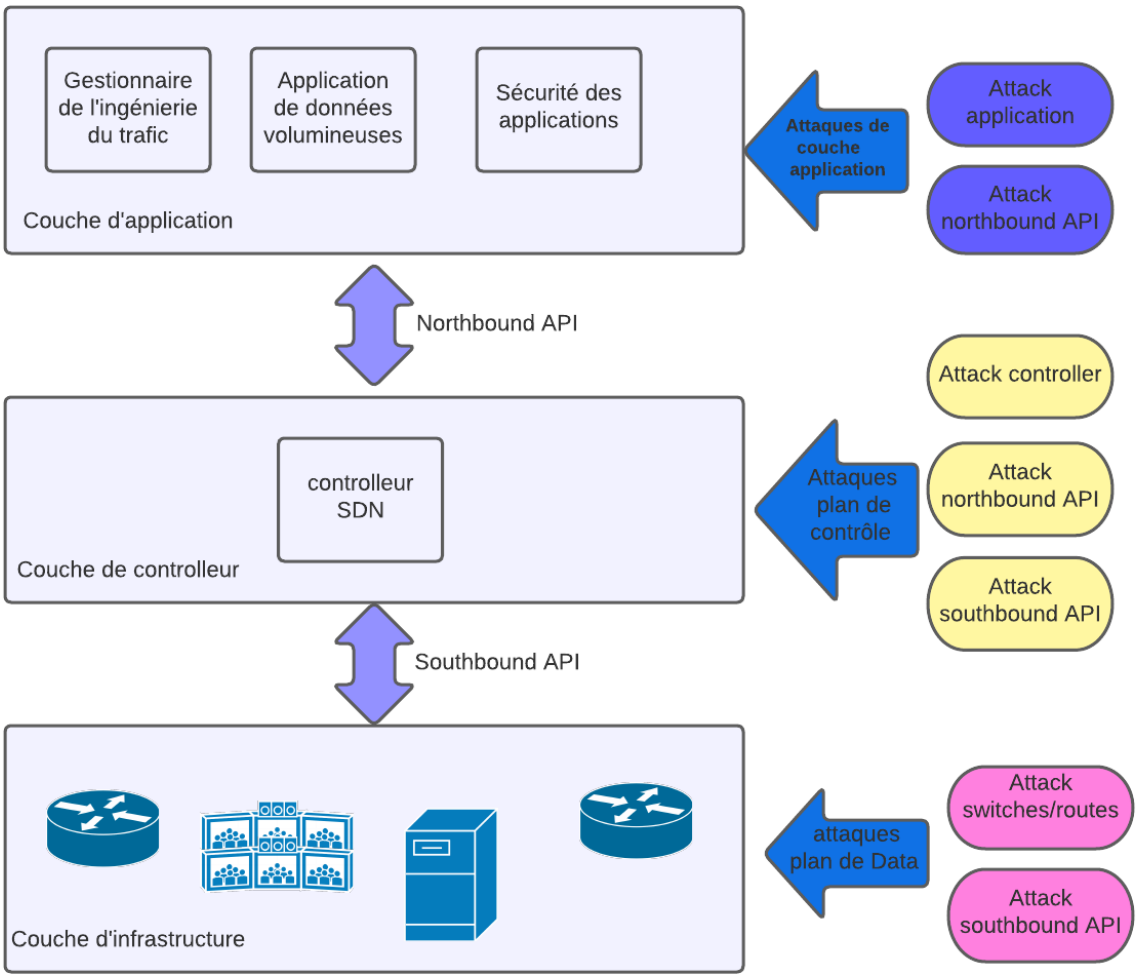


Fig. 3.2 : Les attaques potentielles de l'architecture SDN

Attaque au plan de données

Dans SDN, les règles de flux changent fréquemment en réponse à différents flux, et lorsque nous parlons de périphériques SDN, ils ont une mémoire limitée pour stocker le flux de base, de sorte qu'un attaquant avec une attaque Dos peut créer de grandes quantités de paquets en utilisant différents protocoles pour remplir la table de flux rapidement, ce qui produit une saturation de la capacité mémoire de l'équipement, ce qui entraîne des pannes et des pertes de réseau[11][22].

Attaque au plan de contrôle

Les contrôleurs sont essentiels aux réseaux définis par logiciel, mais comme ils prennent des décisions centralisées qui peuvent déclencher les réseaux en cas de faille de sécurité, cela devient une faiblesse unique[11]. Les attaques DDoS[4] peuvent cibler un contrôleur, dégradant ses performances, sa disponibilité et l'intégrité du réseau. Le déploiement de

[4] *Distributed Denial-of-Service*

plusieurs contrôleurs distribués peut aider à atténuer de tels scénarios[22].

