République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Abderrahmane Mira

Faculté de la Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunications

Spécialité : Réseaux et Télécommunications

Thème

Étude et Simulation D'une Solution SDN dans L'internet Des Objets

Préparé par :

M.Merabtine Abdelaziz

M.SAADI Abdelwahab

Dirigé par :

M.SADI Mustapha

AZNI Mohamed

Examiné par :

M. Bessaad (P)

M.Mokrani

Année universitaire : **2021/2022**

Dédicace

Toutes les lettres ne sauront trouver les mots quil faut Tous les mots ne sauraient exprimer la gratitude, lamour, le respect, la reconnaissance. Aussi, cest tout simplement que je dédie ce travail :

Á mes chers parents, pour votre amour inconditionnel, votre affection, vos sacrifices et vos encouragement. Qu'Allah le tout puissant vous accorde santé,bonheur et longue vie et faire en sorte que jamais je vous déçoive.

Á mes soeurs Sara ,Amel et Lamia pour votre présence a mes cotés , votre gentillesse et bonne humeur ; je ne sais comment exprimer mon amour et ma reconnaissance.

Á ma grand-mère pour toutes tes prières .

Á mon binôme Abdelaziz pour ton courage et ta rigueur .

Á ma princesse Pescyssa ; merci de supporter ma folie !

Á tout l'équipe du I113.

Abdelwahab

Dédicace

Toutes les lettres ne sauront trouver les mots quil faut Tous les mots ne sauraient exprimer la gratitude, lamour, le respect, la reconnaissance. Aussi, cest tout simplement que je dédie ce travail :

Á mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études. Qu'Allah le tout puissant vous accorde la santé, le bonheur et une longue vie et faire en sorte que jamais je vous déçoive.

Á ma soeur Maria et mon frère Zaki pour leurs encouragements permanents, et leurs bonne humeur qui m'est un grand soutien moral.

Á ma grand-mère paternelle et à mon grand père maternel pour toutes vos prières .

Á la mémoire de ma grand-mère maternelle et mon grand père paternel que dieu vous accueil dans son vaste paradis.

Á toute ma famille et mes amis pour leur soutien tout au long de mon parcours universitaire,

Á mon binôme Abdelwahab pour ton courage et ta rigueur .

Abdelaziz

Remerciements

Tout d'abord, nous remercions Dieu le Tout-Puissant qui nous a donné le courage pour élaborer ce modeste travail.

Nous remercions nos parents, source d'amour, de tendresse et la lumière de nos existence, pour leurs sacrifices et leurs encouragements.

Nous remercions aussi notre encadreur *Mr.SADI Mustapha* pour ses suivis, ses conseils et encouragements durant la réalisation de ce mémoire.

Nous remercions vivement les membres de jury de nous avoir fait l'honneur d'accepter, d'examiner et de juger notre travail.

Nos remerciements vont également à tous nos enseignants et toutes les personnes qui ont participé de près ou de loin dans notre formation ou dans l'élaboration de ce mémoire.

Finalement, nous remercions tous et celles qui nous ont soutenus et qui n'ont pas cessé de nous encourager jusquà laboutissement de ce travail.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AD-SAL	API Driven SAL
API	Application Programing Interface (Interface de programmation dapplication)
BGP	Border Gateway Protocol
Br	Bridge
BYOD	Bring Your Own Device(Apporter son propre matériel)
CLI	Command Line Interface (Ligne de commande)
CSDN	Cellular SDN (SDN cellulaire)
CoAP	Constrained Application Protocol
DDoS	Distributed Denial of Service (Déni de service distribué)
DPI	Deep Packet Inspection (Inspection approfondie des paquets)
DS	Differentiated Services (Services différenciés)
FTP	File Transfer Protocol (Protocole de transfert de fichier)
Gb	Giga Bit
HTTP	Hyper Text Transfer protocol (Protocole de Transfer hyper texte)
I2RS	Interface to Routing System(Interface vers système de routage)
IoT	Internet of Things (Internet des objets)
IUT	Union International des Télécommunications
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISO	International Standardization Organisation
ITU-T	International Telecommunication Union - Telecom
LAN	Local Area Network (Réseau local)
MAC	Media Access Control
MD-SAL	Model Driven SAL
MPLS	Multiprotocol Label Switching
NaaS	Network as a Service (Réseau en tant que service)
NETCONF	NETwork CONFiguration protocol (Protocole de configuration réseau)
NIB	Network Information Base (Base dinformation réseau)
NIST	National Institute of Standards and Technology
ODL	Opendaylight
OF	Openflow
OFM	OpenFlow Manager
ONF	Open Networking Foundation
ONOS	Open Network Operating System

OSGI	OpenServicesGateway Initiative
OVF	Open View Finder
OVS	Open Virtual Switch
Ovsdb	open virtual switch data base
OXM	OpenFlow eXtensible Match (Openflow à correspondance extensible)
QOS	Quality of Service (Qualité de service)
RAM	Read Access memory
REST	REpresentational State Transfer (Transfert détat représentatif)
RFID	Radio Frequency Identification
SAL	Service Abstraction Layer (couche dabstraction de service)
SDN	Software Defined Network (Réseau défini par logiciel)
SNMP	Simple Network Management Protocol (Simple protocole de gestion de réseau)
SSH	Secure Shell
STP	SpanningTree Protocol
SDIoT	Software Defined Internet of Things
SDStore	Software Defined Storage
SDSec	Software Defined Security
TCAM	Tenary contenant addressables memory
TCP	Transmission Control Protocol (Protocole de contrôle de transmission)
TLS	Transport Layer Security (Sécurité de la couche de transport)
TLV	Type Length Value (Valeur de type de longueur)
TTL	Time To Live
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network (Réseau local virtuel
VM	Virtual Machine (Machine virtuelle)
VoD	Video on Demand (Vidéo à la demande)
VoIP	Voice over Internet Protocol (Téléphonie IP)
VSDN	Video over SDN (Vidéo par SDN)
WAN	Wide Area Network
WSN	Wireless Sensors Network
6LoWPAN	IPv6 Low power Wireless Personal Area Networks

Introduction Générale

Avec la croissance de l'utilisation de la technologie de l'information, nous assistons à une énorme augmentation du trafic qui circule dans les réseaux. Il y a un grand nombre de périphériques et d'applications modernes qui sont inter-connectés. Les innovations technologiques récentes telles que la virtualisation, le Cloud Computing et l'Internet of Things (IoT) et tant d'autres poussent les réseaux à montrer facilement leurs limites. En effet, depuis déjà plusieurs années, il est communément admis que les architectures IP traditionnelles sont d'une part particulièrement complexes à configurer à cause de la nature distribuée des protocoles réseaux et d'autre part, difficile à faire évoluer en raison du fort couplage qui existe entre le plan de contrôle et le plan de données des équipements d'interconnexion existants. En effet, les administrateurs réseau doivent gérer une gamme de données, de type de services et de périphériques différents. Cette gestion devient très difficile avec les outils des réseaux traditionnels qui nont pas été conçus pour faire face à des topologies évolutives à grande échelle.

Le problème réside en effet dans le manque d'innovation dans l'univers des réseaux qui a fait peser des contraintes importantes sur le déploiement des applications réseaux. Le concept de IoT est relativement simple mais les problèmes sont très nombreux car ces dispositifs connectés ne disposent pas suffisamment de capacité pour gérer les communications et les traitements associés aux applications. La gestion de ces objets hétérogènes nécessite dautres modèles de gestion des réseaux donc Les chercheurs se sont penchés sur des recherches pour apporter aux entreprises et aux fournisseurs de services une solution. Cette dernière tente de résoudre la flexibilité et une gestion plus simple, c'est pourquoi depuis quelques années, il se développe le concept des réseaux définis par logiciel appelé SDN . Il s'agit d'une approche qui centralise puis simplifie la gestion du réseau. Ce qui permet aux administrateurs réseau d'orchestrer et d'automatiser à travers une interface de contrôle logicielle sans accéder aux composants physiques.

L'objectif de ce travail de fin d'études, est de présenter et identifier ce nouveau paradigme incontournable aux réseaux du futur. C'est notamment le réseau programmable (mieux connue sous le nom de SDN) qui permet une disponibilité ouverte et immédiate pour les futures applications réseaux commerciales, Cela permet de concevoir des réseaux plus intelligents et plus flexibles.
Ainsi, ce manuscrit est organisé en quatre chapitres suivis d'une conclusion générale, à savoir :

- **Le premier chapitre** qui sera axé sur le principe du SDN , nous essayons dintroduire la technologie du SDN en détail.
- **Le deuxième chapitre**, qui sera axé sur la technologie IoT, avec quelques solutions SDN, qui répondent aux exigences principales de cette dernière.
- **Le troisième chapitre**, sera axé sur les logiciels nécessaires pour une simulation d'une solution SDN destiné à linternet des objets.
- **Le quatrième chapitre**, sera axé sur la simulation d'un réseau SDN, pour but de montrer l'aspect de gestion centralisé puis la flexibilité qu'apporte un réseau SDN à l'égare de la technologie IoT.
- Nous terminons ce mémoire par une Conclusion générale dans laquelle nous présentons notre contribution, ainsi que des perspectives envisagées pour ce travail.

Chapitre 1

Principe du SDN

1.1 Introduction

Un réseau est le résultat d'interconnexion entre plusieurs machines afin que les utilisateurs et les applications qui y sont exécutées puissent échanger des informations. Le contrôle distribué et les protocoles de réseau de transport en cours d'exécution à l'intérieur des routeurs et des commutateurs sont les technologies clés qui permettent aux informations, sous la forme de paquets numériques, de voyager dans le monde entier. Malgré leur adoption généralisée, les réseaux IP traditionnels sont complexes et difficiles à gérer.
Depuis 2008, on assiste à une nouvelle tendance forte à la mise en réseau avec les réseaux SDN (software-defined networking). L'émergence de cette nouvelle tendance dans l'architecture réseau est motivée par la volonté de mettre en oeuvre des principes de conception dans le domaine du réseau. Les principes que nous allons détailler dans ce chapitre.

1.2 Définition du SDN

Le SDN est une nouvelle façon de concevoir les réseaux de communication, il peut être utilisé et vendu par les opérateurs, il permet la centralisation de la logique de contrôle dans le contrôleur et la séparation des plans de contrôle et de données. SDN est une architecture émergente qui est dynamique, gérable, rentable et adaptable, ce qui la rend idéale pour la nature dynamique et les application à large bande passante daujourdhui. Des protocoles adaptés ont été dévlopés pour cette technologie (par exemple OpenFlow) . Dans ce chapitre nous allons introduire les principes du SDN.

1.3 Plan de données et plan de contrôle

Une représentation simplifiée d'un routeur est décrite par le plan de contrôle, qui détermine le Routage et un plan de données pour prendre en charge le trafic. Avec un équipement réseau traditionnel, le plan de contrôle et le plan de données résident dans le même appareil. La flexibilité apportée par l'algorithme de routage permet d'attribuer dynamiquement des itinéraires optimaux en fonction de l'évolution de la charge de trafic.

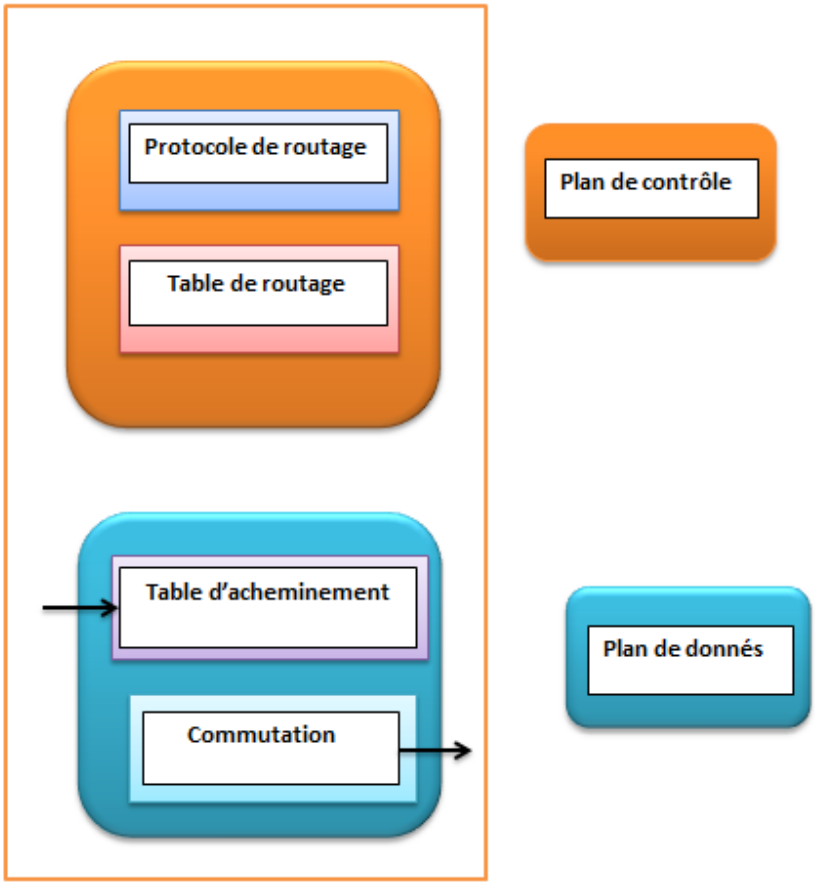

FIGURE 1.1 – Architecture d'un routeur

1.3.1 Plan de données

Il s'agit de la couche de transmission. Cela inclut les dispositifs de transmission tels que les commutateurs physiques et virtuels. Sa tâche principale est de transmettre des données, de surveiller les informations locales et de collecter des statistiques.

1.3.2 Plan de contrôle

Il se compose d'un ou plusieurs logiciels de contrôle (contrôleurs). Ces contrôleurs utilisent une Interface sud ouvertes pour contrôler le comportement des appareils et communiquent avec les couches supérieures via APIs nord pour surveiller et gérer le réseau.

1.4 Architecture du SDN

larchitecture dun réseau SDN est divisée en trois couches à savoir la couche infrastructure, la couche de contrôle et la couche application. La communication entre ces différentes couches est faite à travers les South-bound, Northbound et East-Westbound.

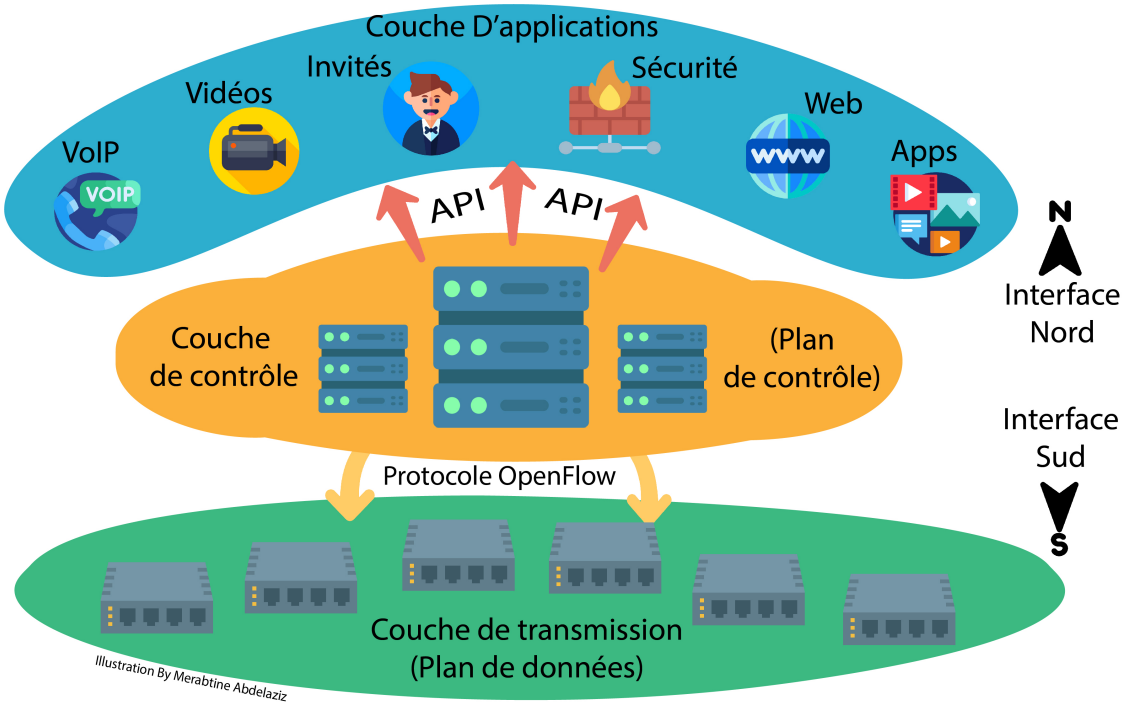

FIGURE 1.2 – Architecture dun réseau SDN

Comme nous le constations, ce qui différencie un réseau SDN et un réseau traditionnel est le fait que L'infrastructure est la principale différence entre les SDN et les réseaux traditionnels , donc le SDN repose sur un logiciel, tandis que les réseaux traditionnels sont basés sur du matériel. Le plan de contrôle étant basé sur un logiciel, le SDN est beaucoup plus flexible qu'un réseau traditionnel. Il permet aux administrateurs de contrôler le réseau, modifier les paramètres de configuration, provisionner les ressources et augmenter la capacité du réseau, et ce à partir d'une interface utilisateur centralisée, sans ajouter de matériel supplémentaire.
Il existe également des différences concernant la sécurité entre le SDN et le réseau traditionnel. Grâce à une visibilité accrue et à la possibilité de définir des voies sécurisées , Cependant, comme les réseaux software-defined utilisent un contrôleur centralisé, la protection du contrôleur est essentielle au maintien dun réseau sécurisé, et ce point de défaillance unique représente une vulnérabilité potentielle du SDN.

1.4.1 Couche de transmission

Appelée aussi « plan de données », elle est composée des équipements dacheminement tels que les switches ou les routeurs, son rôle principal est de transmettre les données, et collecter les statistiques. Le commutateur OpenFlow et le contrôleur communiquent via le protocole OpenFlow que nous allons détailler par la suite. Il y a deux types de commutateurs OpenFlow : le commutateur OpenFlow Only qui supporte uniquement OpenFlow et le commutateur OpenFlow Enable qui joue en même temps le rôle de commutateur traditionnel et OpenFlow. Il y a sur le marché plusieurs marques de commutateurs OpenFlow comme lindique le tableau :

Fabricant	Série
Arista	Arista extensible modular operating system (EOS), Arista 7124FX application switch
Ciena	Ciena Coredirector running firmware version 6.1.1
Cisco	Cisco cat6k, catalyst 3750, 6500 series
Juniper	Juniper MX-240, T-640
HP	HP procurve series- 5400 zl, 8200 zl, 6200 yl, 3500 yl, 6600
NEC	NEC IP8800
Pronto	Pronto 3240, 3290
Toroki	Toroki Lightswitch 4810
Dell	Dell Z9000 and S4810
Quanta	Quanta LB4G
Open vSwitch	Software switch. Latest version : 2.6

TABLE 1.1 – Exemple des commutateurs OpenFlow disponibles sur le marché [32]

Les actions appliquées par un commutateur OpenFlow sont similaires à celles d'un commutateur traditionnel :

1. transférer le paquet vers un ou plusieurs ports de sortie,
2. supprimer ou rejeter le paquet,
3. Modifier le champ d'en-tête du paquet
4. Encapsuler le paquet avant de l'envoyer au contrôleur SDN. Cette dernière action est spécifique au switch OpenFlow

1.4.2 Couche contrôle

appelée aussi « plan de contrôle », elle est constituée principalement d'un ou plusieurs contrôleurs SDN, son rôle est de contrôler et de gérer les équipements de l'infrastructure à travers une interface appelée "south-bound API".
Il existe de nombreux types de contrôleurs logiciels, les plus populaires sont ONOS, OpenDaylight, POX, RYU, Projecteurs et balises. Ils diffèrent par le langage de programmation utilisé et les protocoles pris en charge. Par exemple, le contrôleur OpenDaylight utilisé dans notre travail est on Java et Python [22].

1.4.3 Couche application

Représente les applications qui permettent de déployer de nouvelles fonctionnalités réseau, comme l'ingénierie de trafic, QoS, la sécurité, etc. Ces applications sont construits moyennant une interface de programmation appelée north-bound API
Le réseau est doté d'une multitudes d'applications variées dont la voix, la vidéo, les applications dentreprise, les services réseau tels que l'accès invité et la détection des intrusions qui s' adressent au contrôleurs à travers des API ouvertes pour satisfaire leurs besoins
Par exemple, le trafic vocal peut demander au contrôleur de le traiter avec le moins de latence tandis qu'un serveur de sauvegarde dentreprise peut dire au contrôleur de lui donner de la bande passante chaque fois quelle est disponible.[22], [3]

1.5 Les interfaces de communications

SDN a introduit plusieurs types d'interfaces de programmation pour permettre à divers éléments de l'architecture SDN d'interagir les uns avec les autres [11]. Cette section détaille ces interfaces de programmation.

1.5.1 Interfaces Sud

C'est dans ces interfaces que le contrôleur interagie avec l'ensembles des équipement de la couche infrastructure du réseau, et ce, via le protocole OpenFlow que nous verrons par la suite dans les section suivantes, il existe belle et bien d'autres alternatives que ce protocole, mais il " est actuellement le standard de facto, qui est largement accepté et répandu dans les réseaux SDN. "[14].
La figure ci dessous montre le Processus de communication entre les Switchs OpenFlow et le contrôleur dans ce cas nous avons pris le cas d'un seul switch.

