République Algérienne Démocratique Et Populaire
Ministère de L'enseignement Supérieur et de la Recherche Scientifique
Université A/Mira de Bejaia

Faculté des Sciences Exactes

Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en Informatique

Option :
Administration et sécurité des réseaux informatique

Étude et mise en place d'une solution SDN pour le contrôle de connectivité dans L'IoT Cas : SONATRACH

Réalisé par :

BOUAMAMA Meroua
HADDAD Ouarda

Soutenue le 04 juillet 2023

Devant le jury :

Président	Mme SABRI Salima	M.C.B	U.A/Mira Bejaia
Examinateur	Mme TASSOULT Nadia	M.A.A	U.A/Mira Bejaia
Encadrant	Mme HOUHA Amel	M.A.A	U.A/Mira Bejaia

Année universitaire 2022/2023

Remerciements

Nous remercions « Allah » le tout puissant de nous avoir donné la santé, le courage, la volonté et la patience d'entamer et de terminer notre projet de fin d'études.

Nous tenons à exprimer notre profonde gratitude à notre encadrante Mme HOUHA Amel, pour la confiance qu'il nous à accorder en acceptant de nous encadrer dans ce mémoire. Nous la remercions pour ses conseils avisés, sa rigueur, sa gentillesse, son encouragement, sa disponibilité et sa patience qui ont grandement contribué à la qualité de notre travail.

Comme on exprime notre reconnaissance pour le directeur de la RTC Bejaïa pour nous avoir autorisé à effectuer notre stage. Nous remerciements les plus vifs vont tout particulièrement à notre encadreur de stage Mr RAHMANI Seghir pour son encadrement et orientation avec toute rigueur tout le long de notre stage inspirant en nous curiosité et passion pour promouvoir la réalisation de ce travail.

Nous tenons à adresser nos chaleureux remerciements aux membres du jury qui ont consacré leur temps et leurs efforts pour examiner et évaluer notre travail.

Nous ne saurions oublier de remercier l'ensemble de nos professeurs et les membres du département informatique de l'université A/Mira, qui nous ont prodigué leur savoir et leur soutien tout au long de notre parcours académique.

Dédicaces

Je dédie ce travail

Amis chers parents mon père et ma mère, aucune dédicace ne saurait exprimer l'amour, l'estimer, le dévouement et le respect que j'ai toujours eu pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Je vous dédie ce travail qui est le fruit de vos sacrifices et le témoignage de mon profond. Puisse Dieu, le très haut, vous accorde santé, bonheur et longue vie.

A mon oncle « dada » et sa femme « lala », je leur adresse mes remerciements les plus profonds et ma reconnaissance pour leurs efforts sincères

À mes chers frères « Zinedine et Amar ». À mes belles sœurs « Siham, Basma, Mariem et Mayssa » pour leur soutient et encouragement.

Une spéciale dédicace pour mes grands-parents décédés qui n'ont pas eu la chance d'assister à la fête de fin d'études de leur petite-fille

A ma chère binôme Meroua, pas de mots pour exprimer mes remerciements envers toi, pour ton soutien, tes encouragements que tu m'as donnés et ta patience avec moi pendant cette année.

Je tiens également à exprimer ma profonde gratitude à toute ma famille en particulier ma tante et mes oncles, tous mes amies et toutes celles et ceux qui ont participé du pré ou du loin à la réalisation de ce travail.

Ouarda.

Dédicaces

Je dédie ce travail

A ma Très chère maman,
Il est difficile de trouver les mots justes pour exprimer à quel point je t'aime profondément. Tes sacrifices incalculables ont toujours été une source d'encouragement pour moi. Tu es toujours là pour moi, depuis le tout début.

A mon cher père,
Vous m'avez donné tout votre amour et tu as fait tellement de sacrifices pour que je puisse étudier dans de bonnes conditions. Tu as été un soutien constant dans ma vie.