Lors de l'utilisation de contrôleurs distribués, des problèmes de sécurité se posent en termes d'authentification, de cohérence et d'évolutivité des différentes règles de politique dans chaque domaine. Des canaux sécurisés sont nécessaires pour assurer la communication entre les contrôleurs[22]. De plus, il n'y a pas de spécification standard sur le protocole de communication et sur la manière dont il doit être sécurisé afin d'échanger des informations d'état entre les contrôleurs , si un contrôleur est compromis, plusieurs contrôleurs peuvent commencer à échouer en raison des problèmes évoqués précédemment , c'est ce qu'on appelle un problème d'échec en cascade[22].

Attaque au plan d'application

l'attaquant peut injecter des données malveillantes dans la couche d'application pour surveiller un nœud de réseau qui peut infecter d'autres nœuds de réseau connectés par incération un code malveillant pour surveiller le flux des paquets réseau et voler des informations précieuses, l'attaquant peut obtenir un accès non autorisé au nœud réseau[11].

Attaque au canal de communication

Afin de créer, modifier et supprimer des règles de flux à l'intérieur des appareils SDN, il est impératif d'établir un canal de communication entre le contrôleur et l'appareil[22]. Les attaquants peuvent mettre en œuvre des attaques de saturation, des attaques DDoS, des attaques de "man-in-the-middle", pour épuiser ce canal[22].

En raison de ces attaques, la sécurité d'architecture SDN doit être renforcée en ce qui concerne la détection et l'atténuation des cyberattaques pour chaque plan dans l'architecture[22]. Au niveau des applications, où les applications réseau telles que le routage, la surveillance du trafic et la virtualisation sont déployées, les techniques d'atténuation tentent d'empêcher les accès non autorisés et de fournir de meilleurs mécanismes de sécurité pour empêcher l'injection des entités malveillantes entrent dans le contrôleur via l'API nord[22]. Dans le plan de contrôle, en plus de protéger le contrôleur, les améliorations de sécurité visent à fournir un canal de communication plus robuste entre l'application réseau et le contrôleur (API nord) et entre les contrôleurs dans une architecture multicontrôleurs[22].Au niveau des données, les techniques d'atténuation visent à éviter de dépasser la capacité de mémoire du commutateur et d'éviter de placer des règles fausses ou malveillantes sur la table de transfert, et de sécuriser le canal de communication entre les clés et le contrôleur (API Sud)[22].

Selon [22] , il existe des stratégies qui ont été suggéré est pour améliorer la sécurité, dont certaines sont basées sur des statistiques où un modèle de trafic commun est calculé et utilisé plus tard dans des tests d'inférence pour déterminer si les flux de trafic suivent le modèle calculé, ou par l'apprentissage automatique qui est utilisé dans SDN pour développer des systèmes capables d'apprendre à partir de flux de données, d'extraire des modèles cachés et de mettre en œuvre des décisions automatiquement, cela s'est également avéré être une technique efficace de détection et d'atténuation des attaques qui améliore le taux de détection et réduit les taux de fausses alarmes.

3.8.4 Flexibilité

Les ingénieurs réseaux doivent être disponibles pour pouvoir gérer et prendre en charge le travail du réseau. Un réseau trop complexe ou difficile à maintenir, ne peut pas fonctionner efficacement ou être capable de s'adapter aux changements qui surviennent dans le réseau au cours de son exploitation[18]. Ces changements sont parfois prévisibles, mais ils peuvent souvent se produire sans avertissement, de sorte que le réseau doit être suffisamment flexible pour s'adapter aux développements futurs et aux besoins des concepteurs de réseau. Il doit être conçu. Ses règles et son comportement peuvent être modifiés en fonction des changements dynamiques[18].

La flexibilité du SDN peut être augmentée par l'externalisation, car toutes les opérations ne doivent pas être effectuées au sein du réseau, il permet donc de combiner des technologies modernes et évolutives avec le SDN pour un meilleur niveau d'application et au niveau du contrôle avec les réseaux adaptatifs qui interagissent mieux[18]. des exemples de telles technologies sont le "Big Data", le "Cloud Computing", les commutateurs optiques, etc [18].