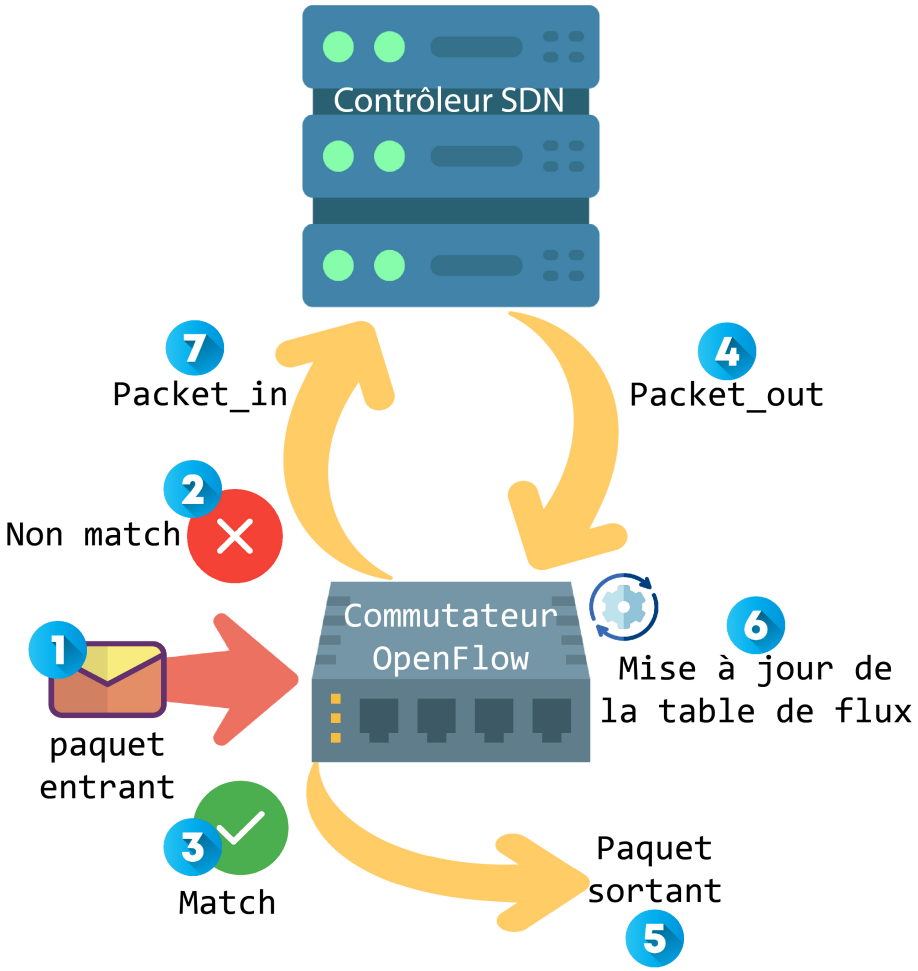

FIGURE 1.3 – Processus de communication entre le Switch et le contrôleur

Quand un paquet est reçu par le commutateur (1), son champ dentête est examiné et comparé avec le champ Match(3) dans les entrées de la table de flux. Si le match est identifié, le commutateur exécute laction correspondante dans la table de flux. Par contre, s'il n'y a pas de match (2), une demande est envoyée au contrôleur (7) sous la forme d'un Packet_in, puis le contrôleur décide de l'action a effectuer soit le paquet est detruit ou bien serait orienter vers une sortie du switch (5). selon sa configuration une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme dun Packet_out et Flow-mod au commutateur (4). Enfin, la table de flux du commutateur est actualisée (6)

1.5.2 Interfaces Nord

Une interface nord est une interface de programmation d'application (API) qui nous permet de communiquer avec un contrôleur, toutes les fonctions programmés du réseau, passent par cette interface, et permettent aux applications de consommer des ressources réseau et de modifier leur comportement de manière dynamique. "Le RESTful considéré comme l'API nord le plus répandue dans les réseaux SDN". [19]

1.5.3 Interfaces Est/Ouest

Les interfaces Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs pour synchroniser létat du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur nest actuellement disponible [3].

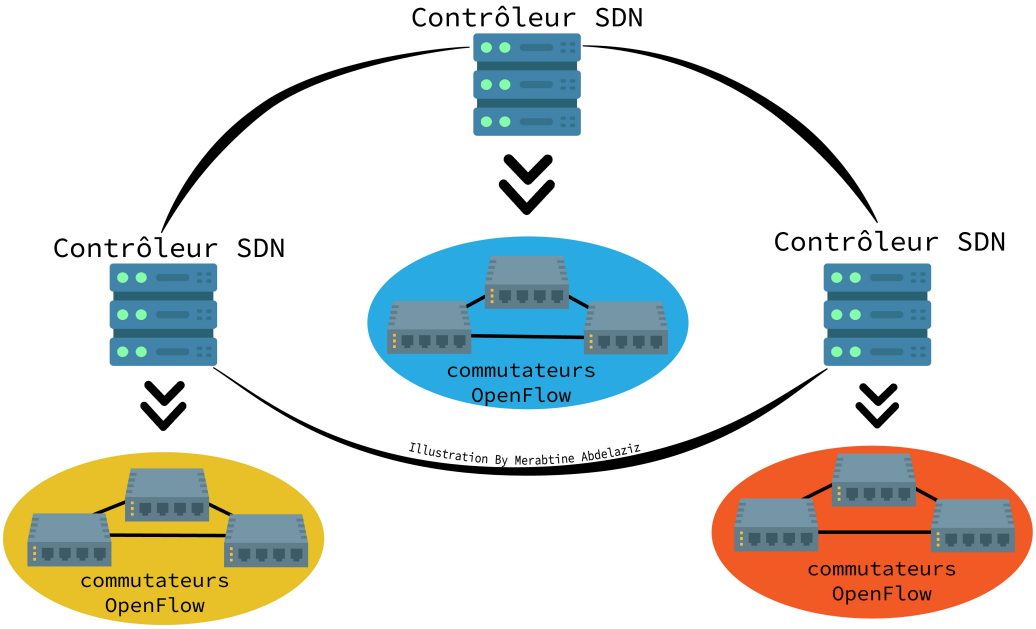

FIGURE 1.4 – Communication inter-contrôleur dans une architecture distribuée

1.6 Avantages du SDN

Le SDN offre de nombreux avantages par rapport aux réseaux traditionnels, notamment :

Contrôle renforcé avec plus de souplesse et un débit accéléré : au lieu de programmer manuellement plusieurs dispositifs matériels spécifiques à un fournisseur, les développeurs peuvent contrôler le flux de trafic sur un réseau simplement via la programmation dun contrôleur logiciel standard ouvert. Les administrateurs réseau bénéficient également de plus de souplesse dans le choix de léquipement réseau, dans la mesure où ils peuvent utiliser un protocole open source pour communiquer avec un nombre quelconque de dispositifs matériels via un contrôleur central.

Infrastructure réseau personnalisable : avec un réseau software-defined, les administrateurs peuvent configurer les services réseau et allouer des ressources virtuelles pour modifier linfrastructure du réseau en temps réel à partir d'un seul et même emplacement centralisé. Ainsi, les administrateurs réseau peuvent optimiser le flux de données à travers le réseau, en donnant la priorité aux applications qui nécessitent une plus grande disponibilité.

Sécurité renforcée : un réseau software-defined offre une visibilité sur lensemble du réseau, avec une vue plus globale des menaces de sécurité. Avec la prolifération de terminaux intelligents qui se connectent à Internet, le SDN offre des avantages évidents par rapport aux réseaux traditionnels. Les développeurs peuvent créer des zones distinctes pour les terminaux qui nécessitent différents niveaux de sécurité, ou mettre immédiatement en quarantaine les terminaux menacés afin quils ne puissent pas infecter le reste du réseau.

Une plus grande fiabilité : en raison de la gestion centralisée et de l'automatisation de la gestion des périphériques réseau. Cela entraînera moins d'erreurs de configuration, une mise en œuvre plus rapide

des modifications à l'échelle du réseau et se traduira par l'application d'une politique de réseau cohérente [3]

Innovation et improvisation beaucoup plus rapides : car cela se fera au niveau du logiciel et sera indépendant du fournisseur. Cela se traduira par un plus grand nombre d'entreprises, de développeurs individuels et de fournisseurs de logiciels utilisant l'API commune pour apporter des services innovants et générer des revenus.

Gestion du Cloud : SDN permet également une gestion simple d'une plateforme cloud. En effet, la dynamique apportée par SDN traite des problèmes spécifiques aux clouds tels que l'évolutivité, l'adaptation, ou des mouvements de machines virtuelles.

1.7 Les protocoles utilisés dans le SDN

1.7.1 Openflow

Openflow est le protocole utilisé pour la communication entre la couche transmission et la couche de contrôle, il a été initialement proposé et implémenté par l'université de Stanford, et standardisé par la suite par l'ONF, sa dernière version est 1.6 .

FIGURE 1.5 – Logo du protocol OpenFlow

Comme tout autre langage, le protocole OpenFlow possède son propre vocabulaire qui est utilisé entre le contrôleur et l'infrastructure réseau. Cependant, cela est beaucoup plus facile à apprendre par rapport à l'apprentissage de nouvelles langues que nous, les humains, parlons.

Spécification OpenFlow	**1.0**	**1.1**	**1.2**	**1.3**
Large déploiement	oui	non	non	oui
Nombre de table de flux	Une	Multiple	Multiple	Multiple
Matching MPLS	non	oui	oui	oui
Table de groupe	non	oui	oui	oui
Support IPv6	non	non	oui	oui
Contrôleurs multiples	non	non	oui	oui

TABLE 1.2 – Fonctionnalités des différentes spécifications dOpenFlow

L'ensemble des messages échangés dans le protocole OpenFlow peut être divisé en trois grands groupes (Messages symétriques, Messages asynchrones et Messages entre le contrôleur et le commutateur) :

Messages symétriques

Ce sont des messages bi-directionnels qui sont envoyés par le contrôleur ou le commutateur entre eux sans aucune sollicitation. Par exemple, des hello ou des requêtes d'écho peuvent être envoyées sans aucune sollicitation et celles-ci sont répondues par le récepteur.

Messages asynchrones

Ces messages permettent au commutateur de s'adresser librement au contrôleur sans demander la permission. C'est là que le commutateur peut informer le contrôleur des paquets abandonnés ou d'une interface en

panne. Ceux-ci sont de quatre types différents.

— STATUT DES PORTS :tout changement d'état du port est envoyé au contrôleur. Le changement d'état inclut non seulement l'arrêt du port, mais également un changement d'état 802.1d (spanning tree).
— LE MESSAGE ERREUR : Comme vous pouvez le deviner, ce message est utilisé par le commutateur pour informer le contrôleur de toute erreur détectée.
— LE MESSAGE FLOW REMOVED :toute entrée de flux ajoutée à la table de flux est associée à un délai d'inactivité et à une valeur de délai d'attente dépassé. Ainsi, chaque fois qu'un flux est supprimé, ce qui peut être dû à un délai d'inactivité, à un délai d'attente dur ou à un message de modification de flux, un message "flux supprimé" est envoyé par le commutateur au contrôleur.

Une chose à savoir ici est qu'un message de modification de flux peut spécifier si le contrôleur veut savoir si ce flux a été supprimé ou non.

— PACKET IN : Ce type de message n'est envoyé au contrôleur que s'il y a un paquet entrant dans le commutateur qui ne correspond à aucune entrée de flux (ce qui signifie qu'il doit être envoyé au contrôleur) ou au paquet correspond à la règle selon laquelle le paquet doit être envoyé au contrôleur.

Messages entre le contrôleur et le commutateur

Ces messages sont initiés par le contrôleur et utilisés pour gérer les commutateurs ou pour obtenir des informations des commutateurs. Ceux-ci sont de six types différents :

État de lecture : utilisé par le contrôleur pour lire les compteurs à partir des compteurs par flux, par port et par file d'attente [22].

Modifier l'état : principalement utilisé pour ajouter, supprimer ou modifier des flux dans les tables de flux sur le commutateur [3].

Envoi le paquet : utilisé pour envoyer des paquets à partir d'un port particulier sur le commutateur.

Demande/réponses de barrière : utilisé pour recevoir des notifications d'opérations terminées à partir du commutateur. Si une demande de barrière est reçue du contrôleur sur le commutateur, le commutateur doit terminer toutes les tâches en attente avant de recevoir la demande. Il crée une barrière dans le commutateur afin qu'il ne puisse passer à la tâche suivante que lorsqu'il a terminé ce qu'il a sous la main. Dès que le commutateur termine ses tâches, il renvoie une réponse de barrière au contrôleur et commence à travailler sur ses tâches suivantes.

Fonctionnalités : il s'agit d'une demande de fonctionnalité au commutateur après l'établissement d'une connexion sécurisée entre le contrôleur et le commutateur. Le commutateur répond uniquement au contrôleur avec les fonctionnalités et capacités qu'il peut prendre en charge.

Configuration : utilisé par le contrôleur pour obtenir et définir les paramètres de configuration dans le commutateur.

1.7.2 OpFlex

OpFlex est un protocole propriétaire Cisco, conçu pour faciliter les communications entre le contrôleur SDN et l'infrastructure (commutateurs et routeurs) , OpFlex est un protocole de l'Interfaces Sud pour un réseau définie par logiciel qui ressemble beaucoup à OpenFlow mais il est assez différent en termes de capacités , son objectif est de créer une norme permettant d'appliquer des politiques sur des commutateurs/routeurs physiques et virtuels dans un environnement multifournisseur.

1.7.3 ForCES

Le protocole ForCES (Forwarding and Control Element Separation) est un protocole maître-esclave , Comprend à la fois la gestion du canal de communication et les messages de contrôle , lélément le plus important de l'architecture ForCES est le LFB (Logical Function Block). Le LFB est un bloc fonctionnel bien défini réside sur les FE (l'élément de transfert) qui est contrôlé par les CE (l'élément de contrôle) via le protocole ForCES. Le LFB permet aux CE de contrôler la configuration des FE et la façon dont les FE traitent les paquets, la véritable force de ForCES réside dans son modèle qui permet la description de nouvelles fonctionnalités de chemin de données sans modifier le protocole entre la couche de contrôle et la couche de transmission , En revanche, OpenFlow nécessite la mise en oeuvre d'un protocole défini sur le contrôleur et le plan de transfert (par exemple, lagent OpenFlow sur le périphérique réseau)[5].

1.8 Quelques Contrôleurs SDN

Le contrôleur SDN permet d'implémenter un changement sur le réseau en traduisant une demande globale ou une suite dopérations sur les équipements réseau .Il existe de nombreux contrôleurs SDN, libres et commerciaux. Les contrôleurs bien connus sont les suivants :

1.8.1 NOX

Initialement développé chez Nicira, NOX est le premier contrôleur OpenFlow, il est Open-source et ne contient que la prise en charge de C++.

FIGURE 1.6 – Logo du contrôleur NOX

1.8.2 POX

Il s'agit d'un contrôleur open source écrit en Python qui fournit un cadre pour développer et tester des contrôleurs OpenFlow comme NOX, mais les performances de POX sont nettement inférieures à celles des autres contrôleurs, donc ne convient pas au déploiement d'entreprise.

FIGURE 1.7 – Logo du contrôleur POX

1.8.3 Beacon

Beacon est un contrôleur Java connu pour sa stabilité. Créé en 2010, il est toujours maintenu et est utilisé dans plusieurs projets de recherche. Ses performances en font une solution fiable pour une utilisation en conditions réelles. Ce contrôleur a également été utilisé dans d'autres projets tels que Floodlight ou OpenDaylight.

FIGURE 1.8 – Logo du contrôleur Beacon

1.8.4 Floodlight

Floodlight est un contrôleur OpenFlow open source basé sur Java et alimenté par BigSwitch Networks. Sous licence Apache. Il est facile à installer et offre d'excellentes performances, Floodlight est plus une solution complète .

FIGURE 1.9 – Logo du contrôleur Floodlight

1.8.5 OpenDaylight

OpenDaylight est un projet de la Fondation Linux pris en charge par l'industrie. C'est Un framework open source pour faciliter l'accès au réseau défini par logiciel (SDN), ODL est un logiciel basé sur Java et pris en charge par lindustrie, géré par le consortium Linux Foundation avec près de 50 entreprises membres, dont Brocade, Cisco, Citrix, Dell, Ericsson, HP, IBM, Juniper, Microsoft et Red Hat. La mission d'ODL est de créer une communauté collaborative qui partage et contribue au succès et à ladoption du SDN.

FIGURE 1.10 – Logo du contrôleur OpenDaylight

1.9 Inconvénients des réseaux SDN

Malgré tous ses avantages, l'approche SDN a des inconvénients importants dans l'utilisation des réseaux. Nous allons en citer quelques uns :

- Un problème de dotation qui veut dire une reformation du personnel ou recrutement de nouvel agent pour bien se familiariser à son utilisation .
- L'échange d'informations entre les sociétés d'applications, les serveurs et les équipes réseau doit être fondamentalement réorganisé. Comme mentionné ci-dessus,Il nécessite une modification de l'ensemble de l'infrastructure réseau pour mettre en œuvre le protocole SDN et le contrôleur SDN. Cela nécessite donc une reconfigurations complète du réseau. Cela augmente les coûts en raison de la reconfiguration.

On peut noter qu'il s'agit d'une nouvelle technologie avec de nouveaux protocoles pouvant entraîner des vulnérabilités et des faiblesses.[34]

1.10 Conclusion

Ce chapitre a introduit le concept de SDN en expliquant les différentes couches de l'architecture de SDN et les API utilisées. Le but était de montrer que cette approche permettait de rendre les réseaux programmables,

évolutifs et faciles à jouer. Les protocoles OpenFlow et OpFlex standard de l'industrie ont aidé à atteindre le SDN.

Chapitre 2

L'IoT et SDN

2.1 Introduction

Définit par l'IUT(l'Union International des Télécommunications), l'IdO (l'Internet des objets) ou bien (Internet of things) IoT est une " infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution " (wikipedia.org).
Les domaines d'applications de l'IoT utilisent divers schémas de réseaux sophistiqués et de nouveaux protocoles émergent aussi spécifiquement pour ce genre de technologie [17], l'IoT devient rapidement complexe en terme d'architectures et de topologies réseau dû au grand nombre dobjets connectées, ce qui provoque aussi une grande difficulté dans le déploiement et la gestion de ces réseaux, et génèrent une grande quantité de données. Dans ce cas, il est donc nécessaire d'adapter de nouvelles architectures ainsi que de nouveaux protocoles d'accès, afin doffrir une certaine orchestration pour la gestion du réseau IoT.
Dans ce chapitre nous allons :
Introduire la technologie IoT, puis caractériser les différents avantages et inconvenants ainsi que les exigences de cette nouvelle technologie en terme de gestions, contrôle et flexibilité du réseau, ainsi montrer les différentes caractéristiques de l'IoT (architecture protocolaire, technologies de communications), pour enfin introduire quelques solutions SDN, qui répondent aux exigences principales de l'IoT.

2.2 L'IoT en générale

Après internet l'IoT (Internet of Things), la révolution la plus importante dans le monde des technologies de l'information. Daprès [8],l'IoT est " Un réseau ouvert et complet d'objets intelligents qui ont la capacité de s'auto-organiser, de partager des informations, des données et des ressources, de réagir et d'agir face aux situations et aux changements de l'environnement vise à unifier tout dans notre monde sous une infrastructure commune, nous donnant non seulement le contrôle des choses qui nous entourent, mais aussi nous tenant informés de l'état de ces choses. " via le réseau internet. l'IoT comme son nom l'indique, « Internet »qui est un système mondial de réseaux informatique inter-connectés qui utilise les protocoles standard (TCP/IP), qui a pour but de servir la population mondiale. Et « Objets »qui représentent toutes sortes dobjets physiques qui nous entourent, converti en leurs équivalents virtuels.
Cette technologie est considérer comme lextension dinternet, car elle offre un plus grand nombre de noeuds (Tout périphérique pouvant e recueillir et de transmettre des données), que ce soit des périphériques de communication (Smartphones, PCs, etc.), mais aussi des objets généralement des capteurs intégrés.

Un nouveau type réseaux est apparu avec linternet des objets, celui-ci est un réseau de capteurs qui utilisent une connectivité à faible puissance et à faible débit de données, ce sont les réseaux de capteurs sans fil ou bien Wireless Sensors Network (WSN). Les WSN gagnent en popularité car ils peuvent accueillir beaucoup plus de noeuds de capteurs tout en conservant la durée de vie de la batterie et couvrant de grandes surfaces.