A mon cher grand frère Saddam,
Je suis reconnaissant de t'avoir dans ma vie. Tu as toujours été là pour moi, me guidant avec amour. Je te souhaite une vie remplie de joie, de succès et d'amour. Je me considère chanceux d'avoir un frère comme toi.

A mon frère Fawzi et sa femme Dihiya, je leur adresse mes remerciements les plus profonds et ma reconnaissance pour leurs efforts sincères

A mon frère Ayoub et à ma petite sœur bien-aimée, Takwa, je souhaite que Dieu vous donne une vie heureuse et réussie dans vos études. Je vous remercie d'être là pour moi.

A mon cher fiancé Kaci mon soutien moral et source de joie et de bonheur, ainsi toutes sa famille.

À mes amis : Imane, Ahlem, Amel, Farida, Lidya, Yasmine et à toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

A mon binôme Ouarda, ma partenaire de mémoire, je souhaite que notre amitié continue à jamais et que nous arrivions à réaliser nos rêves

Meroua.

Table des matières

Remerciements

Dédicaces

Dédicaces

Table des matières I

Liste des figures V

Liste des tableaux VII

Liste des abréviations XIII

Introduction générale 1

Chapitre 1 : Internet of things 4

1.1. Introduction 4

1.2. Objets connectés 4

1.2.1. Définition de l'objet connecté 4

1.2.2. Caractéristiques d'un objet connecté 5

1.2.3. Cycle de vie d'un objet connecté dans l'IoT 6

1.2.4. Exemples d'objets connectés 7

1.3. Internet of things 8

1.3.1. Définition de IoT 8

1.3.2. Historique de IoT 9

1.3.3. Dimension de l'IoT 10

1.3.4. Composants de système IoT 10

1.3.5. Technologies de Fonctionnement de IoT 11

1.3.6. Identification par radiofréquence (RFID) 11

1.3.6.1. Réseaux de capteurs sans fil (WSN) 12

1.3.6.2. Middleware 13

1.3.6.3. Cloud computing 13

1.3.6.4. Applications 14

1.3.7. Domaines d'application de l'Internet des objets 15

1.3.7.1. Domaine du transport et de la logistique 15

1.3.7.2. Domaine de santé 17

1.3.7.3. Domaine de l'environnement 17

1.3.7.4. Domaine personnel et social 18

1.3.7.5. Domaine de l'agriculture 19

1.3.8. Architecture de l'Internet des objets 19

1.3.8.1. Couche perception 20

1.3.8.2. Couche réseau 20

1.3.8.3. Couche traitement de données 20
1.3.8.4. Couche application 21
1.3.8.5. Couche sécurité 21
1.3.9. Les Avantages de IoT 21
1.3.10. Enjeux et les défis de l'Internet des objets 22
1.4. Conclusion 23
Chapitre 2 : La Connectivité 25
2.1. Introduction 25
2.2. Communication dans les réseaux sans fil 25
2.2.1. Réseaux avec infrastructures 25
2.2.2. Réseaux sans infrastructures 26
2.3. Les technologies de communication utilisée dans l'IoT 26
2.3.1. Technologies à courte portée 26
2.3.2. Technologies à moyenne portée 27
2.3.3. Technologies à longue portée 27
2.4. Les protocoles de IoT 29
2.4.1. Les protocoles IP 29
2.4.1.1. Internet Protocol version 4 (IPv4) 29
2.4.1.2. Internet Protocol version 6 (IPv6) 30
2.4.1.3. Intégration d'IPv6 dans l'internet des objets 31
2.4.2. Les Protocoles de communication 32
2.4.2.3. 6LoWPAN 34
2.4.2.4. IEEE 802.15.4 34
2.4.2.5. Message Queuing Telemetry Transport (MQTT) 34
2.5. La connectivité 34
2.5.1. Les défis liés à la connectivité massive de IoT : 34
2.5.2. Les solutions des problèmes de la connectivité dans IoT 35
2.5.2.1. Connectivité IoT basée sur le CS 36
2.5.2.2. Connectivité IoT basée sur NOMA 36
2.5.2.3. Connectivité IoT basée sur mMIMO 37
2.5.2.4. Connectivité IoT assistée par l'apprentissage automatique 38
2.6. Conclusion 40
Chapitre 3 : SDN dans l'IoT 42
3.1. Introduction 42
3.2. Présentation de l'organisme d'accueil 42
3.2.1. Présentation de l'entreprise SONATRACH 42
3.2.2. Organigramme de l'entreprise 43
3.2.3. Présentation de centre informatique 43