Selon[18], les chercheurs ont proposé une architecture pour ajouter des fonctionnalités aux réseaux en utilisant l'externalisation. La figure 3.3 illustre les composants de l'architecture[18], c'est-à-dire le traducteur de politique, l'API de fonctionnalité, le contrôleur et le gestionnaire de ressources[18]. Le traducteur de politique combine toutes les fonctionnalités et définit les règles de trafic associées en fonction des poli-tiques combinées. Le gestionnaire de ressources crée, supprime et surveille le mouvement des instances de fonctionnalité. Les principaux avantages de ce travail sont la réduction des coûts et de la complexité de la gestion, l'amélioration des fonctionnalités grâce à la spécialisation de l'externalisation et l'augmentation des options de services[18].

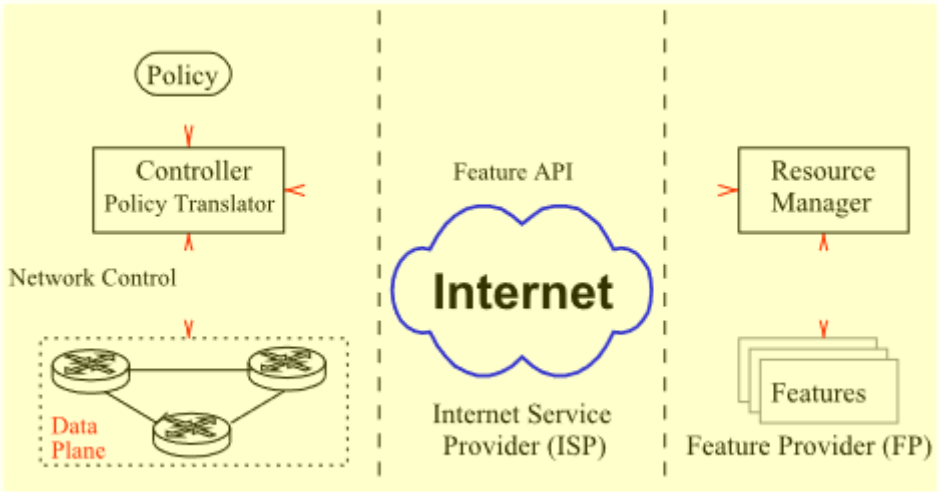


FIG. 3.3 : Externalisation vers un fournisseur de fonctionnalités dans le SDN

3.9 Conclusion

À travers ce chapitre, il a été souligné que le SDN change toute la vision des réseaux par rapport aux technologies précédentes de telle sorte qu'il améliore les performances des réseaux dans différents domaines d'utilisation grâce à cette architecture éteinte qui relie les plans d'infrastructure réseau par des API du nord et du sud comme le protocole Open flow sur lequel repose cette technique.Plus que cela, SDN défié les difficultés, les défis et les obstacles auxquels il a été conforté grâce aux recherches dans ce domaine .

Chapitre 4

Application

4.1 Introduction

Pour montre que les avantages et les applications de SDN sont réalisables on a proposé dans ce chapitre une application explicative de cette technique de telle façon qu'on peut visualiser les réseaux et les flux qui circulent dans les réseaux et aussi pour prouver que n'importe fonctionnalité de réseau classique peuvent implémenter dans le réseau SDN .et comme on a déjà dit que le SDN assure la sécurité d'un réseau et comme connu que le ACL considéré comme un concept de sécurité, dans ce but ,on a proposé une application d'implémentation ACL.

4.2 logiciels utilisés

4.2.1 Mininet

Mininet est un simulateur de réseau écrit en Python et C qui peut créer des hôtes virtuels, des commutateurs et des liens. Il prend en charge divers commutateurs OpenFlow basés sur des logiciels, des topologies personnalisées et fournit une API Python pour la programmation du réseau [14].

4.2.2 OpenDaylight

Le projet OpenDaylight est un projet collaboratif open source hébergée par la Fondation Linux. Ce projet sert de plate-forme pour SDN pour la surveillance ouverte et centralisée des périphériques de réseau informatique[15].

4.3 L'implantation de l'application

Dans cette section on va explique comment on a implémenter l'application avec les étapes .