Le champ dapplication de l'IoT est quasi illimité, ce qui va permettre de rendre l'environnement intelligent et favorable à toute activité humaine [15]. L'IoT est aujourd'hui exploité dans plusieurs domaines de notre vie quotidienne (Agriculture, santé, domotique, transport, etc.).
Voici quelque exemples de dispositifs IoT utilisé dans notre vie quotidienne :

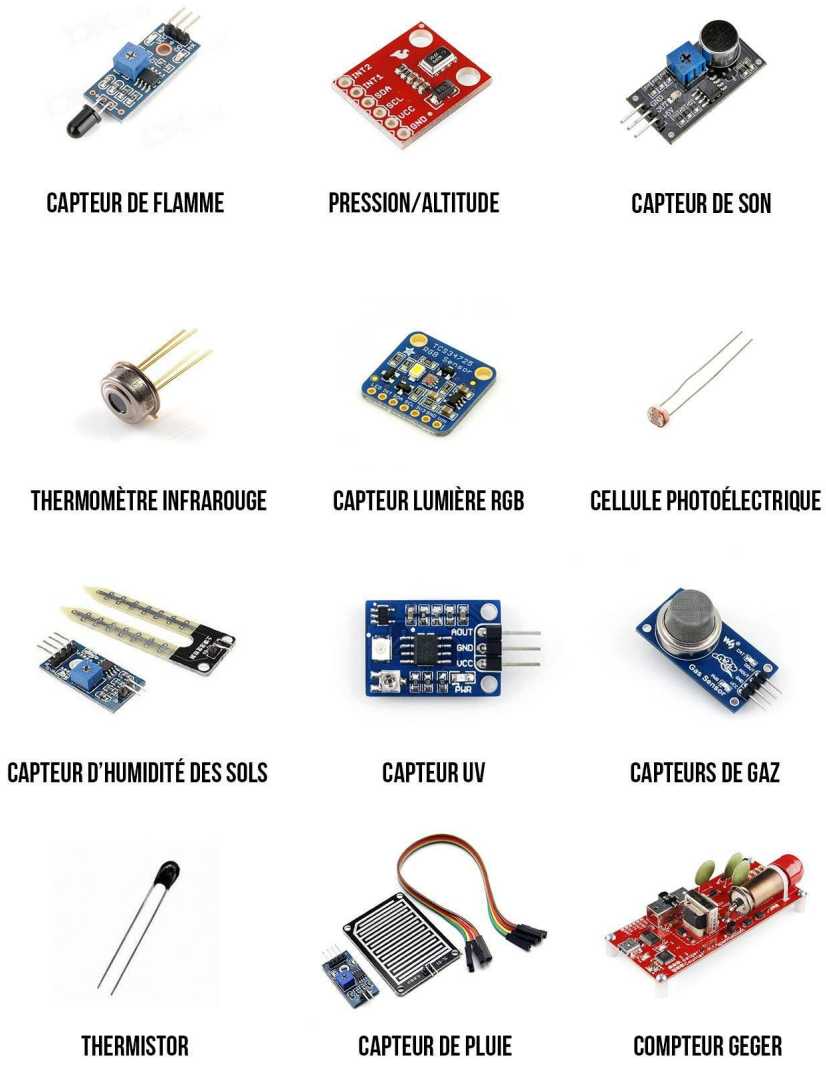

FIGURE 2.1 – Liste de quelques micro-capteurs utilisés dans les appareils IoT

Ces derniers ne se connectent pas directement, ils nécessitent un dispositif de transmission que ce soit, une antenne RFID, une carte réseau sans fil, etc.
La figure suivante montre des dispositifs IoT qui ont la capacité de se connecter directement une fois mis en marche, ils sont généralement composés de plusieurs micro-capteurs.

FIGURE 2.2 – Capteurs IP utilisés dans l'IoT

2.3 Architecture protocolaire de l'IoT

Limmensité de l'IoT et sa capacité à interconnecter plusieurs milliards de réseaux hétérogènes constitue un problème au niveau architecturale de cette technologie, par conséquent plusieurs architectures ont été proposées, pour définir la manière dont les divers technologies communiquent les unes aux autres. De même, mieux gérer l'aspect d'hétérogénéité des différents réseaux qui constituent l'IoT. Daprès (Gestion dynamique et évolutive de règles de sécurité pour l'Internet des Objets), " De nombreux modèles d'architecture sont en cours de développement, comme le modèle NIST Smart Grid (National Institute of Standards and Technology), le modèle ITU-T (International Telecommunication Union - Telecom), le modèle M2M de l'ETSI (European Telecommunications Standards Institute) ou la référence architecturale du projet IoT-A de l'Union Européenne et les autres travaux liées à l'IETF, W3C etc. ".

Parmi ces architectures, la plus populairement admise daprès [16] est la conception de l'IETF (Internet Engineering Task Force), qui travaille sur des normes de gestion automatisée des réseaux qui comme son nom l'indique, vise à améliorer et à rendre plus efficace la gestion des réseaux à mesure qu'ils augmentent en taille et en complexité. (ietf.org) La figure ci-dessous montre le concept architectural qui est composé de six couches à savoir : application, transport, réseau, adaptation, MAC et physique présenter par lIETF pour l'IoT

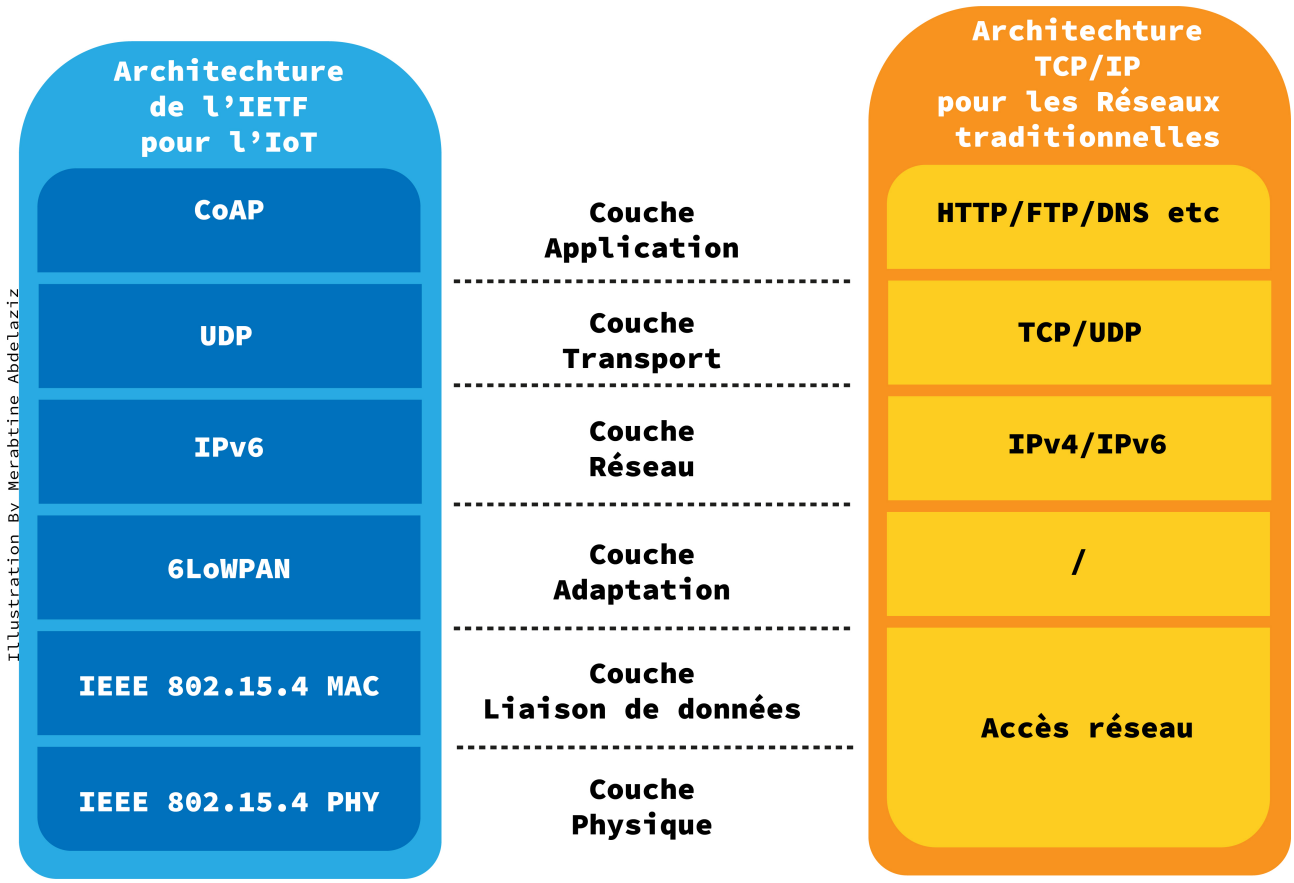

FIGURE 2.3 – Comparaison entre le modèle TCP/IP et l'architecture IoT de l'IETF

2.3.1 CoAP

CoAP (Constrained Application Protocol), est un protocole de transfert Web optimisé pour les périphériques et réseaux contraints, utilisé dans les réseaux de capteurs sans fil pour former l'Internet des objets.(wikipedia.org)

2.3.2 UDP

Le User Datagram Protocol (UDP) est l'un des principaux protocoles de télécommunication utilisés par Internet. Il permet la transmission de données (sous forme de datagrammes) de manière très simple entre deux entités et aucune communication préalable n'est requise pour établir la connexion. (UDP) est considéré comme une bonne alternative pour les communications des appareils IoT. Contrairement à TCP, il n'a pas de règle de fiabilité stricte et il convient aux applications à faible latence et à faible consommation [13]. UDP est principalement utilisé dans Voix sur IP (VoIP) et applications de streaming. UDP peut être bénéfique pour les communications IoT qui se concentrent sur les communications à faible latence plutôt que sur la fiabilité.

2.3.3 IPv6

De nos jours, l'un des points les plus importants pour les applications IPv6 (Internet Protocol version 6) est l'Internet des objets (IoT). Dapres [18], plus de 75 milliards d'appareils IoT seront utilisés dans le monde d'ici 2025. par conséquent les adresses IPv4 deviennent insuffisantes pour prendre en charge cette croissance. Ce manque d'espace d'adressage IPv4 a été le facteur le plus décisif pour la transition vers l'IPv6. Ce dernier est donc conçu pour être le successeur de l'IPv4. Cette nouvelle version du Protocol IP possède un plus grand espace d'adressage de 128 bits pour un total de 340 undécillions (c'est-à-dire 340 suivi de 36 zéros) d'adresses disponibles (netacad.com).

2.3.4 6LoWPAN

IPv6 Low power Wireless Personal Area Networks (6LoWPAN) est un standard de l'IETF qui sur l'échelle architectural, s'occupe de l'adaptation entre la couche IEEE 802.15.4 et la couche Réseau (IPv6). Comme indiqué dans [2] " 6LoWPAN définit comment exécuter (IPv6) sur un faible débit de données, faible consommation dans les réseaux de type LoWPAN (Low power Wireless Personal Area Networks) ".

2.3.5 IEEE 802.15.4

Le 802.15.4 est un protocole de communication défini par l'IEEE (Institute of Electrical and Electronics Engineers). Il est destiné aux réseaux sans fil de la famille des LR WPAN (Low Rate Wireless Personal Area Network), qui utilisent des dispositifs caractérisés par leur faible consommation, ainsi qu'une faible portée, et un faible débit. Ce protocole est lui-même divisé en 2 couches **IEEE 802.15.4 (PHY)** et **IEEE 802.15.4 (MAC)**, la différence entre ces deux dernières cest que :

- La couche physique (PHY) contient l'émetteur/récepteur radio (RF), avec un mécanisme de contrôle de bas niveau (contrôle de la qualité du signal, détection d'énergie et CCA) [23].
- La couche MAC est caracterisé par la gestion des balises, l'accès au canal, la gestion des GTS (Guaranteed Time Slot), la validation des trames, etc. La couche MAC permet aussi d'utiliser des mécanismes de sécurité liés à l'implémentation des applications comme le CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), qui est un protocole utilisé pour réduire le risque de collisions entres les trames et le risque de brouillage. (wikipedia.org)

2.4 Technologies de communication dans l'IoT

Les réseaux IoT sont utilisés pour fournir aux objets une connexion à Internet qui permet le retour d'informations. Divers technologies de communication sont disponibles sur le marché à cet effet, ces technologies ne sont pas les mêmes et nont pas les même caractéristiques en matière de couverture du réseau ou du coût du service. Par conséquent, il peut être difficile pour les entreprises qui se lancent dans l'IoT de choisir le réseau de communication qui convient le mieux à leurs objectifs.
La plupart des objets nécessitent une connectivité aux passerelles IoT. Cela peut prendre la forme d'un réseau local (LAN) tel que les connexions Ethernet et Wi-Fi ou réseau personnel (PAN) tels que ZigBee, Bluetooth et Ultra Wide band (ULB). Pour les capteurs qui ne nécessitent pas de connectivité au capteur agrégateurs (passerelle IoT), leur connectivité au serveurs/applications peut être fournie en utilisant les Réseaux (WAN) tel que les réseaux cellulaires GSM, GPRS, LTE, etc [10]. Voici quelques exemples des différents Technologies d'accès utilisé dans l'IoT : [16]

Technologie	Standard	Fréquences	Portée	Débit
Wifi	EEE 802.11	2.4/5 GHz	50-125m	150Mbps-1.3Gbps
Bluetooth LE	4.2.5.0	2.4GHz	50-240m	1-50Mbps
LoRaWAN	LoRaWAN	Variés	2-15Km	0.3-50Kbps
ZigBee	IEEE 802.15.4	2.5GHz	10-100m	250Kbps
Z-Wave	IUT-T G.9959	900MHz	30m	9.6/40/100Kbps
Cellulaire	GSM GPRS 2G 3G 4G LTE	900/1800/ 1900/2100MHz	35Km GSM 200 Km HSPA	35-170Kbps (GPRS) 120-384Kbps (EDGE) 384K-2Mbps (UMTS) 600k-10Mbps (HSPA) 1-10Mbps (LTE)
NFC	ISO / IEC 1800-3	13.56MHz	10cm	100-420Kbps
Sigfox	Sigfox	900MHz	10-50Km	10-100bps
Neul	Neul	900MHz (ISM) 458MHz (UK) 470-790MHz (White space)	10Km	100Kbps
LowPAN	RFC 6282	2.4GHz	30-100m	200Kbps
LTE-M			15Km	15Kbps
Clean state IoT			15Km	15Kbps-1Mbps

TABLE 2.1 – Technologies de Communications pour l'Internet des Objets [16]

2.5 Architecture IoT pour les solutions SDN

L'Internet des objets (IoT) représente l'état actuel et futur d'Internet. La croissance rapide de cette technologie produit Un grand nombre d'objets qui sont connectés à Internet, ces derniers génèrent une énorme quantité de données et d'informations qui nécessite beaucoup d'efforts en matière de traitement pour les transformer en informations utiles. Collecter, Stocker, gérer, contrôler et sécuriser ces mégas donnés sont considéré comme des enjeux cruciaux, si l'on veut tout connecter à Internet de manière utile et pratique. Alors, l'organisation et le contrôle de ce grand volume de données nécessitent de nouvelles idées dans la conception et la gestion du réseau IoT pour accélérer et améliorer ses performances.

Les réseaux définis par logiciel (SDN) sont une nouvelle solution qui est apparu récemment pour cacher toute la complexité de l'architecture système traditionnelle en faisant abstraction des dispositifs de l'IoT (plan de données) de tous les contrôles et opérations de gestion (plan de contrôle), en les plaçant à l'intérieur d'une couche logicielle intermédiaire (Contrôleur SDN) [7].

— **Le plan de contrôle** offre bien évidement le contrôle, la gestion, la sécurité et dirige les données en utilisant un contrôle de flux. Cela permet aux administrateurs réseau de travailler efficacement avec les SDN et de gérer les réseaux IoT.

— **le plan de données** ne fait que avancer le trafic vers sa destination.[21].

Pour les solutions SDN, larchitecture IoT la plus couramment utilisé d'après [16], est comme le montre la figure suivante.

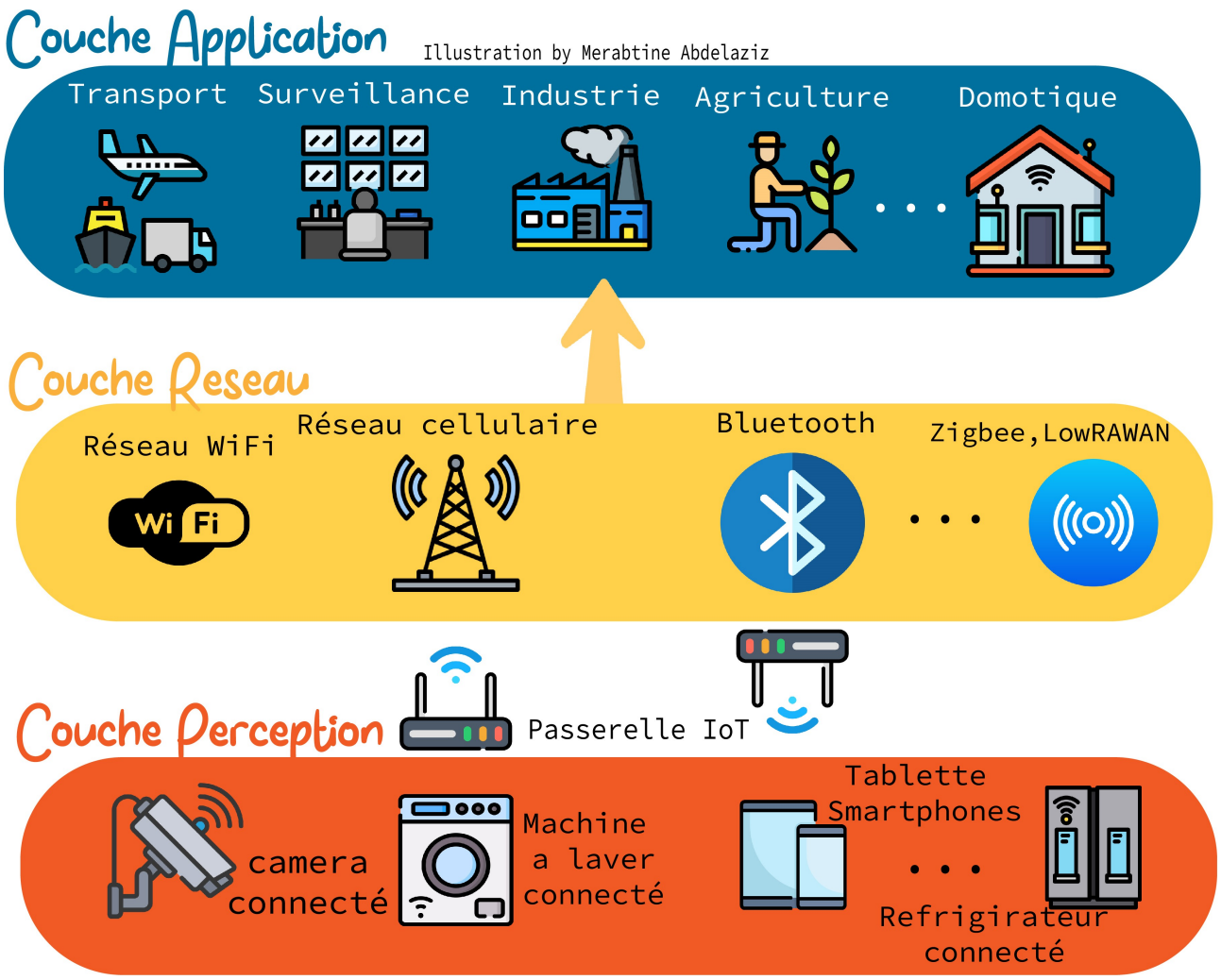

FIGURE 2.4 – Architecture IoT pour les solutions SDN

2.5.1 La couche Perception

Cette couche de reconnaissance est la couche physique de l'architecture IoT. Des capteurs et des systèmes intégrés sont principalement utilisés. Ils collectent de grandes quantités de données selon les besoins. Cela inclut les dispositifs embarqués, les capteurs et les actionneurs qui communiquent avec l'environnement. Par conséquent, ces informations sont envoyées à la couche réseau, d'où provient la grande quantité de données générées par l'IoT. Par conséquent, le coût et l'autonomie des objets connectés sont à ce niveau de limitation, en dehors de la grande quantité de données qu'ils génèrent.

2.5.2 La couche Réseau

Les données obtenues à partir de ces appareils doivent être distribuées et stockées. C'est la responsabilité de cette couche de relier les objets intelligents à d'autres objets intelligents via des techniques daccès au réseau et de protocoles bien définis comme WIFI, WIMAX, 3G, 4G, LTE, etc. Et d'autre part les technologies conçu spécifiquement pour l'IoT mais beaucoup moins déployé telle que ZigBee, LoRa, Sigfox etc. Elle est également en charge de la transmission des données. La couche réseau est responsable de la connexion des objets intelligents, des périphériques réseau et des serveurs. elle est également utilisé pour la distribution et l'analyse des données des capteurs.

2.5.3 La couche Application

Cette couche permet l'interaction directe avec les utilisateurs finaux via un service ou une application particulière, et doffrir une infrastructure de développement logiciel avec des outils clés en main qui permettent dexplorer les données, faire de lanalyse avancée et de la visualisation de données, ces applications peuvent être déployés dans plusieurs domaines de la vie quotidienne comme la surveillance routière, les maison intelligentes, lagriculture intelligente, etc. Et cest dans cette couche que toutes les décisions de contrôle, sécurité et de gestion des applications sont prises.

2.6 Solutions SDN pour l'IoT

Dans les sections précédentes nous avons vue l'importance d'adopter de nouvelles approches telles que le SDN pour une meilleure gestion d'un réseau IoT. Et nous avons vu larchitecture dun réseau IoT pour une solution SDN pour nous donner en fin le SDIoT(Software Defined Internet of Things). Les recherches liées au SDIoT sont assez répandues. Cela a pour but de masquer toutes les complexités de la fonctionnalité de gestion et de contrôle des ressources système et de la sécurité, au sein des réseaux IoT. Par exemple dans [16], [6], on nous fait part de quelques solutions de gestion pour la sécurité définit par logiciel SDSec (Software Defined Security) , utilisé dans les réseaux SDIoT. Autre exemple, dans l'article [7] les auteurs proposent une architecture SDIoT avec sécurité et stockage définies par logiciel SDStore (Software Defined Storage) et SDSec (Software Defined Security). [21] étudie l'architecture SDIoT dune manière détaillé, et souligne les défis les plus importants, qui comprend à la fois des défis qui ont été partiellement relevés et les autres qui continuent d'ouvrir de nouvelles recherches comme :

- **La fiabilité logicielle** : Toute infrastructure SDN repose sur le contrôleur central, Si ce dernier échoue, tout le réseau s'effondre. Par conséquent, les fournisseurs/développeurs doivent exploiter les principales fonctions du contrôleur, s'ils veulent que la fiabilité du réseau soit optimale.
- **Lévolutivité** : le contrôleur SDN doit avoir la capacité de gérer une charge de travail toujours croissante.
- **Le développement d'interface de bas niveau** : pour les configurations des commutateurs OpenFlow.
- **Performances et sécurité** : l'IoT collecte de plus en plus de données personnelles et impose donc des exigences plus élevées en matière de protection de la vie privée.