3.3. Cahier des charges 45
3.3.1. Présentation du sujet 45
3.3.2. Problématique 45
3.3.3. Objectifs de notre travail 45
3.4. Software Difined Networking SDN 46
3.4.1. Définition du SDN 46
3.4.2. Historique de SDN 46
3.4.3. Principe du SDN 47
3.4.4. Architecture de SDN 47
3.4.5. Les interfaces de programmation du SDN 49
3.4.6. Les avantages de SDN 50
3.4.7. Les protocoles SDN 51
3.4.7.1. OpenFlow 51
3.4.7.2. ForCES 55
3.4.7.3. OpFlex 55
3.4.8. Les Contrôleurs SDN 56
3.5. Architecture IoT pour les solutions SDN 56
3.6. Comparaison entre l'architecture de l'IoT et l'architecture IoT dans le SDN : 58
3.7. Inconvénients de SDN 60
3.8. Conclusion 60
Chapitre 4 : Mise en place d'une solution et simulatio 42
4.1. Introduction 62
4.2. Présentation de l'environnement de travail 62
4.2.1. Open vSwitch (OVS) 62
4.2.2. Mininet 64
4.2.3. OpenDaylight 67
4.2.4. Openflow Manager 68
4.2.5. Wireshark 68
4.3. Simulation du réseau 69
4.4. Présentation du scénario 69
4.4.1. Topologie du réseau traditionnelle de sonatrach 69
4.4.2. Topologie proposée sur ODL 70
4.4.3. Topologie proposée sur OFM 71
4.5. Vérification de fonctionnement 76
4.6. Tests de fonctionnement 77
4.6.1. Test d'accessibilité avant la création des VLANs : 77
4.6.2. Test d'accessibilité après la création des VLANs : 78

4.7. Discussion 78
4.8. Conclusion..... 78
Conclusion générale
Annexe 1
Annexe 2
Annexe 3
Bibliographie
Résumés

Liste des figures

Figure 1.1 : Objets connectés 5
Figure 1.2 : Cycle de vie d'un objet connecté 6
Figure 1.3 : Wemo Switch Smart 7
Figure1.4 : Lentille de contact intelligente 7
Figure 1.5 : T-shirt connecté 8
Figure1.6 : Internet des objets 8
Figure 1.7 : Historique de IoT 10
Figure1.8 : Les dimensions de l'IoT Les composants d'un système IoT 11
Figure1.9 : Les composants d'un système IoT 12
Figure 1.10 : (a) étiquette passive, (b) étiquette semi-passive, (c)étiquette active 14
Figure 1.11 : Domaines d'application de IoT 16
Figure 1.12 : Système de transport basée sur l'IoT 16
Figure1.13 : Exemple de système de santé intelligent 17
Figure 1.14 : Maison intelligent 18
Figure 1.15 : IoT dans le domaine de l'agriculture 19
Figure 1.16 : Architecture de l'internet des objets 20

Figure 2.1 : Technologies de communications 29
Figure 2.2 :Pile de protocole IP smart objet 32
Figure 2.3 :Pile de protocole CoAP 33
Figure 2.4 :Diagramme du défi à relever pour les technologies existantes et les solutions prometteuses 35
Figure2.5 : Illustration de l'activité éparse des utilisateurs dans un MTC massif 36
Figure 2.6 :Systèmes NOMA simplifiés dans le domaine de la puissance 37
Figure 2.7 :Illustration des systèmes mMIMO 37