- Étape 1 : On va créer un réseau de 1 serveur et 5 hôtes avec Mininet .On propose que le hôte 1 est l'administrateur et hôte 6 est le serveur.

```
class ACLTopology(Topo):

    def __init__(self, **opts):
        """Create custom topo."""

        # Initialize topology
        super(ACLTopology, self).__init__(**opts)
        #Topo.__init__(self)

        # Add hosts and switches
        h1 = self.addHost('h1', ip='10.0.0.1', mac="00:00:00:00:00:01")
        h2 = self.addHost('h2', ip='10.0.0.2', mac="00:00:00:00:00:02")
        h3 = self.addHost('h3', ip='10.0.0.3', mac="00:00:00:00:00:03")
        h4 = self.addHost('h4', ip='10.0.0.4', mac="00:00:00:00:00:04" )
        h5 = self.addHost('h5' ,ip='10.0.0.5', mac="00:00:00:00:00:05")
        h6 = self.addHost('h6' ,ip='10.0.0.6', mac="00:00:00:00:00:06")

        opts = dict(protocols='OpenFlow13')

        # Adding switches
        s1 = self.addSwitch('s1', dpid="0000000000000001")

        # Add links
        self.addLink(h1, s1)
        self.addLink(h2, s1)
        self.addLink(h3, s1)
        self.addLink(h4, s1)
        self.addLink(h5, s1)
        self.addLink(h6, s1)

    def run():
        c = RemoteController('c', '127.0.0.1', 6633)
        net = Mininet(topo=ACLTopology(),  controller=None)
        net.addController(c)
        net.start()
```

- Étape 2 : On va lancer le contrôleur "OpenDayLight" comme illustre comme illustré sur la figure 4.1 avec la commande :

```
sudo  ./bin/karaf
```

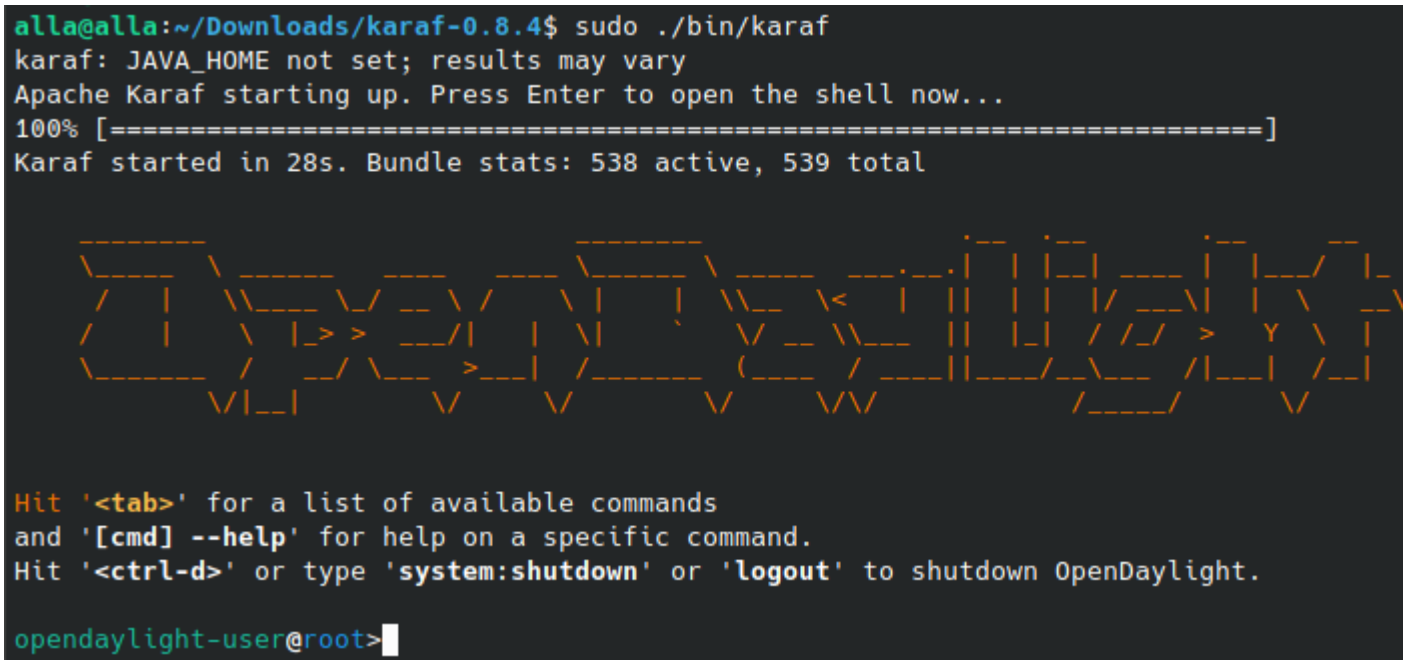


FIG. 4.1 : Lancement de contrôleur

- Étape 3 : On va lancer la topologie comme illustré sur la figure 4.2 qui on a créé avec Mininet avec la commande :

```
sudo python3 topo.py
```