Nous retrouvons aussi CORAL-SDN, une solution proposé par [12], qui est un protocole et une stratégie de contrôle des réseau SDN destinés pour des applications IoT, dans le cas des WSN (Wireless Sensors Network). Daprès ce dernier CORAL-SDN :

- Utilise des mécanismes de contrôle centralisés intelligents pour ajuster dynamiquement les fonctionnalités du protocole
- Prend en charge l'élasticité aux exigences difficiles des réseaux WSN.
- Maintient une architecture réseau évolutive.
- Présente une gestion et un fonctionnement améliorés du réseau en termes de performances et d'utilisation des ressources.

Dans [17], propose la mise en oeuvre d'un contrôleur sensible aux applications (AAC)(Applications Aware Contrôler) qui a pour but d'établir des mécanismes de communication pour un réseau défini par logiciel (SDN) qui intègre des appareils IoT, et de simplifier la complexité globale du système en déplaçant une partie de la logique dans le AAC.

2.7 Conclusion

Dans ce chapitre nous avons introduit la technologie IoT, son architecture protocolaire, ainsi que les différentes technologies de communication utilisées. Nous avons vue aussi les avantages qu'apporte cette technologie et son influence globale sur le développement de plusieurs domaines de vie quotidienne, mais aussi nous avons introduit l'un des principaux problèmes de l'IoT, et les causes principales qui nous pousse à adopter de nouvelles

approches telles que les réseaux SDN qui ont pour but de réduire la complexité de gestion et de contrôle d'une ou de plusieurs infrastructures réseau, dans notre cas l'IoT d'où le terme SDIoT.

Chapitre 3

Préparation des prérequis pour une solution SDN

3.1 Introduction

Après avoir introduit les réseaux définit par logiciel (SDN), et l'Internet des objets (IoT), puis cité les différèrent problèmes que rencontre l'IoT et les solutions que peut apporter le SDN pour une meilleurs version d'internet des objets. Ce chapitre serra axé sur les logiciels nécessaires pour la simulation d'une solution SDN pour l'internet des objets, nous allons commencer par présenter :

- OVS (Open Virtual Switch) et son fonctionnement, ainsi que ses composantes et son rôle dans la simulation.
- L'émulateur de réseaux (Mininet), ainsi que ses différentes fonctionnalités.
- Le contrôleur SDN que nous utiliserons serait OpenDaylight, nous allons donc le présenter, et citer ces différentes fonctionnalité et interfaces, aussi bien que son architecture.
- Par la suite, nous allons introduire OpenFlow Manager qui servira d'application de gestion et de contrôle du réseau.

3.2 Open vSwitch

Open vSwitch (OVS) est un commutateur virtuel multicouches de qualité production sous licence open source Apache 2.0. Il est conçu pour permettre une automatisation massive du réseau via une extension programmable, tout en prenant en charge les interfaces et protocoles de gestion standard (par exemple, NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). En outre, il est conçu pour prendre en charge la distribution sur plusieurs serveurs physiques similaires au vswitch distribué vNetwork de VMware ou au Nexus 1000V de Cisco [30].

3.2.1 Composantes d'Open vSwitch

Comme le montre la figure 3.1, OVS est principalement composé des éléments suivants [9] :

- Un serveur de base de données qui stocke diverses configurations du commutateur, celui-ci permet de garder une continuité dans le comportement du switch au-delà du redémarrage de celui-ci, on peut l'atteindre via la commande **ovsdb-server**
- Le répertoire d'OVS est le processus principal de fonctionnement du commutateur et permet la communication avec le contrôleur via OpenFlow, ou la récupération des données dans la base de données, on peut l'atteindre via la commande **ovs-vswitchd**
- Le Kernel est le coeur du système d'exploitation et la partie qui commute les paquets.

La communication des utilisateurs avec ces composants est une étape importante dans la mise en œuvre d'Open vSwitch on peut l'atteindte via la commande **openvswitch.ko** (Module Kernel). Trois outils d'es-

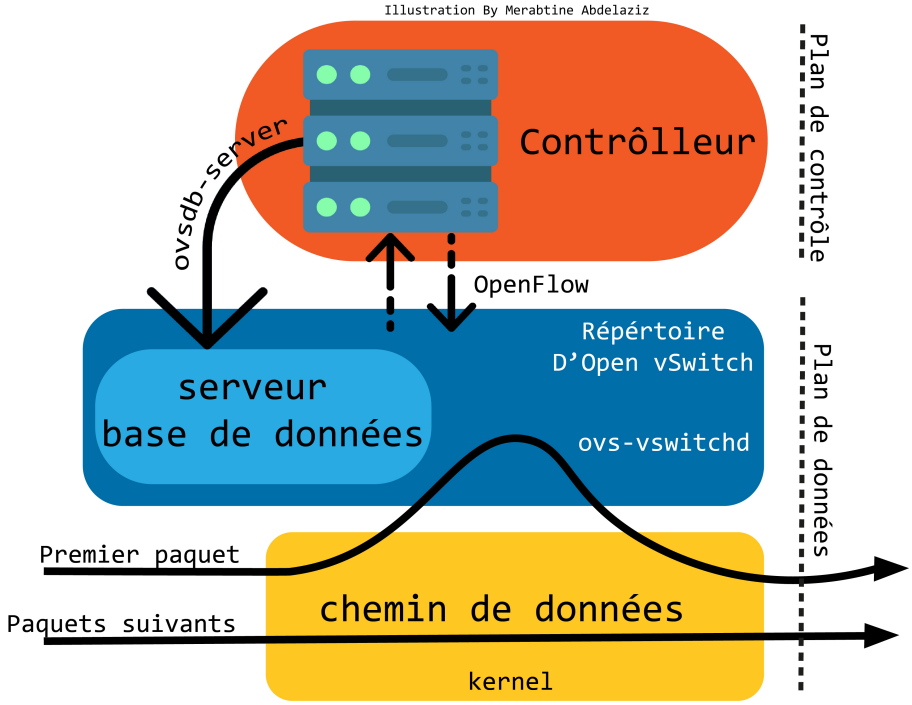

FIGURE 3.1 – Les composants et les interfaces d'Open vSwitch

pace utilisateur sont disponibles pour configurer et surveiller l'état du commutateur **ovs-vsctl**, **ovs-ofctl** et **ovs-dpctl**, chacun de ces outils fournit un ensemble de commandes pour interagir avec le commutateur.

3.2.2 Fonctionnement d'Open vSwitch

Open vswitch permet de créer des bridges virtuels entre plusieurs VMs pour les relier ensemble tout en garantissant un isolement complet entre les bridges [9]. Cela permet de créer plusieurs réseaux indépendants sur une même machine physique. Pour configurer l'OVS on communique avec ses différentes composantes en utilisant les outils de l'espace utilisateur cités dans la section précédente.
Dans ce qui suit, nous avons présenté les commandes d'Open vSwitch utilisé dans une architecture OpenFlow.[31]

- Pour créer un pont et par défaut une interface du même nom (br0) on utilise la commande **ovs-vsctl add-br br0**.
- En ce qui concerne la création de ports/interfaces, la commande à soumettre est **ovs-vsctl add-port br0 eth1** On peut éventuellement y attribuer un Vlan en ajoutant loption tag=10 ou 10 est le numéro du vlan
- Pour connaître l'état global du switch il faut entrer l'instruction suivant : **ovs-vsctl show**

```
mininet> sh ovs-vsctl show
97060324-da2c-4cc5-ac47-5e2006b159a7
    Bridge "s5"
        Controller "ptcp:6658"
        Controller "tcp:192.168.56.117:6633"
        fail_mode: secure
        Port "s5-eth3"
            Interface "s5-eth3"
        Port "s5-eth2"
            Interface "s5-eth2"
        Port "s5"
            Interface "s5"
                type: internal
        Port "s5-eth1"
            Interface "s5-eth1"
```

FIGURE 3.2 – Affichage de la commande show dOVS

Pour l'architecture SDN, nous devons ensuite connecter le switch à un contrôleur, pour ce faire on introduit **ovs-vsctl set-controller br0 tcp :192.168.56.113 :6633** ou « 192.168.56.113 »est ladresse IP du contrôleur et 6633 le port d'écoute du protocole OpenFlow. A partir de là pour communiquer avec OpenFlow l'outil privilégié est **ovs-ofctl**.
ovs-ofctl O OpenFlow 13 dump-flows br0 Affichera les différents flux installés sur le switch.
L'outil permet aussi d'ajouter des flux directement sur le switch sans passer par un contrôleur avec la commande **add-flow**. Dans lexemple présent on ajoute un simple flux
ovs-ofctl O OpenFlow13 add-flow br0 in_port=1,actions=output :2 où tous les paquets reçus dans le port 1 sont renvoyés par le port 2. On peut vérifier que le flux a été bien ajouté en repassant par la commande **dump-flows**.

3.3 Mininet

Mininet est un émulateur de réseau qui crée un réseau d'hôtes virtuels, de commutateurs, de contrôleurs et de liens. Les hôtes Mininet exécutent un logiciel réseau Linux standard et ses commutateurs prennent en charge OpenFlow pour un routage personnalisé très flexible et une mise en réseau définie par logiciel.

" Mininet prend en charge la recherche, le développement, l'apprentissage, le prototypage, les tests, le débogage et toute autre tâche qui pourrait bénéficier d'un réseau expérimental complet sur un ordinateur portable ou un autre PC. " [24]

3.3.1 Ligne de commande Mininet

Grâce à la commande **mn** et la multitude doptions offertes par lémulateur, il est possible de réaliser maintes formes de topologies et de réseaux virtuels (Mininet, s.d) :

- Loption **- -topo** permet de créer 3 types de topologies de base : tree (Arbre), linear (linéaire) et single (étoile).
- Pour connecter notre topologie à un contrôleur loption adéquate est **- -controller** dans lexemple dun contrôleur distant on ajoute **remote, ip=192.168.56.113**.

3.3.2 Créer une topologie avec Mininet

L'émulateur Mininet fournit une interface de programmation Python. L'une de ces utilisations consiste à créer une topologie personnalisée en définissant des propriétés arbitraires d'éléments de réseau.Une telle approche offre l'avantage de maximiser le temps et les ressources disponibles. Dans la figure 3.3 on illustre avec un exemple

la structure générale d'un script python comprenant les informations du réseau voulu. Dans le cas présent nous avons généré 3 switchs, une passerelle et 2 hôtes auxquelles nous avons alloué différents identifiants. Le fichier python doit être sauvegardé dans le répertoire **mininet/custom/**, de plus pour pouvoir faire appelle à la topologie par loption **- -custom** dernière ligne du script est primordiale, en gros nous y avons définis la topologie par le nom « exemple ». La commande complète pour générer cette topologie devra comporter le chemin daccès du script et le nom de la topologie en question