Figure 3.1 : Organigramme de Sonatrach. 43
Figure 3.2 : Organigramme du centre informatique 43
Figure3.3 : Comparaison entre un réseau traditionnel et un réseau SDN 48
Figure 3.4 : Architecture d'un réseau SDN 49
Figure3.5 : processus de communication entre le switch et le contrôleur 52
Figure 3.6 : Architecture IoT pour les solutions SDN 57
Figure 4.1 : Les composants et les interfaces d'Open vSwitch 63
Figure 4.2 : Affichage de la commande show Dovs. 64
Figure 4.3: Fichier test4.py. 66
Figure 4.4 : Affichage de la commande show help. 67
Figure 4.5: Création des éléments du réseau dans Mininet. 67
Figure 4.6: Topologie du réseau pour sonatrach. 70
Figure 4.7 : Topologie du scénario affichée sur ODL. 71
Figure 4.8 : Topologie du scénario affichée sur OFM. 72
Figure 4.9 : Accès à la gestion de flux. 72
Figure 4.10 : Programmation de flux sur OFM. 73
Figure 4.11 : Installation d'un filtre de niveau 2 sur s7. 73
Figure 4.12 : Installation de l'entrée flux 305 sur OFM. 74
Figure 4.13: Capture wireshark d'un Openflow 75
Figure 4.14: Capture wireshark d'un Openflow et LLDP. 75
Figure 4.15 : Table de flux SW.OF.7. 76
Figure 4.16 : Résultat de la commande dump-flows s2. 77

Figure 4.17 : Résultat du test ping. 77
Figure 4.18 : Test Avant la création des VLANs. 77
Figure 4.19 : Test après la création des VLANs. 78
Figure 4.20 : lancement de OpenDaylight 87
Figure 4.21 : Interface de connexion à de OpenDayligh. 88
Figure 4.22: Interface web d'OpenDayligh..... 89
Figure 4.23 : Le fichier env.module.js pour introduire l'adresse IP du contrôleur..... 91
Figure 4.24 : Lancement de OpenFlow Manager..... 91
Figure 4.25 : Interface utilisateur d'OFM 92

Liste des tableaux

Tableau 2.1 : Résumé des points forts et des limites des technologies prometteuses pour la connectivité massive 39
Tableau 3.1 : Structure d'une entrée de table de flux d'un commutateur openflow 1.0 52
Tableau 3.2 : Fonctionnalités des différentes spécifications d'OpenFlow. 53
Tableau 3.3 : Contrôleurs SDN 56
Tableau 3.4 : Comparaison entre l'architecture traditionnelle de l'IoT et l'architecture IoT pour les solutions SDN 59
Tableau 4.1 : Plan d'adressage. 71

Liste des abréviations

3G	**T**roisième **G**énération
3GPP	**3**rd **G**eneration **P**artnership **P**roject
4G	**Q**uatrième **G**énération
5G	**C**inquième **G**énération
6LoWPAN	IPv**6** **Lo**w Power **W**ireless **P**ersonal **A**rea **N**etworks
ANT	**A**dvanced and **A**daptive **N**etwork **T**echnology
API	**A**pplication **P**rogramming **I**nterface
BSS	**B**asic **S**ervice **S**et
CDM	**C**ode-**D**omain **M**ultiplexing
CoAP	**Co**nstrained **A**pplication **P**rotocol
CS	**C**ompressive **S**ensing
CSMA/CA	**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
DHCP	**D**ynamic **H**ost **C**onfiguration **P**rotocol
EPC	**E**lectronic **P**roduct **C**ode
GPRS	**G**eneral **P**acket **R**adio **S**ervices
GPS	**G**lobal **P**ositioning **S**ystem
GSM	**G**lobal **S**ystem for **M**obile communication
GSN	**G**lobal **S**ensor **N**etworks
HART	**H**ighway **A**ddressable **R**emote **T**ransducer
IaaS	**I**nfrastructure-**a**s-**a**-**S**ervice
IBSS	**I**ndependent **B**asic **S**ervice **S**et
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IoT	**I**nternet **o**f **T**hings
IP	**I**nternet **P**rotocol
IPSec	**I**nternet **P**rotocol **Sec**urity
IPV4	**I**nternet **P**rotocol **v**ersion **4**
IPV6	**I**nternet **P**rotocol **v**ersion **6**
ISM	**I**ndustriel, **S**cientifique et **M**édical
ISM	Industriel, Scientifique et Médical
LCD	**L**iquid **C**rystal **D**isplay
LFB	Logical Function Block
LG	**L**ucky-**G**oldstar
LLDP	Link Layer Discovery Protocol
LoRa	**Lo**ng **Ra**nge
LoRaWAN	**Lo**ng **Ra**nge **W**ide **A**rea **N**etwork
LoWPAN	**Lo**w Power **W**ireless **P**ersonal **A**rea **N**etworks
LPWAN	Low Power Wide Area Networks
LTE	**L**ong **T**erm **E**volution
M2M	**M**achine **t**o **M**achine
MAC	**M**edia **A**ccess **C**ontrol
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput