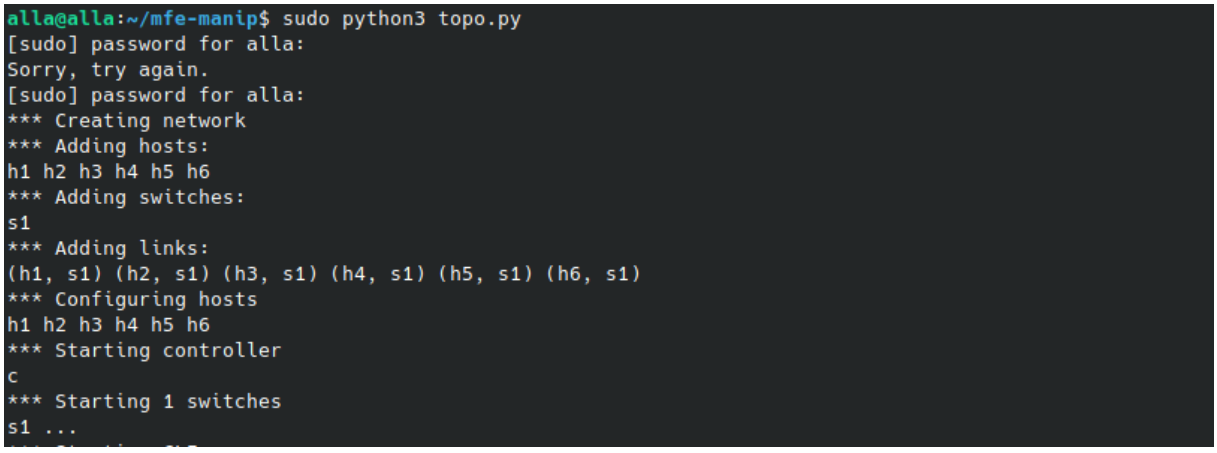


FIG. 4.2 : Lancement du la topologie

La topologie est connectée avec opendaylight, donc maintenant on peut visualiser notre réseau avec le contrôleur.la figure 4.3 illustré la visualisation du réseau .