sudo mn - -custom mininet/custom/test.py - -topo=exemple

```
"""Custom topology example

Two directly connected switches plus a host for each switch:

   host --- switch --- switch --- host

Adding the 'topos' dict with a key/value pair to generate our newly defined
topology enables one to pass in '--topo=mytopo' from the command line.
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

FIGURE 3.3 – Exemple d'un script Python dune topologie personnalisée

3.4 Opendaylight

OpenDaylight (ODL) est un contrôleur / framework SDN open source, hébergé par la Linux Foundation. C'est l'un des contrôleurs SDN (open source) les plus populaires en ce moment, ODL est une plate-forme ouverte modulaire pour la personnalisation et l'automatisation de réseaux de toute taille et d'échelle. Le projet ODL est né du mouvement SDN, avec une orientation claire sur la programmabilité du réseau, il est utilisé pour obtenir

une administration centralisée du réseau.
ODL est le contrôleur qui est compatible avec la plus grande variété des protocoles ceci avec les réseau traditionnelle ou les réseau software-defined

3.4.1 Distributions dOpenDaylight

Depuis la sortie de la première version du contrôleur en 2014, il y a eu sept autres versions, chacune avec son propre lot D'amélioration..
le tableau suivant représente une liste non exhaustives des différentes versions du dOpenDaylight :

Nom de la version	Date de sortie
HYDROGEN	Février 2014
HELIUM	Octobre 2014
LITHIUM	Juin 2015
BERYLIUM	Février 2016
BORON	Novembre 2016
CARBON	Juin 2017
NITROGEN	Septembre 2017
OXYGEN	Mars 2018
FLOURINE	Juin 2019
NEON	Décembre 2019
SODIUM	Août 2020
MAGNESIUM	Juillet 2020
ALUMINIUM	Novembre 2020

TABLE 3.1 – Versions dOpenDaylight [32]

Lithium est la version utilisée dans ce projet et sa plate-forme de service est développée avec Apache Karaf pour un déploiement et une gestion faciles , Depuis la base du contrôleur, l'utilisateur peut définir diverses fonctions requises. Cette version se concentre principalement sur la programmabilité du réseau

3.4.2 Structure Opendaylight

ODL prend en charge une architecture en couches avec des points d'intégration clairs et des API qui permettent aux utilisateurs finaux et aux fournisseurs de réseaux de participer aux puissantes capacités SDN d'ODL. L' interface sud garantit que les technologies et le matériel réseau de divers fournisseurs peuvent être exploités à l'aide d'ODL. L' interface nord fournit des API pour les utilisateurs finaux et d'autres technologies cloud telles qu'OpenStack. la figure 3.4 capture l'architecture d'ODL

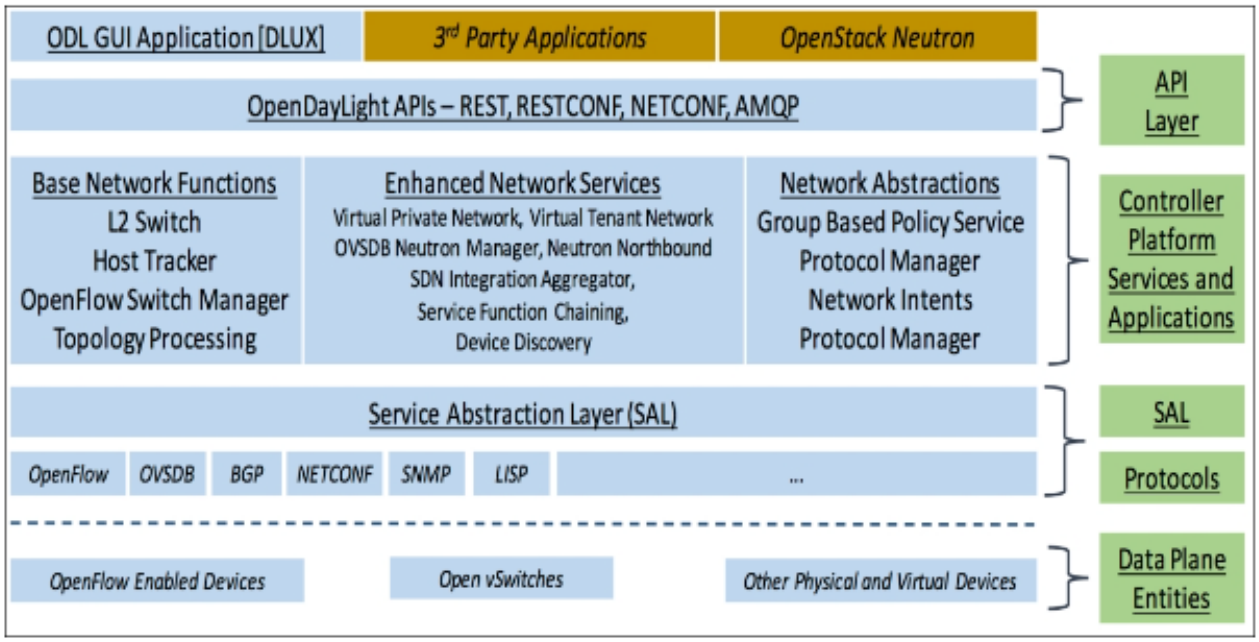

FIGURE 3.4 – Architecture simplifiée d'Opendaylight [4]

La plate-forme du contrôleur

ODL est un contrôleur modulaire, flexible, la plate-forme du contrôleur représente la couche principale de son architecture qui permet l'abstraction SDN . Cette couche expose les API NB (North Bound) ouvertes aux applications réseau pour contrôler et gérer les éléments physiques et virtuels au sein du réseau. Il comprend également les fonctions de service de réseau de base BNSF (Base Network Service Fonction), La collecte des statistique des fonctions du réseaux et la couche d'abstraction de service (SAL), qui sont couvertes dans le texte suivant.[27]

La collecte des statistique des fonctions du réseaux : les BNSF sont responsables de la collecte de statistiques, d'informations sur les éléments de l'ensemble du réseau et de leurs capacités.Ils exposent les API NB aux applications pour accéder aux informations et statistiques collectées . Actuellement, la plate-forme ODL dispose des services réseau intégrés suivants :

- Topology Manager stocke les informations sur les commutateurs gérés dans la sous-arborescence opérationnelle de la topologie
- Statistics Manager collecte des informations statistiques à partir des commutateurs gérés
- Switch Manager stocke des détails sur les commutateurs et leurs ports pour identifier les commutateurs découverts
- Forwarding Rules Manager (FRM) vérifie les mises à jour de flux,résout leurs conflits et les valide
- Le gestionnaire d'inventaire maintient une base de données et la mis à jour des commutateurs découverts
- Host Tracker suit l'emplacement de l'hôte final dans l'ensemble du réseau

Les plate-formes des fonctions de service réseau le contrôleur ODL contient des services orientés qui exécutent des tâches de mise en réseau spécifiques et d'autres extensions pour améliorer la fonctionnalité SDN. Certains de ces services sont décrits ci-dessous :

- Affinity Metadata Service fournit une API NB pour exprimer les exigences réseau des applications et communiquer la charge de travail au contrôleur
- Virtual Tenant Network (VTN) Manager crée et gère un réseau virtuel multi-utilisateur .
- L2 Switch fournit la fonctionnalité de commutation L2 et crée plusieurs services génériques réutilisables, tels que le suivi des adresses, le protocole Spanning Tree de base, la gestion modulaire des paquets pilotée par les événements et les calculs de chemin optimaux.

— Service Function Chaining (SFC) offre la possibilité de définir une chaîne de services réseau
— La politique basée sur les groupes (GBP) sépare les exigences de connectivité des applications des détails sous-jacents des éléments de réseau via un modèle de politique axé sur les applications .
— Authentification, autorisation et comptabilité (AAA) Le service est proposé pour fournir un modèle généralisé pour les fonctions AAA dans le projet ODL.[1] [26] [25]

La couche d'abstraction de service : en tant que cœur de l'ODL, la SAL (Sevice Abstraction Layer) permet à l'ODL de prendre en charge plusieurs protocoles SB (via des plugins SB) et de fournir un ensemble uniforme de services à d'autres modules et applications réseau , La découverte de périphériques est un service fourni par la SAL et utilisé par Topology Manager pour former la topologie du réseau et pour créer des fonctionnalités d'élément. La plupart des services SAL sont construits sur la base des fonctionnalités des plugins SB. le service demandé pour un commutateur donné est rempli par le SAL, quel que soit le protocole SB sous-jacent

les interfaces et protocoles SB (South bound)

Pour permettre une communication sécurisée entre le contrôleur et les éléments de réseau, les protocoles SB sont utilisés. Les éléments de réseau peuvent être gérés, configurés et surveillés par ces protocoles. ODL prend en charge plusieurs protocoles SB (via des plug-ins SB). Ces protocoles SB permettent à ODL de prendre en charge des réseaux hétérogènes et d'assurer l'interopérabilité avec d'autres technologies et entre d'autres fournisseurs. Certains des plugins SB pris en charge (qui implémentent les protocoles SB) sont décrits ci-dessous

— OpenFlow Plugin implémente les spécifications du protocole OF au fur et à mesure de son évolution.
— Plugin Open vSwtich Database (OVSDB) est le protocole de gestion qui gère et configure les switches ouverts.
— Le plugin SNMP a proposé de développer un plugin SB de protocole de gestion de réseau simple (SNMP) pour gérer les commutateurs Ethernet standard.
— Les plugins BGP-LS/PCEP implémentent le protocole BGP (Border Gateway Protocol) basé sur Java et le protocole PCEP (Path Computation Element Protocol) .
— Plugin de protocole de configuration réseau (NETCONF) développé pour permettre à l'ODL de gérer et de configurer les éléments de réseau prenant en charge le protocole NETCONF.[20]

Les applications et services du réseau

Sur la couche supérieure d'ODL se trouvent les applications et services réseau qui contrôlent, gèrent, surveillent l'ensemble du réseau. La plupart de ces applications et services sont liés aux services de réseau de plate-forme correspondants tels que le coordinateur VTN et le gestionnaire VTN. Cette couche comprend également des services d'orchestration qui gèrent le trafic, en fonction des exigences d'environnements tels que le NVF et le cloud Certaines des applications réseau de l'ODL sont discutées

— OpenDayLight User eXperience (DLUX) est une nouvelle interface utilisateur (UI) basée sur le Web pour la deuxième version de l'ODL .
— Le coordinateur VTN est une application externe qui fournit des API REST permettant aux utilisateurs de construire VTN et coordonne les réseaux virtuels s'étendant sur plusieurs contrôleurs ODL
— SDNi Wrapper fait partie de l'application ODL-SDNi pour permettre la communication entre les contrôleurs SDN. Il utilise l'API SDNi Rest pour collecter les informations à partager entre les contrôleurs.
— La protection DDoS est une application permettant de détecter et d'atténuer les attaques par déni de service distribué (DDoS) [29] [28]

3.4.3 Openflow Manager

Le Software Defined Networking (SDN) implique une application interagissant avec un réseau (composé d'appareils spécifiques à un domaine) dans le but de simplifier les opérations ou d'activer un service. Un contrôleur est positionné entre l'application et le réseau et interagit avec les éléments du réseau (par exemple, les commutateurs) dans la direction sud en utilisant une variété de protocoles différents. Dans la direction nord, il présente une abstraction du réseau utilisant en pratique des API REST communes. Le véhicule contrôleur pour cette application est ODL. L'OpenFlow Manager (OFM) est une application qui tire parti de cette innovation pour gérer le réseau OpenFlow.[33]

Architecture de lapplication OFM

Openflow Manager est une application développée par Cisco pour son contrôleur commercialisé basé sur OpenDaylight, ce qui fait que lapplication est compatible avec ODL. Elle est faite pour une architecture (MD-SAL), offrant la possibilité de créer des flux grâce à une interface graphique accueillante. Lapplication communique avec les modèles (YANG) se trouvant dans le contrôleur à travers linterface (RESTCONF) (Les mêmes concepts que la section antérieure). L'approche est simple, en clinquant sur une case pour choisir l'action ou les champs de correspondance par exemple, on génère une ligne de code en Json, une fois l'opération validée, lapplication envoie la requête via RESTCONF au contrôleur où le SAL traduira le modèle au langage de l'équipement en question (Cisco Dev Net, 2014).

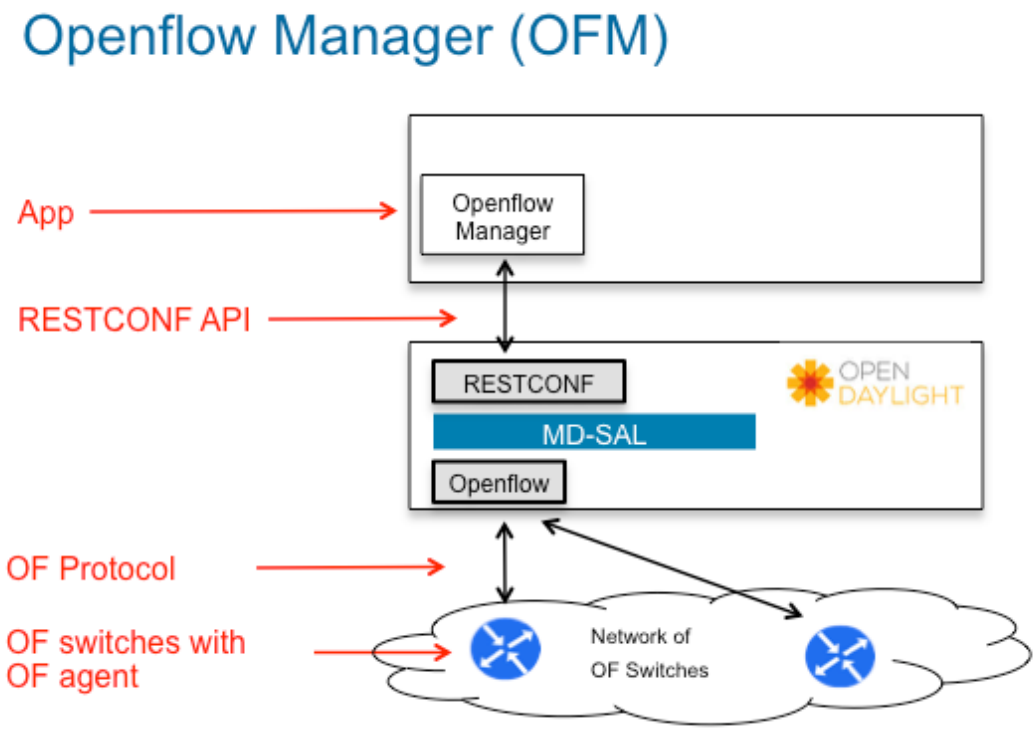

FIGURE 3.5 – Architecture dOFM [33]

Fonctionnalités d'OFM

Les fonctions de bases proposées par l'application affichée en haut de lécran de l'interface graphique (annexe 3) sont subdivisées en 4 onglets :

— **Basic view :** Vue globale de la topologie sous-jacente. OFM schématise la structure du réseau en affichant les switchs OpenFlow et les hôtes qui leurs sont connectés.
— **Flow management :** Ou gestion de flux. Permet de visualiser, ajouter, modifier ou supprimer les différents flux.
— **Statistics :** Fournit les statistiques liées aux flux et aux ports des switchs.
— **Hosts :** Résume les informations concernant les hôtes gérés par OFM.

3.5 Conclusion

Ce chapitre décrit les principaux outils utilisés dans la mise en œuvre du scénario que nous avons détaillé dans le chapitre suivant. Nous avons examiné et testé chaque élément de l'architecture SDN et montré différentes fonctionnalités, mais surtout, nous avons fait une étude bibliographique qui démontre leur fiabilité avant de l'utiliser plus tard.

Chapitre 4

Simulation d'un Réseau SDN

4.1 Introduction

Dans ce chapitre nous avons montré l'aspect de gestion centralisé puis la flexibilité d'un réseau SDN à l'égare de la technologie IoT, en simulant un réseau assez complexe dont le but est d'inter-connecter plusieurs zones composées de plusieurs noeuds IoT, via des commutateurs OpenFlow qui serviront de passerelles IoT entre les autres noeuds et zones, tout en gardant une gestion centralisé et dynamique, nous avons tout d'abord montré et expliqué, comment administrer un tel réseau. puis nous avons attaqué la flexibilité en implémentant d'une manière centralisé :

- Des politiques d'entrée et de sortie de flux pour le transfert de données entre les différentes régions.
- De la sécurité au réseau en imposant des politiques de par-feu (filtres) de différents niveaux.
- Un service WEB sur l'un des hôtes du réseau jouant le rôle de serveur HTTP.

4.2 Présentation du scénario

Le Software Defined Networking (SDN) est un concept de réseau qui permet la gestion et le contrôle centralisés à l'aide de logiciels. ce qui permet donc la programmation du comportement d'un réseau d'une manière dynamique et fluide.
Dans le cadre de ce mémoire nous avons simulé un contrôleur SDN capable de communiquer avec différentes applications réseau déployées sur des nœuds IoT
Pour cela nous avons fait appelle a l'emulateur (Mininet) et pour exploité les fonctionnalités du commutateur OpenFlow, nous avons donc utilisé (Open vSwitch), afin de mettre en œuvre le controller SDN,et utilisé OpenDaylight, et grâce a l'API OpenFlow Manager, nous effectuerons nos différentes configurations et politiques du réseau.

4.2.1 Topologie proposée

Pour notre cas nous avons déployé un réseau qui se compose de 5 Régions "A", "B", "C" et "D", chaque zone est conçu d'une manière hiérarchique (Accès, Distribution, Core), de plus la zone "**E**" qui est le coeur du réseau, c'est là que le contrôleur SDN est connecté avec le Switch OpenFlow 1 (SW.OF.1), mais il doit être invisible dans le réseau pour des raisons de sécurité, car c'est l'élément le plus important du réseau.
Nous supposons que le réseau considéré comprend des appareils IoT (appareils sans fil avec capteurs connectés, équipement de contrôle et de surveillance , etc.) qui collectent des données et communiquent entre eux.

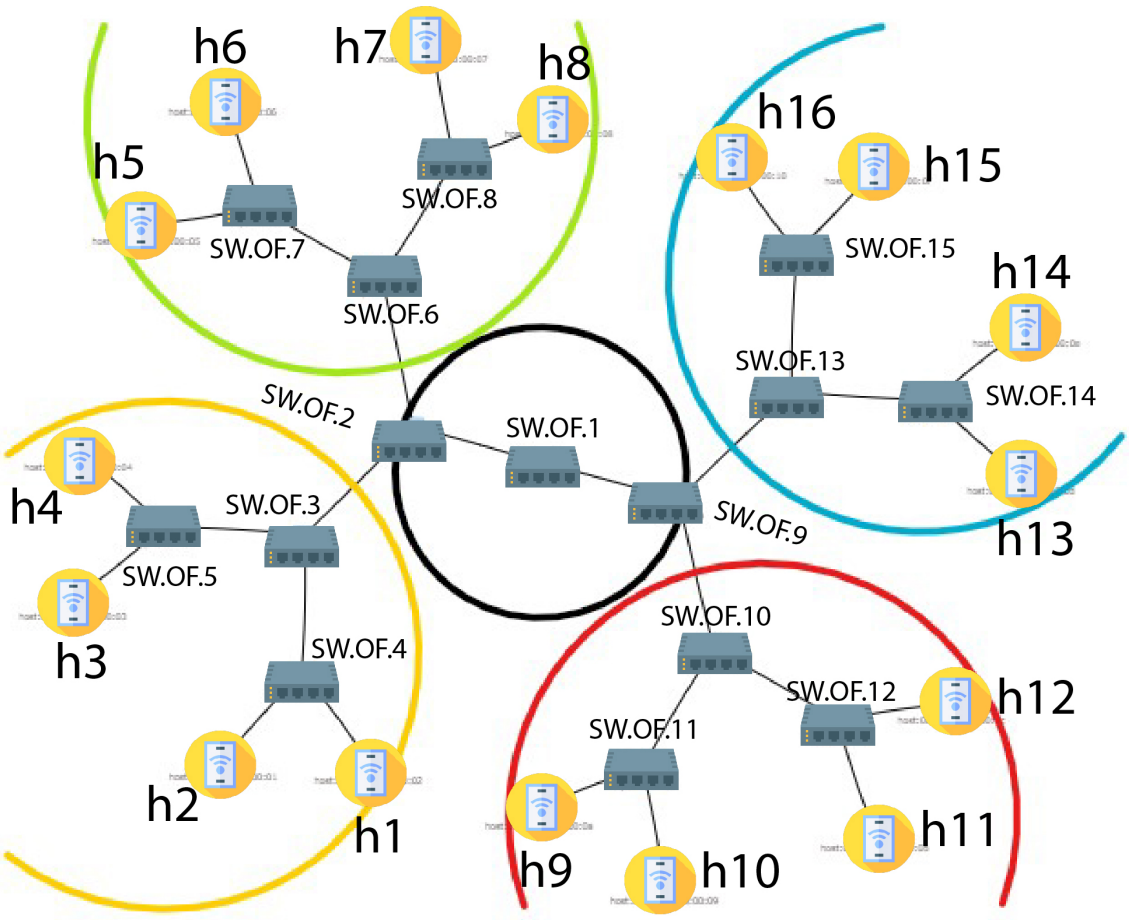

FIGURE 4.1 – Topologie du réseau

4.2.2 Plan d'adressage

Pour notre part, les adresses IP et les adresses MAC ont été distribuer d'une manière automatique par le serveur DHCP local de Mininet.
le tableau 4.1 montre le plans d'adressage du réseau :
(dans le champ d'adresses MAC seul les 2 dernier octets qui sont pris en considération).

Région	Hôte	Adresse MAC	Adresse IP
A	h1	:01	10.0.0.1/8
	h2	:02	10.0.0.2/8
	h3	:03	10.0.0.3/8
	h4	:04	10.0.0.4/8
B	h5	:05	10.0.0.5/8
	h6	:06	10.0..0.6/8
	h7	:07	10.0..0.7/8
	h8	:08	10.0..0.8/8
C	h9	:09	10.0.0.9/8
	h10	:0a	10.0.0.10/8
	h11	:0b	10.0.0.11/8
	h12	:0c	10.0.0.12/8
D	h13	:0d	10.0.0.13/8
	h14	:0e	10.0.0.14/8
	h15	:0f	10.0.0.15/8
	h16	:10	10.0.0.16/8

TABLE 4.1 – Plan d'adressage

4.3 Opérations d'administration

Le transfert de données

Concernant le SDN ,le routage et la commutation ne correspondent pas a la terminologie adéquate. La terminologie utilisée est simplement la transmission, en séparant la partie intelligente des équipements de réseau de leurs partie de traitement, ce qui fait que les décisions de routage ou de commutation ne sont plus prises par léquipement lui-même, du coup les protocoles de routage et de commutation ne sont plus nécessaires. Cependant, ils peuvent toujours être mis en œuvre en permettant le déploiement de réseaux hybrides avec des Commutateurs OpenFlow, des routeurs et des commutateurs traditionnels.
Étant donné que les commutateur OpenFlow ont une structure multi-couches, et en les combinant avec un contrôleur, on aura la possibilité de contrôle des flux de donnés, ce qui nous permet de toucher à la gestion du comportement du réseau.
Nous avons aussi mis en oeuvre une certaine politique de sécurité, en utilisant des règles de par-feu de différents niveaux.

La sécurité

A fin de générer des politiques de sécurité, nous avons exploité les tables de flux.
En utilisant laction *drop* d'OpenFlow à partir du contrôler, nous avons bloqué un trafic en fonction :

- De son type de protocole de transport voire le port découte de source et de destination TCP/UDP(Niveau 4).
- Des adresses IP de source et de destination (Niveau 3).
- Des adresses MAC de source et de destination (Niveau 2).
- Et même des Tags de VLANs ou MPLS.

Cependant, nous avons besoin d'une application d'inspection et d'identification des paquets, pour tirer

pleinement parti du potentiel de sécurité du SDN, nous avons donc opté pour Wireshark, lancé à partir de Mininet.

4.4 Simulation du réseau

Pour simuler notre réseau nous avons installé 2 machines virtuelles sur la station de virtualisation (Virtual Box 6.1).
Nos VMs (Virtual Machine) serrons dotés de deux cartes réseau virtuel.

- Une est configuré en (Bridge) pour télécharcher les packages nécessaires pour le fonctionnement d'ODL (Open Day Light) et Mininet.
- Et l'autre est configuré en mode (Réseau privé de hôtes), permettant la communication entre les machines virtuelles uniquement.

Ceci nous permettra douvrir des terminaux SSH pour une meilleure manipulation des VMs avec surtout la possibilité dexécuter toutes sortes de programmes linux sur le serveur d'affichage X11 et de pouvoir lancer les interfaces graphiques d'ODL et OFM sur un navigateur local.
La configuration des machines est la suivante :

4.4.1 Mininet

Toutes les versions de l'émulateur sont disponibles sur (Github, 2017), sous formes de VM préparées, il suffit d'importer le fichier .OVF sur l'hyperviseur. La version installée dans notre cas est la 2.2.2, sur un OS ubuntu serveur 14.04.4 amd La configuration de la machine est illustrée dans la figure 4.2 :

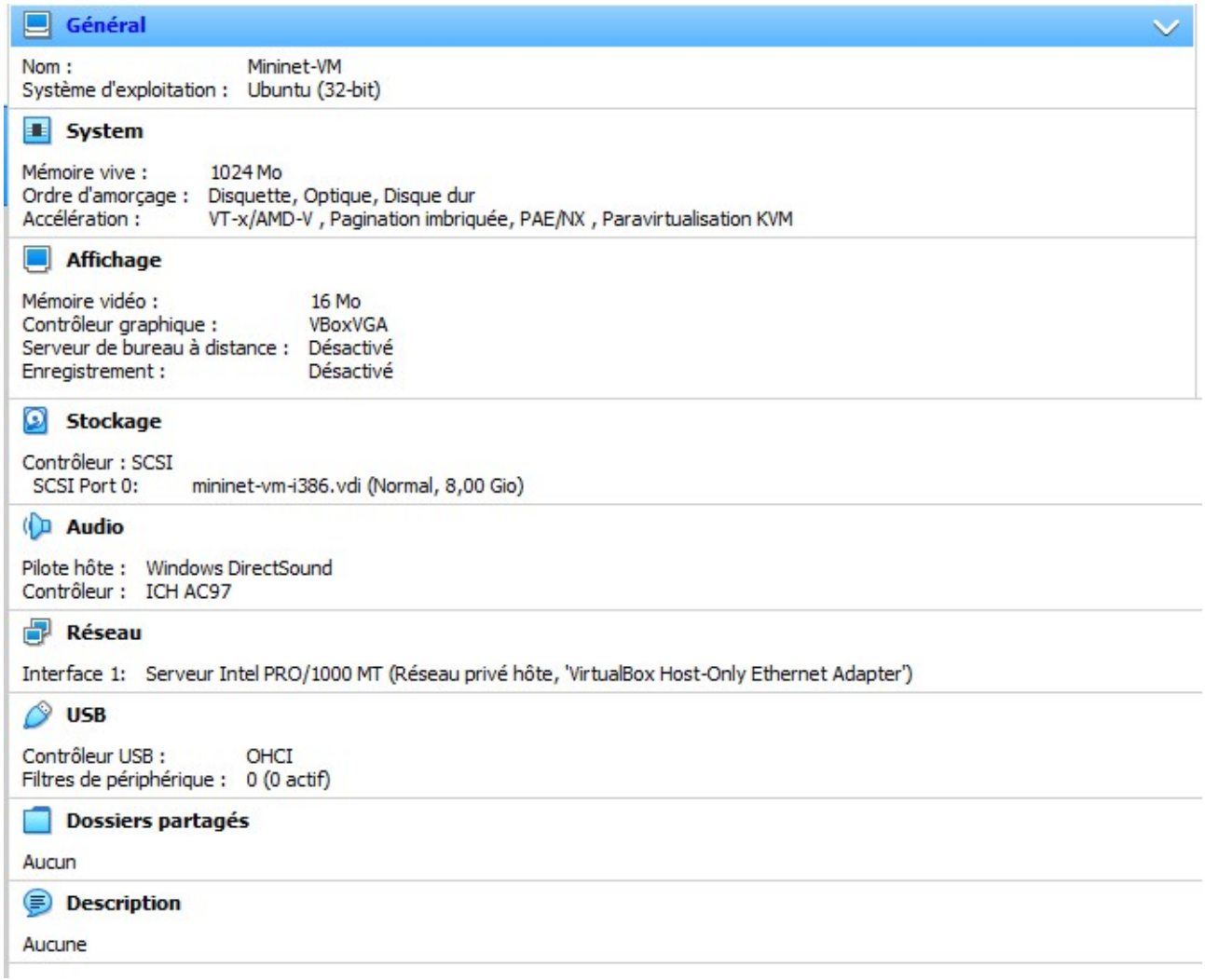

FIGURE 4.2 – Configuration de la machine Mininet

4.4.2 Opendaylight

Nous avons opté pour la version Beryllium du contrôleur, disponible dans le dépôt dOpendaylight. Le contrôleur a été installé sur une plateforme ubuntu 20.4 LTS amd64.
La configuration de la machine est illustrée dans la figure 4.3 :

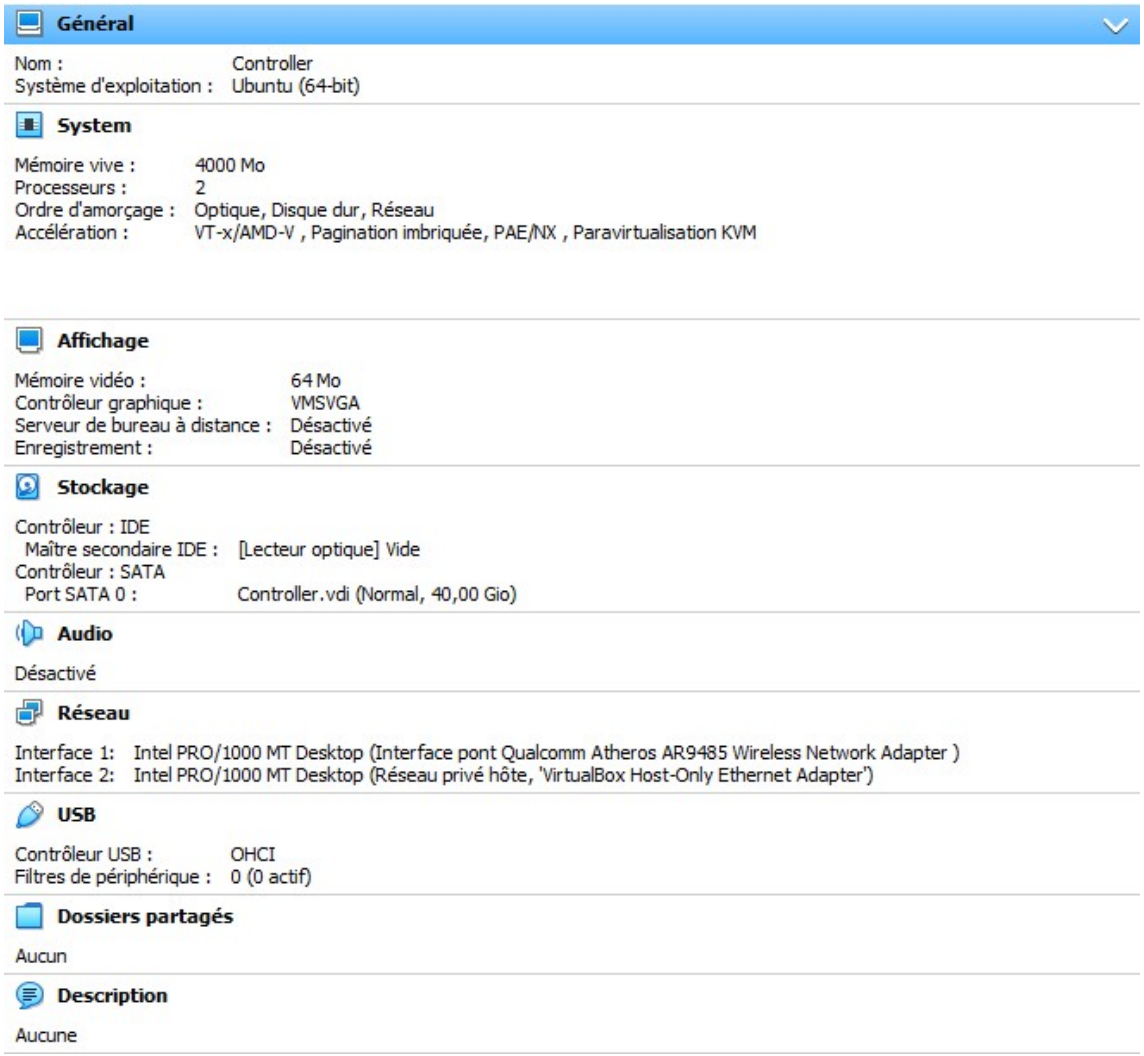

FIGURE 4.3 – Configuration de la machine ODL

4.4.3 Topologie du Réseau

Notre topologie serra crée à partir des topologies pré chargé de Mininet. On peut bien évidement créer une topologie customisé par le biais d'un script Python sauvegardé dans le répertorie **mininet/custom**, mais nous avons choisi ces topologie pré charger pour éviter de réécrire le tout script python, pour nous permettre d'étendre notre réseau facilement .
La commande pour créer le réseau personnalisé cité ci-dessus dans (l'annexe 3)

Le contrôleur détecte les Switchs Openflow, et après un ping global détecte les hôtes et parvient à afficher la structure du réseau. Sur l'application OpenFlow Manager déployée sur l'interface nord du contrôleur écoutant sur le port 9000. La topologie en est immédiatement extraite et affichée dans l'interface graphique comme le montre la figure 4.4 :

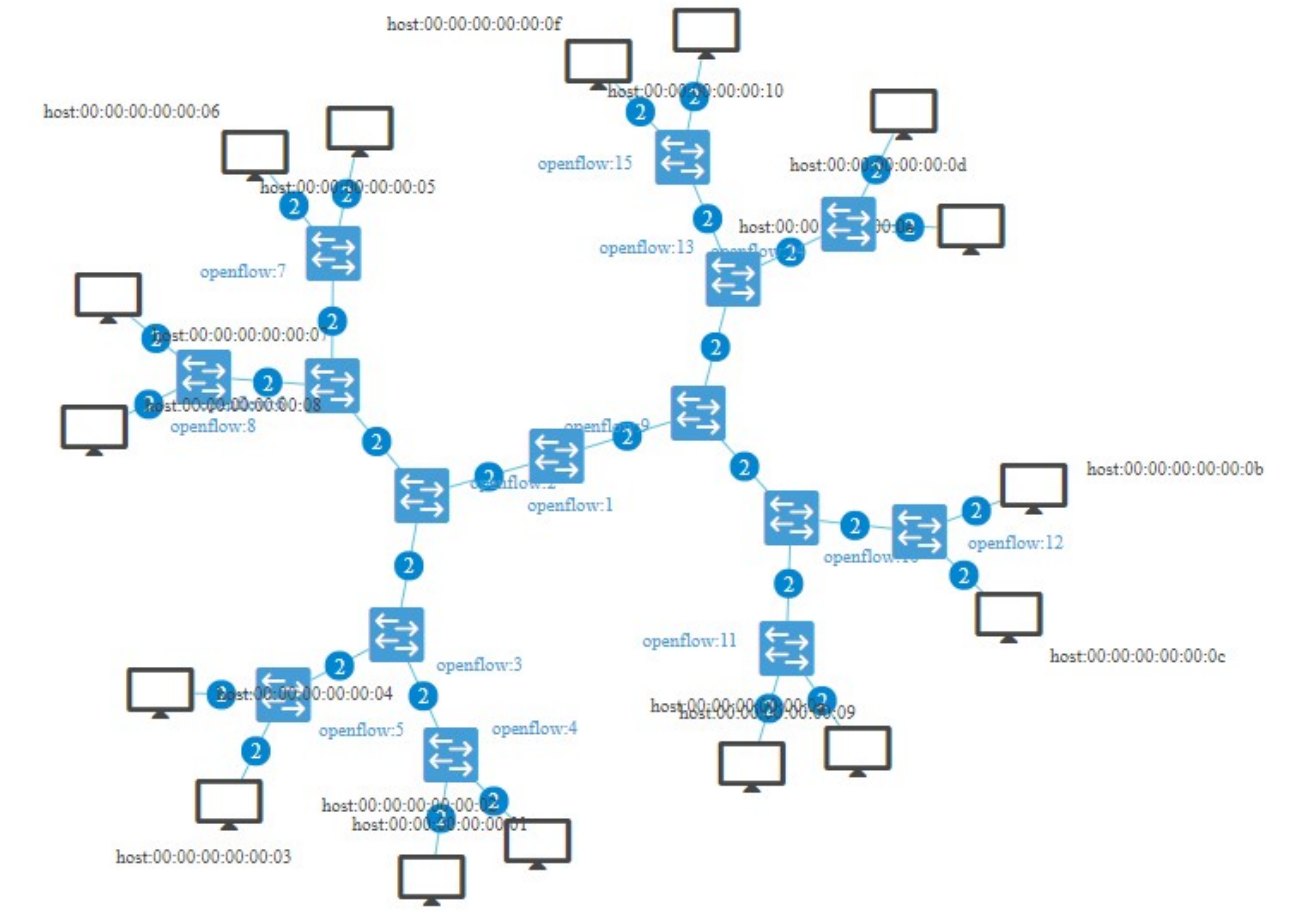

FIGURE 4.4 – Topologie du scénario détectée et affichée sur OFM

Le protocol OpenFlow

Lors du lancement de la topologie nous avons remarqué sur Wireshark que des messages OpenFlow transitent entre le contrôleur et Mininet. Pour la gestion de la topologie, ces paquets sont transmis périodiquement accompagnés avec des paquets LLDP (Link Layer Discovery Protocol). Ce dernier est un protocole qui sert à la découverte des topologies réseau, mais aussi à apporter des mécanismes d'échanges d'informations entre équipements réseaux, et utilisateurs finaux, comme l'atteste la figure 4.5 :

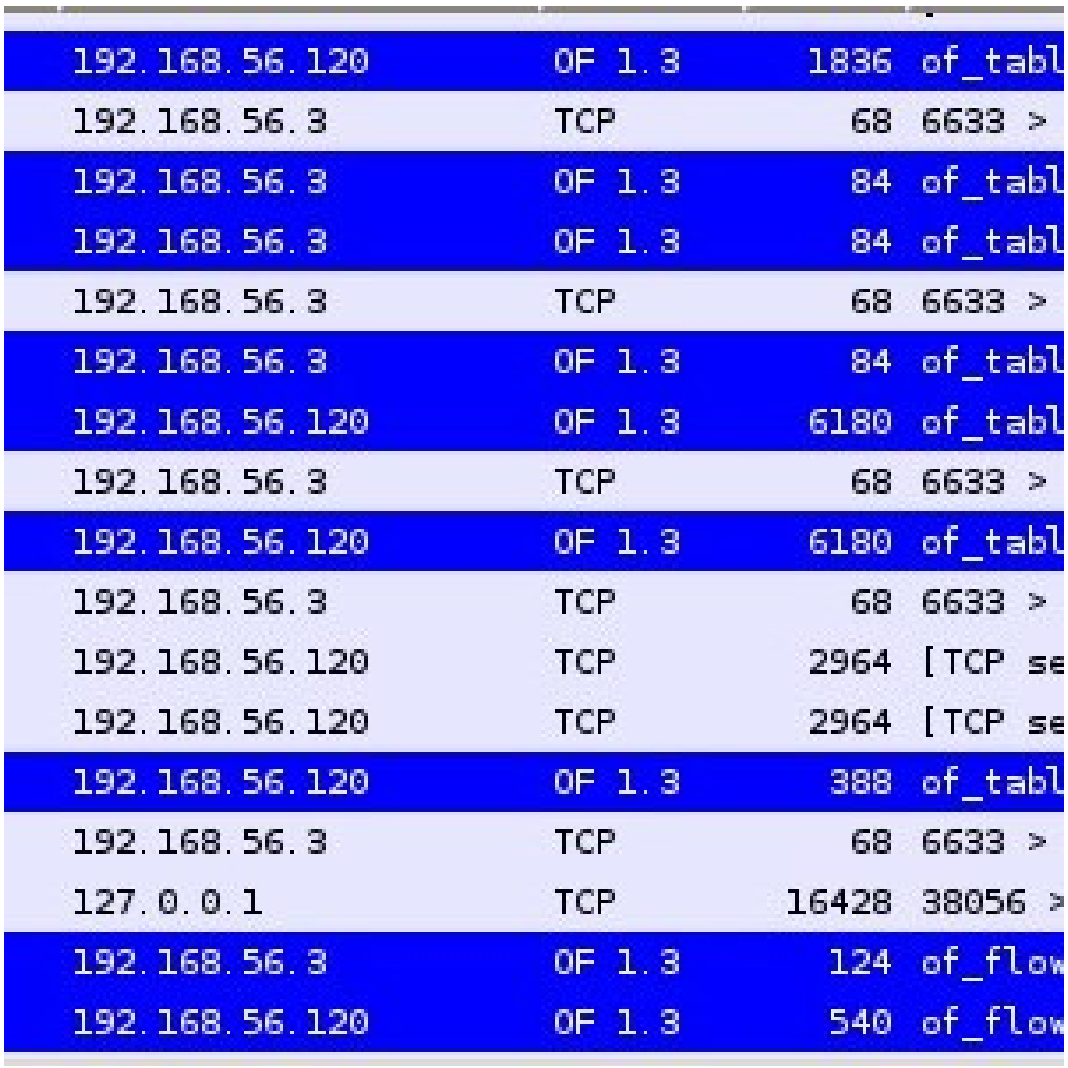

FIGURE 4.5 – Openflow et LLDP

Plan de comportement du réseau

Ici nous avons résumé le comportement du réseau en précisant, les trafics bloqués et l'attribution des VLANs

Région A

- Tout le réseau de la région serra attribué au VLAN 3.
- Le hôte h1 serra un serveur WEB.
- Sauf h9 qui sera joignable dans la région "C"

Région B

- Tout le réseau de la région serra attribué au VLAN 6.
- Les hôtes h7 et h8 ne communiquent que entre eux.
- les hôtes h5 et h6 communiquent avec toutes les région mais pas entre eux.

Région C

- Tout le réseau de la région serra attribué au VLAN 10.
- Les hôtes h11 et h12 ne communiquent que entre eux.
- Le hôte h10 ne communique pas avec la région A.

Région D

- Tout le réseau de la région serra attribué au VLAN 13.
- Cette région ne subira aucune restriction, elle peut donc communiquer avec les autres région.

Tables de flux introduites

Nous avons mis quelques configuration des tables de flux que nous avons crée et introduites dans les différents Switchs de la topologie via le contrôleur SDN, décrivant leurs comportement en fonction des paquets reçu.

Openflow 2			
ID	**Iput port**	**Action**	**Priorité**
VLN6	1	Set VLAN ID=10 -Output port : 2 - Output port : 3	20
VLN3	2	Set VLAN ID=10 -Output port : 1 - Output port : 3	20

FIGURE 4.6 – Table de flux SW.OF.2

Openflow 9			
ID	**Input port**	**Action**	**Priorité**
VLN13	2	Set VLAN ID=13 -Output port : 1 - Output port : 3	20
VLN10	1	Set VLAN ID=10 -Output port : 2 - Output port : 3	

FIGURE 4.7 – Table de flux SW.OF.9

Openflow 8						
ID	**Eth Type**	**Src MAC**	**Dst MAC**	**Src IP**	**Action**	**Priorité**
708		: 07	: 08		Output port : 2	50
807		: 08	: 07		Output port : 1	50
780N	2048			10.0.0.0/8	Drop	20

FIGURE 4.8 – Table de flux SW.OF.8

Openflow 7						
ID	**Eth Type**	**Src MAC**	**Dst MAC**	**Src IP**	**Action**	**Priorité**
506	2048			10.0.0.6/8	Drop	20
605	2048			10.0.0.5/8	Drop	20

FIGURE 4.9 – Table de flux SW.OF.7

Openflow 12							
ID	Eth Type	Src MAC	Dst MAC	Src IP	Dst IP	Action	Priorité
9012x	2048			10.0.0.0/8		Drop	20

FIGURE 4.10 – Table de flux SW.OF.6

Openflow 5							
ID	Eth Type	Src MAC	Dst MAC	Vlan ID	Dst IP	Action	Priorité
1043	2048			6		Drop	20

FIGURE 4.11 – Table de flux SW.OF.5

Expliquer le déploiement de tous les flux dans notre réseau est fastidieux et encore plus itératif. Dans (l'annexe 3) vous trouverez quelques flux introduits dans le cadre de la configuration du réseau.

4.5 Tests de fonctionnement

4.5.1 Test d'accessibilité avant l'implantation de la politique

ceci représente le premier test lors-ce que nous avons lancé notre topologie sur Mininet, et on remarque que tout les hôtes communiquent entre eux.

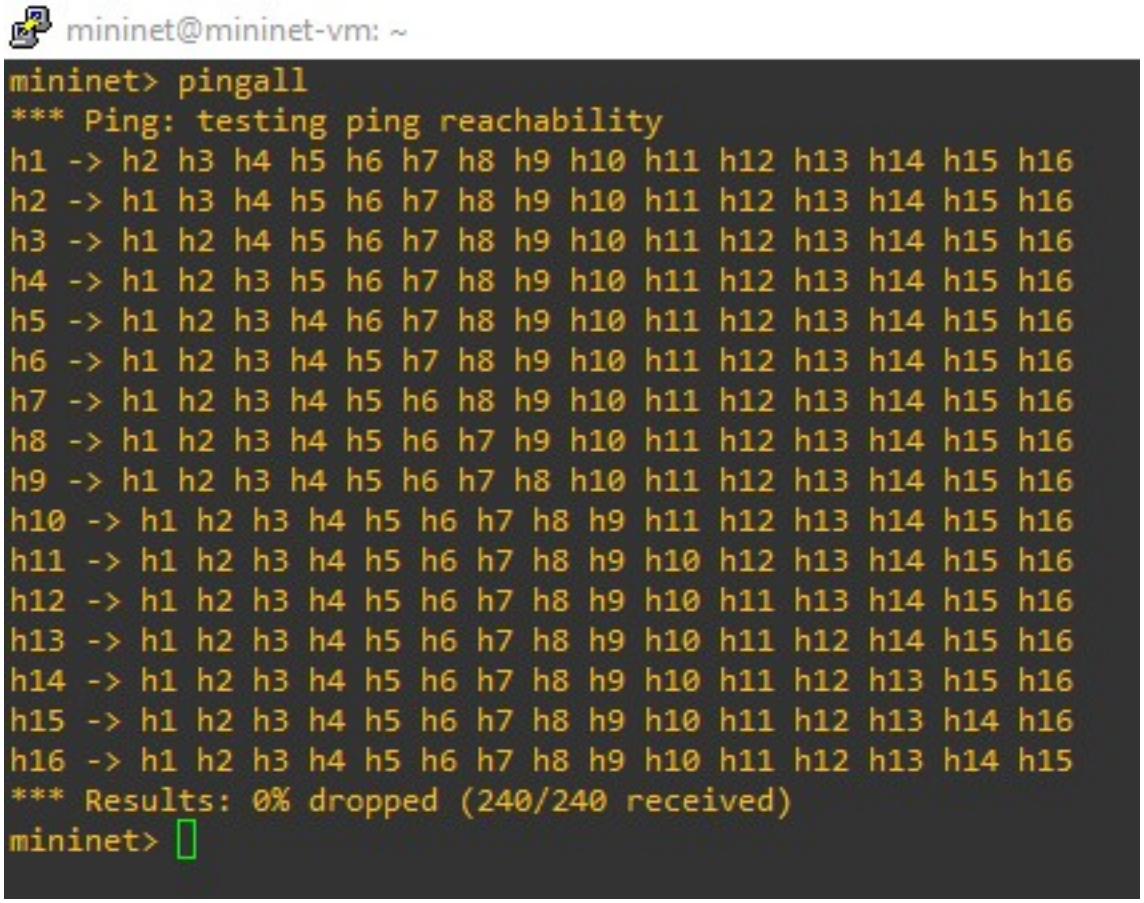

FIGURE 4.12 – Test Avant l'implantation de la politique

4.5.2 Test d'accessibilité après l'implantation de la politique de sécurité

Comme l'atteste la figure 4.13, le réseau se comporte comme prévu en termes d'accessibilité, ce qui confirme la fiabilité des restrictions appliquées.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 X X h9 X X X h13 h14 h15 h16
h2 -> h1 h3 h4 h5 h6 X X h9 X X X h13 h14 h15 h16
h3 -> h1 h2 h4 h5 h6 X X h9 X X X h13 h14 h15 h16
h4 -> h1 h2 h3 h5 h6 X X h9 X X X h13 h14 h15 h16
h5 -> h1 h2 h3 h4 X X X h9 h10 X X h13 h14 h15 h16
h6 -> h1 h2 h3 h4 X X X h9 h10 X X h13 h14 h15 h16
h7 -> X X X X X X h8 X X X X X X X
h8 -> X X X X X X h7 X X X X X X X
h9 -> h1 h2 h3 h4 h5 h6 X X h10 X X h13 h14 h15 h16
h10 -> X X X X h5 h6 X X h9 X X h13 h14 h15 h16
h11 -> X X X X X X X X X h12 X X X X
h12 -> X X X X X X X X X h11 X X X X
h13 -> h1 h2 h3 h4 h5 h6 X X h9 h10 X X h14 h15 h16
h14 -> h1 h2 h3 h4 h5 h6 X X h9 h10 X X h13 h15 h16
h15 -> h1 h2 h3 h4 h5 h6 X X h9 h10 X X h13 h14 h16
h16 -> h1 h2 h3 h4 h5 h6 X X h9 h10 X X h13 h14 h15
*** Results: 47% dropped (126/240 received)
mininet>
```