ML	**M**achine **L**earning
Mmimo	**m**assive **M**ultiple-**I**nput **M**ultiple **O**utput
MPLS	Multiprotocol Label Switching
MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport
MTC	**M**achine **T**ype **C**ommunications
NB-IoT	**N**arrow **B**and-**I**nternet **o**f **T**hings
NFC	**N**ear **F**ield **C**ommunication
NOMA	**N**on-**O**rthogonal **M**ultiple **A**ccess
NTT	Numéro Technique Temporaire.
ODL	OpenDaylight
OFM	Openflow Manager
ONF	Open Networking Foundation
OSI	Open Systems interconnection
OVS	Open vSwitch
PAN	**P**ersonal **A**rea **N**etwork
PC	Personal Computer
PDM	**P**ower-**D**omain **M**ultiplexing
PHY	Physical Layer
QoS	**Q**uality **o**f **S**ervice
RA	**R**andom **A**ccess
REST	**Re**presentational **S**tate **T**ransfer
RFID	**R**adio **F**requency **Id**entification
SaaS	**S**oftware **a**s **a** **S**ervice
SB	**S**tation de **B**ase
SDN	**S**oftware-**D**efined **N**etworking
SDR	**S**oftware-**D**efined **R**adio
SIC	**S**uccessive **I**nterference **C**ancellation
SMS	**S**hort **M**essage **S**ystem
SSH	Secure Socket Shell
TCP	**T**ransmission **C**ontrol **P**rotocol
TI	**T**echnologie de l'**I**nformation
TLS	**T**ransport **L**ayer **S**ecurity
UDP	**U**ser **D**atagram **P**rotocol
UM	**U**nites **M**obiles
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
VM	Virtual Machine
WEB	**W**orde Wide Web
Wi-Fi	**W**ireless **F**idelity
WIFI	**W**ireless **F**idelity
WPAN	**W**ireless **P**ersonal **A**rea **N**etworks
WSN	**W**ireless **S**ensor **N**etwork
Z-Wave	**Z**ensys **Wave**

Introduction générale

Introduction générale

Dans notre ère moderne, nous assistons à l'émergence de nombreuses applications innovantes reposant sur des concepts novateurs tels que les réseaux intelligents, les maisons intelligentes et les transports intelligents. Ces systèmes d'infrastructure sont en train de relier notre monde de manière bien plus étroite que nous ne l'aurions jamais imaginé. Ce phénomène est rendu possible grâce à un concept révolutionnaire : l'Internet des objets.

L'Internet des objets ou l'*Internet of Things* (IoT) est une révolution technologique qui est en train de transformer notre monde connecté. Il s'agit d'un réseau d'objets physiques interconnectés, dotés de capteurs, de logiciels et de technologies de communication, qui collectent, échangent et analysent des données via Internet. L'importance de l'IoT réside dans sa capacité à créer un écosystème numérique où les objets physiques deviennent des entités intelligentes qui interagissent entre elles et avec les humains. Cette connectivité permet d'optimiser ces opérations, d'améliorer la productivité, de créer de nouvelles expériences utilisateur et de résoudre des problèmes complexes.