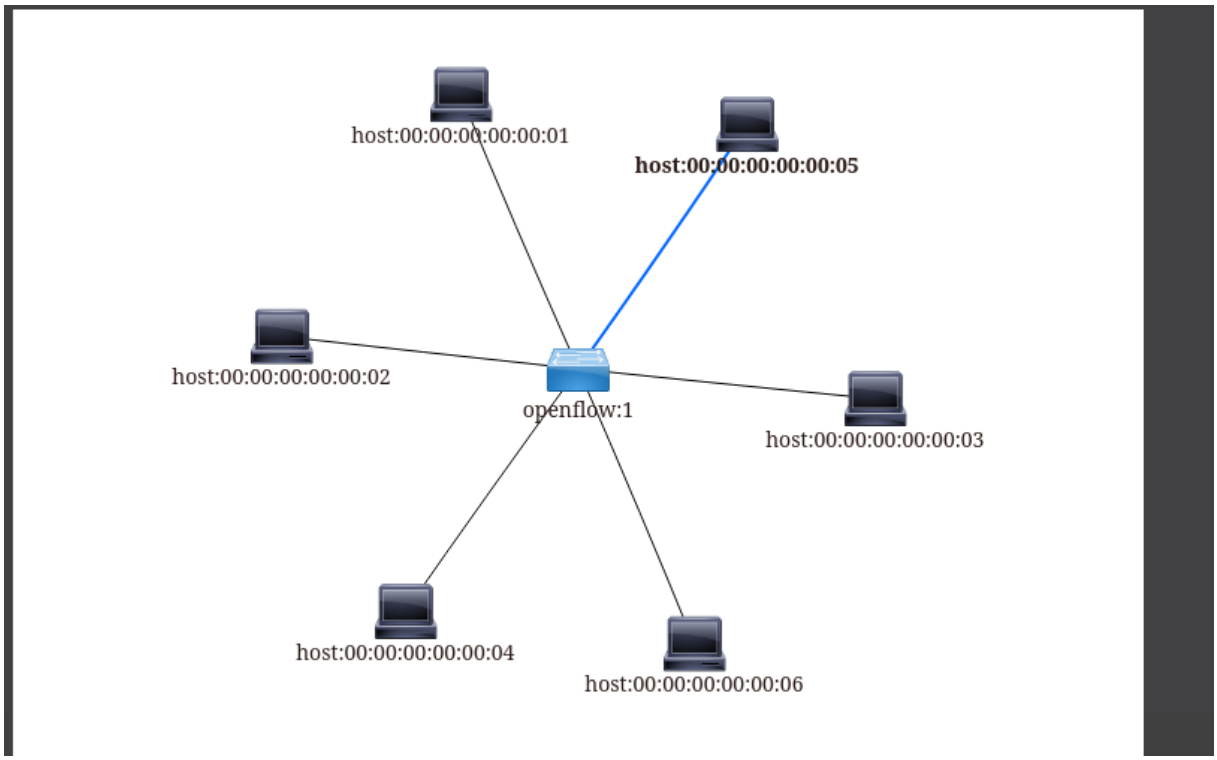


FIG. 4.3 : Visualisation du réseau

- Etape 4 : Implimentation d'ACL : dans cette étape on va ajouter deux flux pour réaliser l'implémentation de l'ACL.
  Le première Flow sert à bloquer tous les paquets du type TCP sur le port 80 des tous les hôtes vers le serveur (hôte 6). Le code ci-dessous est le code **JSON !**[1] qui décrit ce Flow.

```
{
  "flow-node-inventory:flow": [
    {
      "table_id": "0",
      "id": "127",
      "priority": "1400",
      "match": {
        "ip-match": {
          "ip-dscp": 2,
          "ip-protocol": 6,
          "ip-ecn": 2
        },
        "tcp-destination-port": 80,
        "ethernet-match": {
          "ethernet-type": {
            "type": 2048
          },
```

[1] **JSON !**

```
            "ethernet-destination": {
              "address": "00:00:00:00:00:06"
            }
          }
        },
        "instructions": {
          "instruction": [
            {
              "order": 0,
              "apply-actions": {
                "action": [
                  {
                    "order": 0,
                    "drop-action": {}
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
```


Si on test une requête ICMP[2] (Ping), on remarque qu'elle fonctionne, mais si on test une requête HTTP[3] qui utilise le protocole TCP avec le port destination 80 elle ne marche pas comme illustré dans les figure 4.4 et 4.5.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
```

FIG. 4.4 : Test de Ping entre les hôtes et serveur

[2] *Internet Control Message Protocol*
[3] *Hypertext Transfer Protocol*

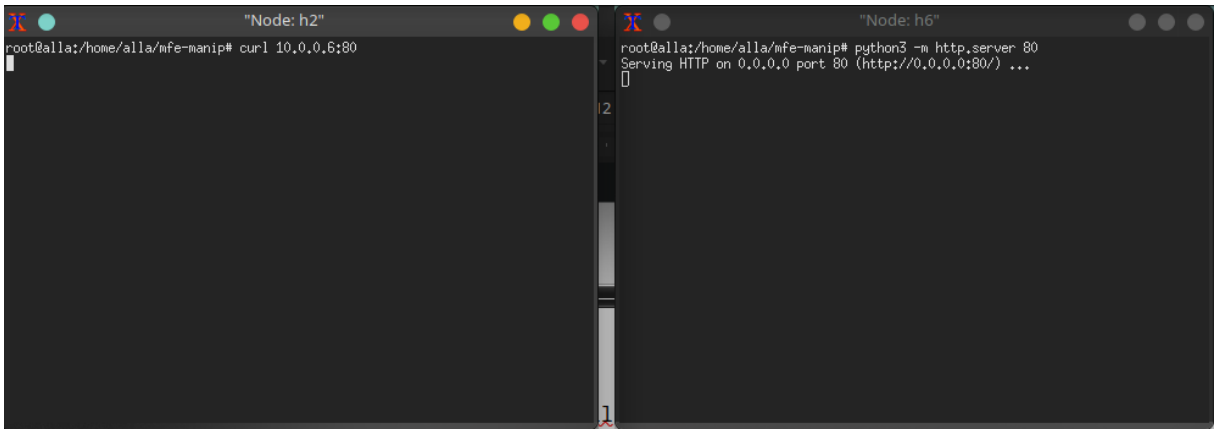


FIG. 4.5 : La connexion de l'administrateur au serveur a échoué au port 80.

- Le deuxième flux permet à l'hôte 1 d'accéder au port 80 du serveur (hôte 6) . le code de deuxième flux est illustre dans ci dessus


```
{
  "flow-node-inventory:flow": [
    {
    "table_id": "0",
    "id": "128",
    "priority": "1400",
      "match": {
        "ip-match": {
          "ip-dscp": 2,
          "ip-protocol": 6,
          "ip-ecn": 2
        },
        "tcp-destination-port": 80,
        "ethernet-match": {
          "ethernet-type": {
            "type": 2048
          },
          "ethernet-destination": {
            "address": "00:00:00:00:00:06"
          },
          "ethernet-source": {
            "address": "00:00:00:00:00:01"
          }
        }
      },
      "instructions": {
        "instruction": [
          {
            "order": 0,
            "apply-actions": {
```