FIGURE 4.13 – Test Après l'implantation de la politique

4.5.3 Test d'accessibilité du serveur HTTP

Nous avons effectué une requête HTTP via l'un des hôtes ayant le privilège de communiquer avec "h1", pour notre part nous avons choisi "h16" comme exemple. La commande utilisé effectuer la requête "WEB" est **wget**, et pour activer le service "WEB" sur le serveur nous avons utilisé la commande **python -m SimpleHTTPServer 80** sur le serveur "h1"

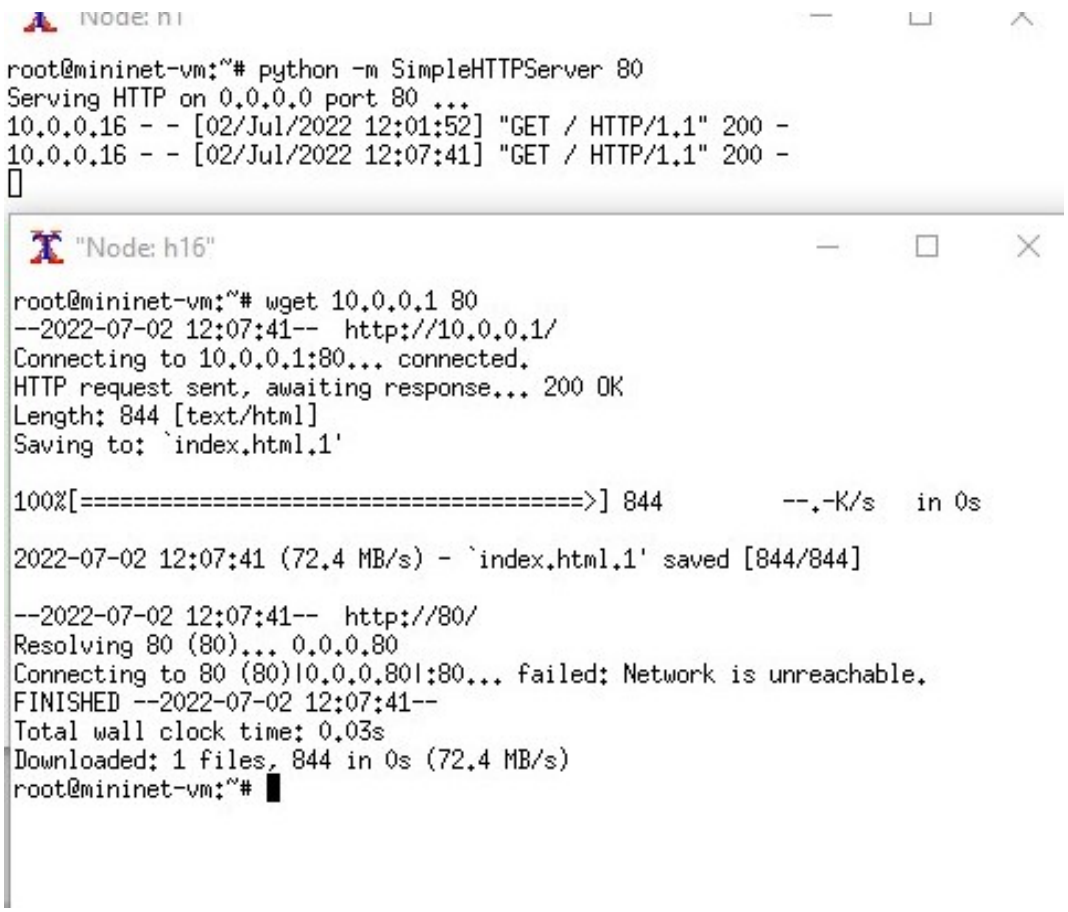

FIGURE 4.14 – Introduction des commandes pour une requête WEB

la figure 4.15 montre les paquet HTTP envoyés et reçus par "SW.OF.4" sur le quel est connecter le serveur "h1".

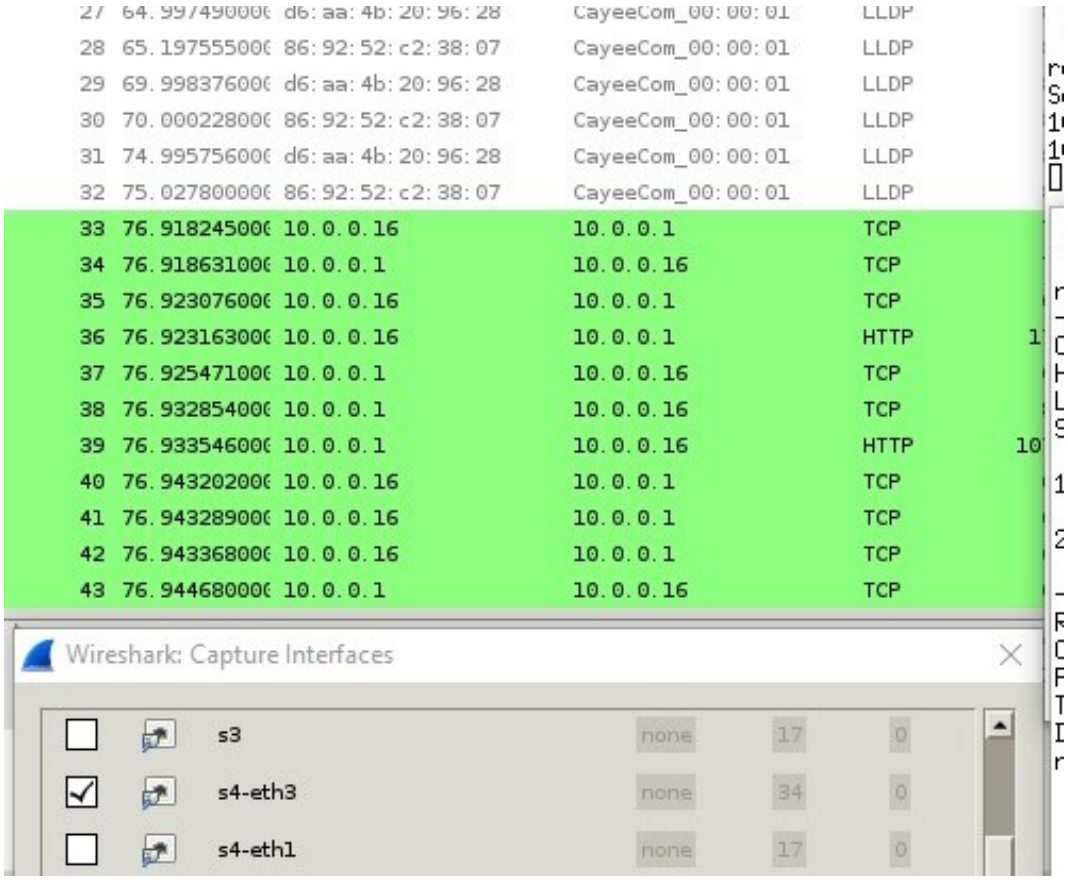

FIGURE 4.15 – Teste du service WEB via h16

4.5.4 Discussion

Avec la gestion centralisé qu'offre le SDN nous avons pu mettre en oeuvre différentes politiques de sécurité, en bloquant le trafic entre certains hôtes en utilisant plusieurs types de filtres (MAC,IP,TCP), et nous avons introduit un serveur HTTP et vérifier son bon fonctionnement en générant une requête WEB et l'observer sur wireshark.
Ce scénario a pour but de montrer l'aspect de gestion centralisé qui permet la flexibilité et la facilité d'administrer un réseau IoT avec une architecture SDN.

4.6 Conclusion

Dans ce chapitre nous avons décrit les opérations d'administration les plus fréquentes. En plus de la simplicité de l'implémentation et le gain de temps considérable en ce qui concerne la programmation du comportement des réseaux ce qui nous rend beaucoup moins limités en termes de réalisation de topologies complexes comme dans le cas de l'IoT. Les résultats ont démontrés la fiabilité de l'architecture en termes du respect des politiques prédéfinis. Un test d'accessibilité a montré le comportement du réseau en termes de restrictions, celui-ci a confirmé la conformité des communications envers ce qui a été pré-établi.

Conclusion

Dans ce travail nous avons passé en revue le concept des réseaux définit par logiciel (SDN), pour ensuite limplémenter et en déduire son apport sur le comportement des réseau tout en testant sa fiabilité, mais aussi nous avons introduit l'un des principaux problèmes de l'internet des objets (IoT), et les causes principales qui nous pousse à adopter de nouvelles approches telles que les réseaux SDN qui ont pour but de réduire la complexité de gestion et de contrôle d'une ou de plusieurs infrastructures réseau. Nous avons rencontré au court de notre étude bibliographique dinnombrables définitions sur ce quest ou ce que devrait être le SDN. Par contre lobjectif de celui-ci reste clair et unifié, apporter une plateforme de contrôle et de programmation pour la communication et le déploiement de services automatisés, simplifier le matériel réseau et accélérer le développement en adoptant le modèle Open Source de linux. Le seul standard actuellement reste OpenFlow, qui a ouvert les discussions sur le SDN, avec lidée de découplage du plan de contrôle et du plan de données. Malgré son déploiement massif par les plus grandes sociétés (Google, Microsoft, Facebook, Amazon, etc.), SDN reste tout de même une technologie immature dans sa façon daborder les applications. Nous avons choisis limplémentation du modèle Openflow, en lintégrant dans un environnement virtuel. L'un des points forts de celui-ci est la facilité de son déploiement grâce notamment à la probabilité du switch OVS, qui peut fonctionner en virtuel ou être installé sur du très simple matériel. L'étude s'est ensuite focalisée sur le contrôleur OpenDaylight, qui est le fruit d'un projet géré par la communauté réseau et financé par les plus grands industriels du domaine. Ce contrôleur qui supporte la version 1.3 d'OpenFlow, offre une plateforme dabstraction et de programmablilitée très riche en diversité de fonctionnalités, elle exploite le modèle correspondance/Actions du protocole OpenFlow et offre entre autres une interface fluide aux administrateurs pour programmer les réseaux à travers l'API RESTCONF. Nous avons exploités les différents éléments cités antérieurement pour réaliser un réseau SDIoT (Software Defined Internet of Things) programmable grâce à linterface OpenFlow Manager. A travers ce travail, nous avons utilisés de nombreux outils de test qui permettent de confirmer la fiabilité de cette architecture. A partir des résultats de tests, nous pouvons dire que le SDN remet en question le modèle actuel des réseaux traditionnelles, et offre de nombreuses solutions quant à la programmation du comportement des flux de données. Il reste cependant du chemin à faire en termes de standardisation, de quoi préparer le terrain pour une transition globale inévitable vers ce type de réseaux. Le point fort de celui-ci étant la couche application, et sa perméabilité au déploiement instantané de services.

Annexes

Annexe 1

Installation de opendaylight (ODL)

Prérequis

1-Préparer le système d'exploitation

Mettez à jour votre système d'exploitation, vos applications et vos outils de sécurité via le gestionnaire de packages apt.
Exécutez une mise à jour apt-get, qui actualise la liste des packages disponibles.

```
$ sudo apt-get -y update
```

Maintenant, mettez à niveau les packages via l'option de mise à niveau.

```
$ sudo apt-get -y upgrade
```

Installez unzip, pour décompresser l'archive OpenDaylight.

```
$ sudo apt-get -y install unzip
```

Installer le JRE Java

Les architectes OpenDaylight ont conçu OpenDaylight pour l'écosystème Java. OpenDaylight nécessite un environnement d'exécution Java (JRE) pour fonctionner. OpenDaylight peut tirer parti d'un JRE autonome sur le JRE fourni dans un kit de développement logiciel Java.
La commande suivante installe le JRE JAVA 8.

```
$ sudo apt-get -y install openjdk-8-jre
```

Utilisez la commande update-alternatives pour définir Java par défaut sur JAVA 8. update-alternatives présente une liste des versions Java installées et vous permet de sélectionner la version par défaut souhaitée. Si update-alternatives fournit une liste de versions, sélectionnez JAVA 8 dans la liste.

```
$ sudo update-alternatives –config java
```

"update-alternatives" affichera le chemin complet vers votre exécutable JAVA. Copiez ce chemin, vous en aurez besoin pour définir la variable d'environnement "JAVA_HOME" à l'étape suivante.

Définir JAVA_HOME

Récupérez le chemin complet de votre exécutable JAVA. Si vous avez perdu la trace, vous pouvez exécuter la commande suivante :

```
$ ls -l /etc/alternatives/java
```

OpenDaylight veut que la variable d'environnement "JAVA_HOME" reflète l'emplacement de l'ensemble d'outils JAVA, et pas seulement l'exécutable JAVA. Pour cette raison, supprimez bin/java du chemin. Cela définit "JAVA_HOME" sur l'emplacement du JRE.

Sur Ubuntu LTS 20.04, le JRE JAVA 8 réside dans "/usr/lib/jvm/java-8-openjdk-amd64/jre"

Pour définir (et conserver) la valeur de "JAVA_HOME", modifiez votre fichier de ressources "BASH"

$ éxport JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' » /.bashrc

Ubuntu lit votre fichier de ressources BASH chaque fois que vous vous connectez au shell. Pour définir "JAVA_HOME" pour la première fois, vous pouvez soit vous déconnecter puis vous reconnecter à votre shell, soit simplement sourcer le fichier de ressources. Pour sourcer le fichier, exécutez la commande suivante :

$ source /.bashrc

Une fois que vous avez source le fichier, assurez-vous que "$JAVA_HOME se termine par /jre".

$ echo $JAVA_HOME

Téléchagez la version OpenDaylight souhaitée

Collez le lien (https ://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.4.4-Beryllium-SR4/distribution-karaf-0.4.4-Beryllium-SR4.zip) dans une commande CURL

$ curl -XGET -O (lien)

Décompressez et installez OpenDaylight

Assurez-vous d'avoir téléchargé le fichier zip.

$ ls

vous devrez retrouver le fichier "distribution-karaf-0.4.4-Beryllium-SR4.zip" dans le répertoire.
décompresser le fichier avec

$ unzip distribution-karaf-0.4.4-Beryllium-SR4.zip

puis retrouvez le dossier "distribution-karaf-0.4.4-Beryllium-SR4"

Démarrer OpenDaylight

Maintenat vous pouvez lancer Opendaylight avec la commande

$ ./distribution-karaf-0.4.4-Beryllium-SR4/bin/karaf

on retrouve donc l'interface suivante :

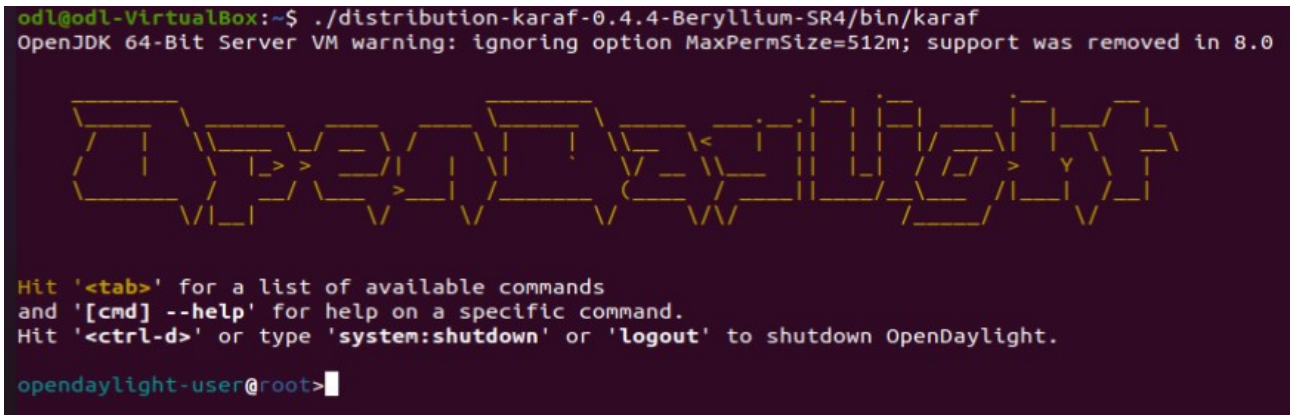

FIGURE 4.16 – L'ancement de OpenDaylight

A présent vous devez installer les fonctionnalités suivantes :

- Opendaylight-user@root>feature :install odl-restconf-all
- Opendaylight-user@root>feature :install odl-openflowplugin-all
- Opendaylight-user@root>feature :install odl-l2switch-all
- Opendaylight-user@root>feature :install odl-mdsal-all
- Opendaylight-user@root>feature :install odl-yangtools-common

Interface graphique d'opendaylight

Pour accéder au GUI (Graphical User Interface) (Interface D'utilisateur Graphique), vous pouvez utiliser un navigateur WEB, puis tapez sue le champ URL : <IP de la VM> :8181/index.html.

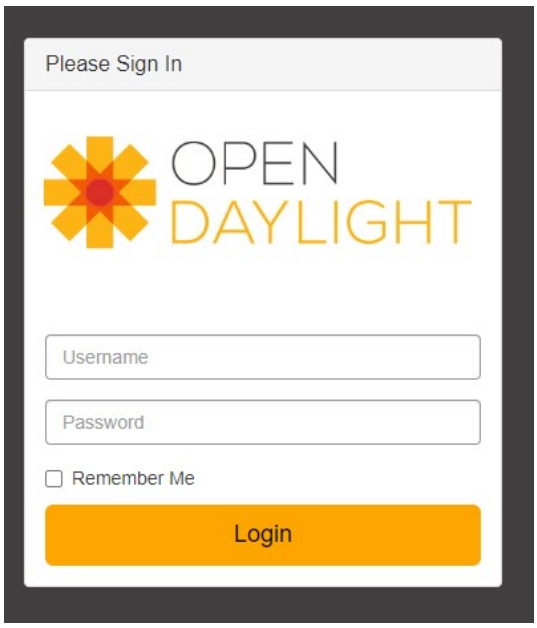

FIGURE 4.17 – Interface de connexion à de OpenDayligh

Pour se connecter il suffit d'introduire :

— username=admin
— password=admin

Annexe 2

Installation dOpenFlow Manager (OFM)

Installation de Nodejs

Un environnement dexécution Javascript côté serveur. Ceci permettra de faire fonctionner grunt.

```
# curl sl https ://deb.nodesource.com/setup_4.x | sudo -E bash
#apt-get install nodejs
```

Téléchargement dOFM

Clonage du répertoire github de lapplication.

```
# git clone https ://github.com/CiscoDevNet/opendaylight-openflow-App.git
```

Configuration de ladresse IP du serveur web

modifiez le fichier (env.module.js),pour ce vous devez rendre vers le repertoire suivant

```
$ nano /opendaylightopenflowApp/ofm/src/common/config/env.module.js
```

puis mettez l'adresse du controlleur opendaylight au champ indiqué dans la figure suivante :

```
 GNU nano 4.8                                                    env.module.js
define(['angularAMD'], function(ng) {
    'use strict';

    var config = angular.module('config', [])
        .constant('ENV', {
          baseURL: "http://192.168.56.120:",
          adSalPort: "8181",
          mdSalPort : "8181",
          ofmPort : "8181",
          configEnv : "ENV_DEV",
          odlUserName: 'admin',
          odlUserPassword: 'admin',
          getBaseURL : function(salType){
              if(salType!==undefined){
                  var urlPrefix = "";
                  if(this.configEnv==="ENV_DEV"){
                      urlPrefix = this.baseURL;
                  }else{
                      urlPrefix = window.location.protocol+"//"+window.location.hostname+":";
                  }

                  if(salType==="AD_SAL"){
                      return urlPrefix + this.adSalPort;
                  }else if(salType==="MD_SAL"){
                      return  urlPrefix + this.mdSalPort;
                  }else if(salType==="CONTROLLER"){
                      return  urlPrefix + this.ofmPort;
                  }
              }
              //default behavior
              return "";
```

Figure 4.18 – Le fichier env.module.js pour introduire l'adresse IP du contrôleur

Installation de grunt

```
# npm install g grunt-cli
```

Exécution du serveur web dOFM

- La commande `$ sudo opendaylightopenflowApp/grunt` démarrera une instance du serveur web, et affichera ceci sur la ligne de commande

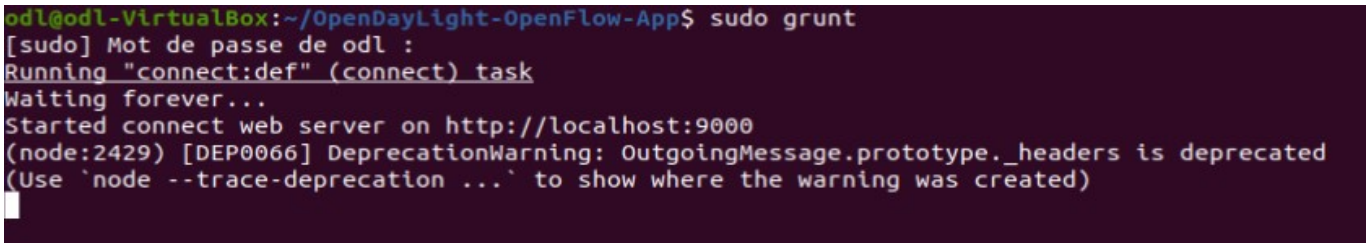

FIGURE 4.19 – L'ancement de OpenFlow Manager

Interface graphique d'openflow manager

À partir d'un navigateur sur votre machine local tapez (<IP de la VM> :9000)

Annexe 3

Création de la topologie sur mininet

```
sudo mn –topo tree,4 –controller remote, ip=192.168.56.120,port=6633 –mac  –switch ovsk,protocols=OpenFlow13
```

Celle-ci générera les différents éléments de la couche physique qui sera connectée au contrôleur OpenDaylight à distance.

```
mininet@mininet-vm:~$ sudo mn --topo tree,4 --mac --controller remote,ip=192.168
.56.120,port=6633 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15
*** Adding links:
(s1, s2) (s1, s9) (s2, s3) (s2, s6) (s3, s4) (s3, s5) (s4, h1) (s4, h2) (s5, h3)
 (s5, h4) (s6, s7) (s6, s8) (s7, h5) (s7, h6) (s8, h7) (s8, h8) (s9, s10) (s9, s
13) (s10, s11) (s10, s12) (s11, h9) (s11, h10) (s12, h11) (s12, h12) (s13, s14)
(s13, s15) (s14, h13) (s14, h14) (s15, h15) (s15, h16)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16
*** Starting controller
c0
*** Starting 15 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 ...
*** Starting CLI:
mininet>
```

FIGURE 4.20 – Création des éléments du réseau dans Mininet

Interface d'OpenFlow Manager

Une fois lapplication installée et configurée, lurl qui nous renvoie vers linterface graphique est le suivant : **<ADRESSE IP du contrôleur> :9000**

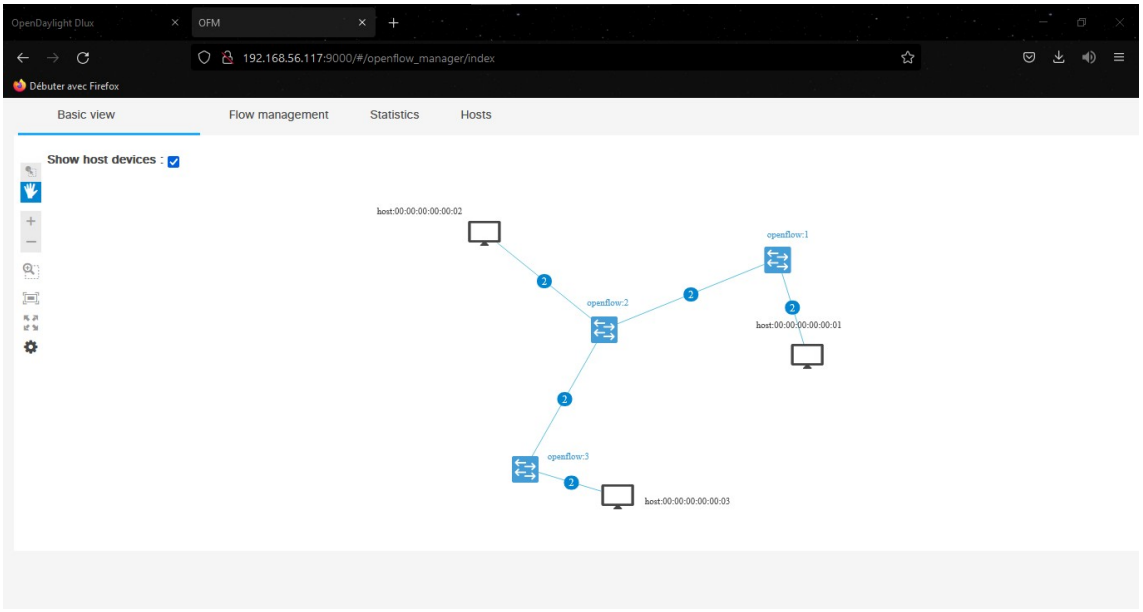

FIGURE 4.21 – Interface utilisateur dOFM

En ce qui concerne la programmation de flux, OFM supporte toutes les actions et identifications de paquets disponibles sur OpenFlow.
Pour créer un flux , il suffit daller sur longlet Flow Management, et de cliquer sur le bouton **show window for managing flows'** comme suit :

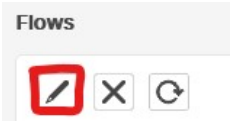

FIGURE 4.22 – Accès à la gestion de flux

Ensuite il ne reste plus quà remplir les cases sélectionnées pour générer lentrée de flux sur le switch choisi, et à envoyer la requête en cliquant sur "send request".