```
                "action": [
                  {
                    "order": 0,
                    "output-action": {
                      "output-node-connector": "6",
                      "max-length": "65535"
                    }
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
```

- Après l'addition de ce flux, l'hôte 1 va être capable d'accéder le port 80 du serveur comme illustré dans la figure 4.6.

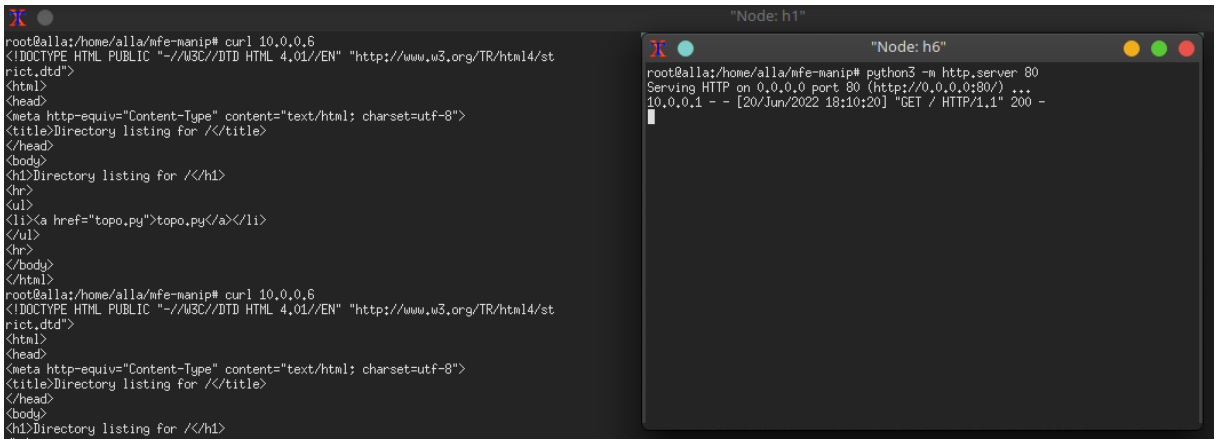


Fig. 4.6 : le code de deuxième flux

4.4 Conclusion

Dans ce chapitre on a conclu que les applications de SDN sont réalisables et fonctionnelles pour la gestion de réseaux de telle façon qu'ils peuvent contrôler les flux qui passent dans les infrastructures réseau avec plusieurs types de plate-forme de contrôleur semblable à Opendaylight et avec des services différents.

On a conclu aussi que cette implantation d'ACL avec SDN mettre une solution convient aux les petites entreprises car les coûts pour obtenir un commutateur ou pare-feu sont chers par contre une implémentions base sur le SDN qui fait le même fonctionnalité implémenté d'une manière plus moderne et flexible .

Conclusion Générale

Ce travail illustre les aspects fondamentaux, les avantages et les défis de l'architecture SDN, ainsi que le SDN abordant de nombreux problèmes qui ont rencontrés les réseaux traditionnels en séparant les niveaux de contrôle. Il illustre l'historique, le fonctionnement et les défis de SDN et OpenFlow.
Finalement nous avons également fait une illustration du mécanisme SDN, où nous avons fait une implémentassion d'ACL basé sur Openflow dans un réseau SDN et conclure que les applications SDN sont flexibles et facilement réalisables et adaptées aux petites entreprises car les coûts pour obtenir une solution SDN sont moins chère qu'une solution classique. Le SDN a le potentiel de se développer et être utilisé dans tout les domaines.

Bibliographie

[1] Suhail Ahmad et Ajaz Hussain Mir. "Scalability, consistency, reliability and security in SDN controllers : a survey of diverse SDN controllers". In : *Journal of Network and Systems Management* 29.1 (2021), p. 1-59.

[2] Mustafa Hasan Albowarab, Nurul Azma Zakaria et Zaheera Zainal Abidin. "Software Defined Network : Architecture and Programming Language Survey". In : *Int. J. Pure Appl. Math* 119.18 (2018), p. 561-572.

[4] Andrew T Campbell et al. "Open signaling for ATM, internet and mobile networks (OPENSIG'98)". In : *ACM SIGCOMM Computer Communication Review* 29.1 (1999), p. 97-108.

[5] Martin Casado et al. "Ethane : Taking control of the enterprise". In : *ACM SIGCOMM computer communication review* 37.4 (2007), p. 1-12.

[6] A Crouch et al. "Forwarding and control element separation (ForCES) applicability statement". In : (2010).

[8] Giuseppe Di Fatta et Giuseppe Lo Re. "Active networks : an evolution of the internet". In : (2001).

[9] Avri Doria et al. *Forwarding and control element separation (ForCES) protocol specification*. Rapp. tech. 2010.

[10] Open Networking Fundation. "Software-defined networking : The new norm for networks". In : *ONF White Paper* 2.2-6 (2012), p. 11.

[11] Saad H Haji et al. "Comparison of software defined networking with traditional networking". In : *Asian Journal of Research in Computer Science* (2021), p. 1-18.

[12] Fei Hu, Qi Hao et Ke Bao. "A survey on software-defined network and openflow : From concept to implementation". In : *IEEE Communications Surveys & Tutorials* 16.4 (2014), p. 2181-2206.

[18] Surbhi Saraswat et al. "Challenges and solutions in software defined networking : A survey". In : *Journal of Network and Computer Applications* 141 (2019), p. 23-58.

[20] Hadar Sufiev et al. "Dynamic SDN controller load balancing". In : *Future Internet* 11.3 (2019), p. 75.

[21] PV Tijare et D Vasudevan. "The northbound APIs of software defined networks". In : *International journal of engineering sciences and research technology* 5.10 (2016), p. 501-513.

[22] Ismael Amezcua Valdovinos et al. “Emerging DDoS attack detection and mitigation strategies in software-defined networks : Taxonomy, challenges and future directions”. In : *Journal of Network and Computer Applications* 187 (2021), p. 103093.

[23] Richard Wang, Dana Butnariu et Jennifer Rexford. “{OpenFlow-Based} Server Load Balancing Gone Wild”. In : *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 11)*. 2011.

[24] James Yu et Imad Al Ajarmeh. “An empirical study of the NETCONF protocol”. In : *2010 Sixth International Conference on Networking and Services*. IEEE. 2010, p. 253-258.

Webographie

[3] *Applying NOX to the Datacenter*. URL : https://conferences.sigcomm.org/hotnets/2009/papers/hotnets2009-final103.pdf.

[7] *DCAN*. URL : https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/dcan/.

[13] LOGICFINDER. *SDN Technology The Transformer of Network Automation*. URL : https://www.logicfinder.net/sdn-technology-the-transformer-of-network-automation/.

[14] *Mininet*. URL : http://mininet.org/.

[15] *OpenDaylight*. URL : https://en.wikipedia.org/wiki/OpenDaylight_Project.

[16] *Opennetworking*. URL : https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/.

[17] Linda ROSENCRANCE. *software-defined networking (SDN)*. URL : https://www.techtarget.com/searchnetworking/definition/software-defined-networking-SDN.

[19] Nirmal SHARMA. *Software Defined Networking (SDN) Advantages*. URL : https://www.serverwatch.com/guides/software-defined-networking-advantages/.

ملخص

الشبكات المعرفة بالبرمجيات هي تقنية تفصل مستوى التحكم عن مستوى البيانات لجهاز الشبكة. تعتبر الشبكة المعرفة بالبرمجيات من أهم التقنيات الحديثة في مجال الشبكات ، لاحتوائها على ديناميكياتها وبنيتها التي تسهل الإدارة وتوفر مساحة لتطبيق عدد كبير من التطبيقات ، وهذا ما دفع الباحثين إلى النظر فيه وتطويره للأفضل.

في هذا الموجز، سنتحدث عن الشبكة المحددة بالبرمجيات بشكل عام، وهيكلها، وخصائصها والتحديات التي واجهتها.

كلمات مفتاحية :. فتح تيار ، تبديل ، متحكم ، إنترنت ، شبكة.

Résumé

Le réseau définie par logiciel est une technique qui permet de séparer le plan de contrôle du plan de données d'un équipement réseau. le réseau définie par logiciel est considéré comme l'un des techniques modernes le plus important dans le domaine des réseaux, car il contient sa dynamique et sa structure qui facilitent la gestion et donnent de l'espace pour l'application d'un grand nombre d'applications, et c'est ce qui a poussé les chercheurs à le considérer et le développer pour le mieux.
Dans cette mémoire, nous parlerons du réseau défini par logiciel en général, de sa structure, de ses caractéristiques et des défis rencontrés.

Mots clés : Flux ouvert , commutateur , contrôleur , internet , réseau .

Abstract

Software-defined network is a technique for separating the control plane from the data plane of a network equipment. the software-defined network is considered to be one of the most important modern networking techniques because it contains its dynamics and structure which facilitate management and give space for the application of a large number of applications, and this is what prompted researchers to consider it and develop it for the better.

In this brief, we will talk about the software-defined network in general, its structure, its characteristics and the challenges encountered.

Keywords : OpenFlow ,switch , internet , network , controller .