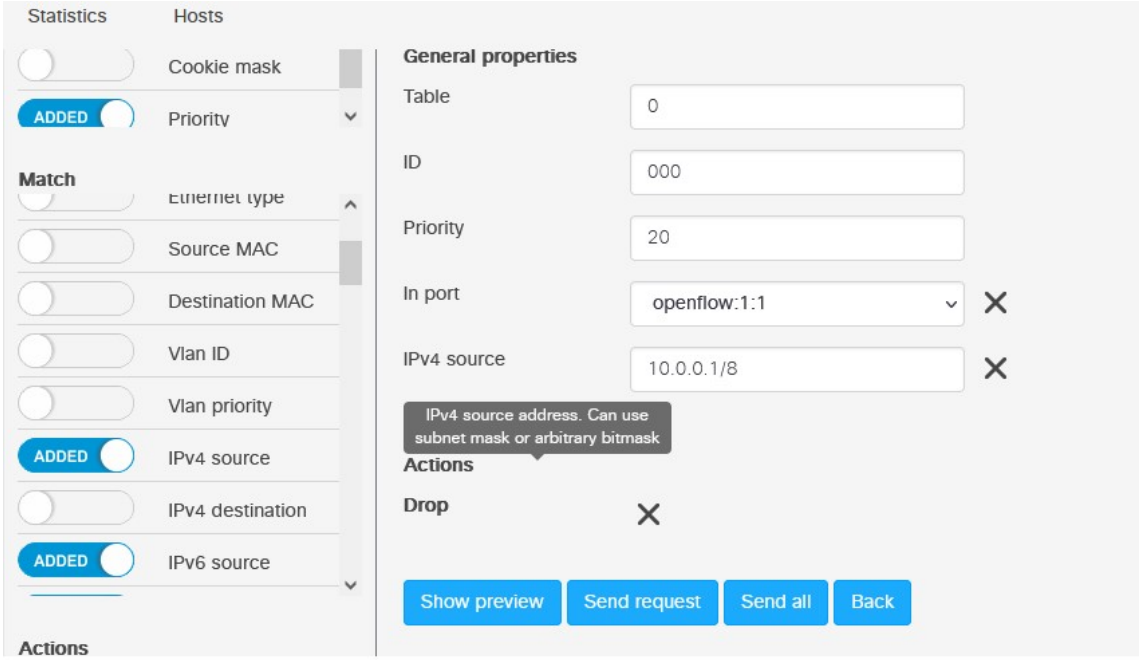

FIGURE 4.23 – Programmation de flux sur OFM

Voici Quelques flux introduit

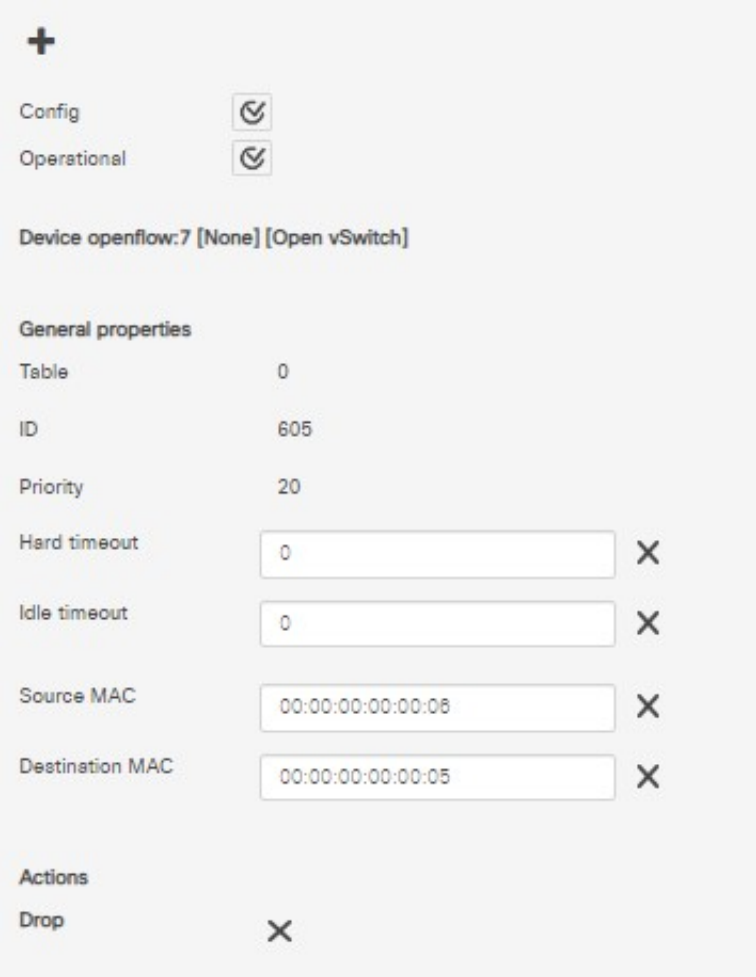

FIGURE 4.24 – Installation d'un filtre de niveau 2 sur s7

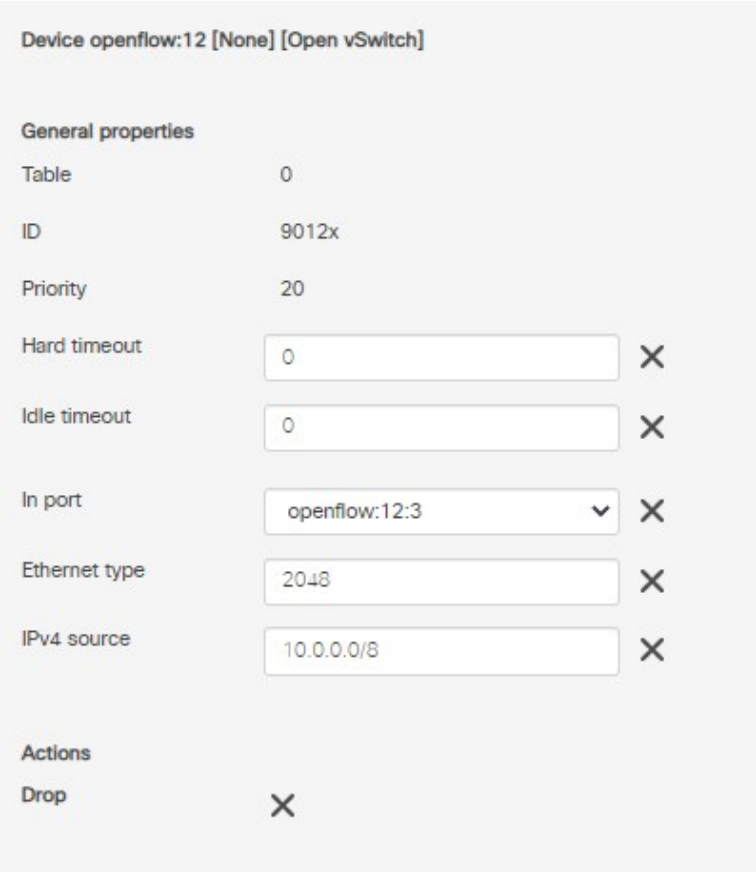

FIGURE 4.25 – Installation d'un filtre de niveau 3 sur s12

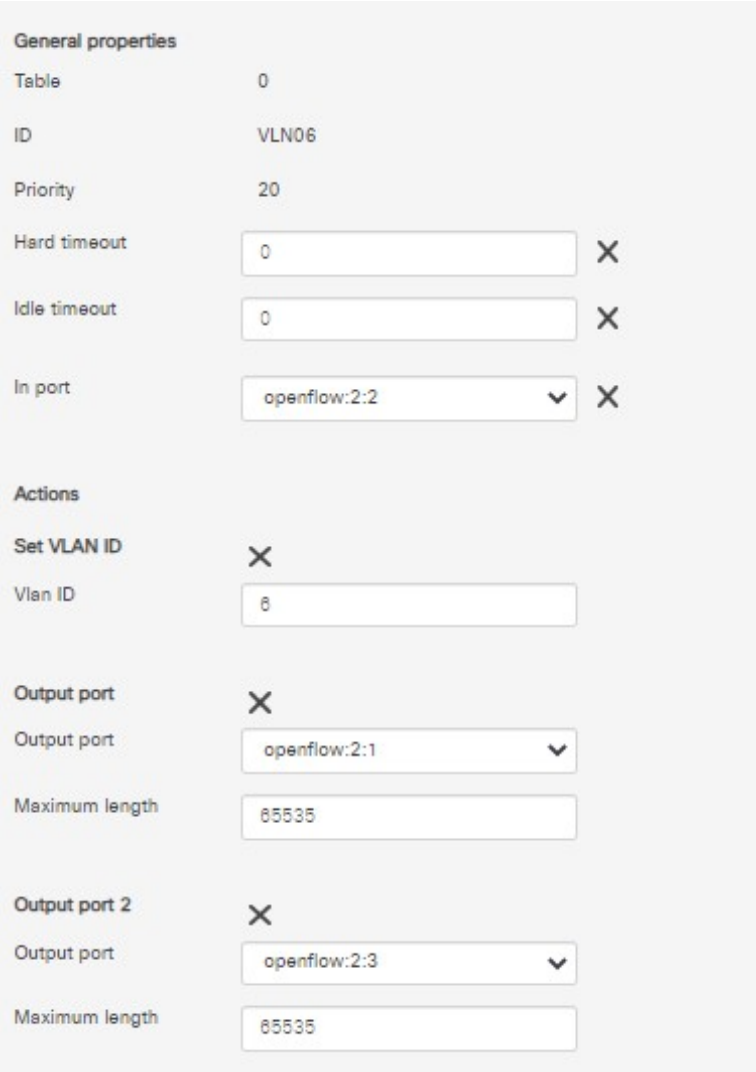

FIGURE 4.26 – Installation du VLAN 6 sur s6

Gardez à l'esprit qu'il est essentiel d'informer le contrôleur du type Ethernet utilisé pour l'identification des paquets. dans lexemple suivant la valeur 2048 (0x0800) représente le type Ethernet IPv4, ce champ qui est définie par le consortium DIX (soit Digital Equipment Corporation, Intel et Xerox), le champ EtherType indique le type du protocole encapsulé dans le champ "données" de la trame Ethernet. Il occupe deux octets. La figure suivante nous montre d'autres EtherTypes (iana.org)

Ethertype (decimal)	Ethertype (hex)	Exp. Ethernet (decimal)	Exp. Ethernet (octal)	Description
0000	0000-05DC	-	-	IEEE802.3 Length Field
0257	0101-01FF	-	-	Experimental
0512	0200	512	1000	XEROX PUP (see 0A00)
0513	0201	-	-	PUP Addr Trans (see 0A01)
	0400			Nixdorf
1536	0600	1536	3000	XEROX NS IDP
	0660			DLOG
	0661			DLOG
2048	0800	513	1001	Internet Protocol version 4 (IPv4)
2049	0801	-	-	X.75 Internet
2050	0802	-	-	NBS Internet
2051	0803	-	-	ECMA Internet
2052	0804	-	-	Chaosnet
2053	0805	-	-	X.25 Level 3
2054	0806	-	-	Address Resolution Protocol (ARP)
2055	0807	-	-	XNS Compatability
2056	0808	-	-	Frame Relay ARP
2076	081C	-	-	Symbolics Private

FIGURE 4.27 – D'autres types de liaisons

Les differentes liaisons son peuvent être illustrer sur mininet via la commande `net`

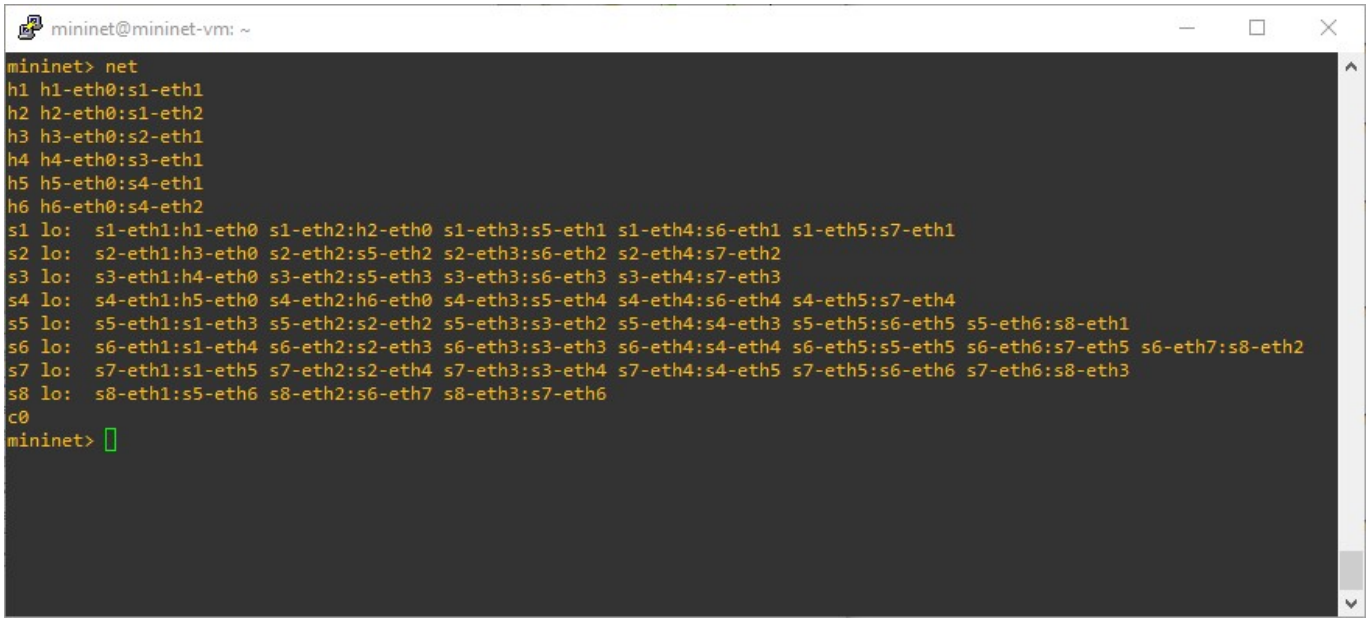

FIGURE 4.28 – Les différentes liaisons du réseau affichées sur Mininet

Bibliographie

[1] Nikolas Stathopoulos, André Peny et Georges Amar. “Formes et fonctions des points-de-réseaux”. In : *FLUX Cahiers scientifiques internationaux Réseaux et Territoires* 9.12 (1993), p. 29-45.

[2] Geoff Mulligan. “The 6LoWPAN architecture”. In : *Proceedings of the 4th workshop on Embedded networked sensors*. 2007, p. 78-82.

[3] Vivek Tiwari. “SDN and OpenFlow for beginners with hands on labs”. In : *MMDD Multimedia LLC., Kindle Edition, Northville* (2013).

[4] Jan Medved et al. “Opendaylight : Towards a model-driven sdn controller architecture”. In : *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE. 2014, p. 1-6.

[5] Bruno Astuto A Nunes et al. “A survey of software-defined networking : Past, present, and future of programmable networks”. In : *IEEE Communications surveys & tutorials* 16.3 (2014), p. 1617-1634.

[6] Olivier Flauzac, Carlos Gonzalez et Florent Nolot. “Original secure architecture for IoT based on SDN”. In : *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*. IEEE. 2015, p. 1-6.

[7] Yaser Jararweh et al. “SDIoT : a software defined based internet of things framework”. In : *Journal of Ambient Intelligence and Humanized Computing* 6.4 (2015), p. 453-461.

[8] Somayya Madakam et al. “Internet of Things (IoT) : A literature review”. In : *Journal of Computer and Communications* 3.05 (2015), p. 164.

[9] Ben Pfaff et al. “The Design and Implementation of Open {vSwitch}”. In : *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 2015, p. 117-130.

[10] Keyur K Patel, Sunil M Patel et P Scholar. “Internet of things-IOT : definition, characteristics, architecture, enabling technologies, application & future challenges”. In : *International journal of engineering science and computing* 6.5 (2016).

[11] Takayuki Sasaki et al. “SDNsec : Forwarding accountability for the SDN data plane”. In : *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2016, p. 1-10.

[12] Tryfon Theodorou et Lefteris Mamatas. “CORAL-SDN : A software-defined networking solution for the Internet of Things”. In : *2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*. IEEE. 2017, p. 1-2.

[13] Lucas Chai et Regina Reine. “Performance of UDP-Lite for IoT network”. In : *IOP Conference Series : Materials Science and Engineering*. T. 495. 1. IOP Publishing. 2019, p. 012038.

[14] Ihssane Choukri, Mohammed Ouzzif et Khalid Bouragba. “Software Defined Networking (SDN) : Etat de L’art”. In : *Colloque sur les Objets et systèmes Connectés*. 2019.

[15] Fatima Zahra Fagroud, Sanaa Elfilali, Hicham Toumi et al. “IOT et Cloud Computing : état de l’art”. In : *Colloque sur les Objets et systèmes Connectés*. 2019.

[16] Flauzac Olivier Nolot Florent et al. “Gestion dynamique et évolutive de règles de sécurité pour l’Internet des Objets”. In : *(Thèse de doctorat) Université de Reims* (2019).

[17] Yaroslav Krainyk, Olga Dvornik et Oleksii Krainyk. “Software-defined network application-aware controller for Internet-of-Things”. In : *2019 3rd International Conference on Advanced Information and Communications Technologies (AICT)*. IEEE. 2019, p. 165-169.

[18] Renjie Liu et al. “Addressless : enhancing IoT server security using IPv6”. In : *IEEE Access* 8 (2020), p. 90294-90315.

[19] Yvon Zafimahefa Andrianirina. “Développement dun réseau défini par logiciel (SDN) programmable, transparent et ouvert”. In : *(Thèse de doctorat) Université du Québec en Outaouais* (2021).

[21] Mohammed Al Jaafreh et al. “Toward integrating software defined networks with the Internet of Things : a review”. In : *Cluster Computing* (2021), p. 1-18.

[22] Avis de lAnses. “Exposition aux champs électromagnétiques liée au déploiement de la technologie «5G»”. In : (2021).

[23] “Adaptive CCA for IEEE 802.15.4 Wireless Sensor Networks to Mitigate Interference”. In : *Conférence IEEE 2010 sur la communication et les réseaux sans fil*. DOI : 10.1109/WCNC.2010.5506124.

Webographie

[24] *Bob Lantz, Brian OConnor, Cody Burkhard . Mininet ,disponible sur :* URL : http://mininet.org.

[25] *Cisco Systems. Les bases de la mise en réseau disponible sur.* URL : https://www.cisco.com/c/fr_ca/solutions/small-business/resource-center/networking/networking-basics.html.

[26] *Jekyll Minimal Mistakes .Composants de base du réseau disponible sur.* URL : https://cisco.goffinet.org/ccna/fondamentaux/composants-reseau/.

[27] *La plate-forme de contrôleur Consulté sur.* URL : http://www.opendaylight.org/.

[28] *Lee Doyle, Doyle Research . Comment les applications SDN vont changer les services réseau des couches 4 à 7 . août 2014 disponible sur.* URL : https://www.lemagit.fr/conseil/Comment-les-applications-SDN-vont-changer-les-services-reseau-des-couches-4-a-7.

[29] *Les applications et services du réseau Consulté sur.* URL : http://www.opendaylight.org/project/technical-overview/..

[30] *Linux Foundation Collaborative Projec . 2016 Production Quality, Multilayer Open Virtual Switch .disponible sur.* URL : https://www.openvswitch.org/.

[31] *Linux Foundation Collaborative Project . 1 April 2013 .disponible sur.* URL : https://air.imag.fr/index.php/Open%5C_vSwitch.

[32] *MAHAMAT CHARFADINE SALIM .2 juillet 2019.Gestion dynamique et évolutive de règles de sécurité pour lInternet des Objets consulter sur.* URL : https://www.theses.fr/2019REIMS011.pdf.

[33] *Stanislav Jamrich Charles Eckel . OpenDaylight OpenFlow Manager (OFM) App disponible sur.* URL : https://github.com/CiscoDevNet/OpenDaylight-Openflow-App.

[34] *TOUMIAT, Soltana. Problème de placement du contrôleur dans les réseaux programmable. 2020. Thèse de doctorat. Faculté : mathématique et informatique Département : informatique-Option : RTIC. consuler sur.* URL : http://dspace.univmsila.dz:8080//xmlui/handle/123456789/21693.

Résumé

L'internet des objets (IoT), est une technologie de communication récente. Considérer comme la nouvelle version d'internet, cette dernière nous permet d'inter-connecter plusieurs Objets au caractéristiques hétérogène, dans une zone géographique réduite, ce qui génère une forte densité de trafique réseau ainsi qu'une grande quantité de données, par conséquent la gestion d'une tel infrastructure réseau, avec les technologies de commutation et de routage traditionnelles, peut s'avérer très complexe en ce qui concerne la gestion et le contrôle de l'aspect d'hétérogénéité. Cependant plusieurs recherches ont été menées a ce sujet, et plusieurs solutions on été proposé, comme Ansible, Node-red, OpenRemote, Flutter, etc. Le point commun entre ces différents solutions c'est que ce sont uniquement des plat-formes dédier a une certaine tache, contrairement aux réseaux définit par logiciel (SDN), qui sont toute une infrastructure, avec ces propres platformes de gestion comme OpenDaylight, Onos, POX, NOX , etc, mais aussi ces propres équipements réseaux (Switch OpenFlow), et un nouveau protocole (OpenFlow) mis en oeuvre pour nous offrir la possibilité de programmer l'ensemble du réseau, ce qui repousse les limites en matière de topologies réseaux, et d'exploité de plus complexes. SDN offre une gestion et un contrôle centralisés de toute l'infrastructure réseau et prend en charge l'aspect d'hétérogénéité des différents noeuds existant. Dans ce mémoire nous avons étudiée et simulé une solution SDN dans le cas de l'internet des objets, nous avons introduit et définit les diffèrent composantes d'une infrastructure SDN, La technologie IoT et son architecture, ainsi que ses diffèrent impactes sur notre vie quotidienne, Ensuite nous avons cité différentes solutions SDN existantes pour faire face au défit que pose l'IoT, et simuler un réseau pour montrer l'aspect de gestion centralisé et la flexibilité qu'offre le SDN , en utilisant l'emulateur de réseaux Mininet et le contrôleur SDN OpenDylight.