En raison du manque d'innovation dans le domaine des réseaux, le déploiement d'applications liées à l'Internet des objets a été entravé par de nombreuses contraintes. Bien que le concept de l'IoT soit relativement simple, la gestion des communications et des traitements associés à ces dispositifs connectés pose de nombreux problèmes en raison de leurs capacités limitées. En effet, depuis plusieurs années, il est très difficile, voire impossible d'innover ou apporter des changements au réseau. Aussi les taches de configuration et de gestion des réseaux sont plus complexes. Une des raisons de cette difficulté d'évoluer, est le fort couplage qui existe entre le plan de contrôle et le plan de données des équipements d'interconnexions dans les architectures des réseaux actuels. C'est dans ce contexte qu'a apparu le concept des réseaux définis par les logiciels (Software Defined Networking ou SDN), afin de répondre à la rigidité architecturale des réseaux actuels, notamment en les rendant plus programmables. L'idée principale de ce nouveau paradigme, est de sortir la partie intelligente des équipements d'interconnexions, et la placer vers un seul point de contrôle appelé contrôleur, ce dernier fournit une vue centrale de réseau, ce qui simplifie la gestion et la configuration de réseau. Avec le SDN, les administrateurs réseaux n'ont plus à configurer chaque équipement séparément comme dans les réseaux traditionnels ou le risque d'erreur est élevé. Cette solution offre une flexibilité et une gestion simplifiée, ce qui en fait une réponse adaptée aux besoins de l'IoT.

L’entreprise SONATRACH de Bejaia fait partie de celles qui utilisent les outils informatiques les plus robustes pour la gestion et la sécurisation de leur réseau. L’objectif de notre travail pondant la période de stage est de simulé la solution SDN (Software-Defined Networking) dans l'Internet des Objets qui permet de gérer et de contrôler les réseaux IoT de l’entreprise de manière centralisée et programmable.

Organisation du mémoire

Ce mémoire se divise en quatre chapitres et structuré comme suit :

Dans le premier chapitre nous allons présenter une vue générale sur l’objet connectée et nous allons aborder le concept de l'internet des objets en détail avec leur Technologies, défis et domaines d’application.

Dans le deuxième chapitre nous nous sommes concentrés sur la connectivité dans l'IoT. Nous allons examiner les réseaux de communications avec ses technologies, par la suite nous allons détailler les protocoles utilisés dans l’IoT et les défis liés à la connectivité avec ses solutions.

Dans le troisième chapitre nous allons présenter l’entreprise SONATRACH, l’organisme d’accueil, le centre informatique et nous établirons un cahier des charges qui exprime les spécifications de notre projet. Ensuite, nous allons aborder la solution du réseau défini par logiciel (SDN) proposé au l’entreprise.

Le quatrième chapitre consacré à la définition des différents outils et logiciels ayant servis à l’élaboration de notre implémentation, et la simulation d'un réseau SDN pour illustrer la gestion centralisée et la flexibilité qu'il offre dans le contexte de l'IoT.

Enfin, nous terminons ce mémoire par une conclusion générale qui résume nos objectifs estimés et quelques perspectives.

Chapitre 1: Internet of things

Chapitre 1: Internet of things

1.1. Introduction

Pendant de nombreuses années, Internet a connu une évolution remarquable. La dernière avancée majeure est l'exploitation de ce réseau mondial pour la communication entre objets, un phénomène connu sous le nom d'Internet des objets (IoT). Cette évolution englobe divers secteurs et applications, allant d'appareils simples à des déploiements massifs de technologies intégrées interconnectées en temps réel sur plusieurs plateformes.

Ce premier chapitre sera structuré en deux parties, dans la première partie nous commencerons par définir les objets connectés ainsi que nous donnerons des exemples sur ces objets les plus utilisées dans un système IoT. Dans la deuxième partie, nous allons introduire les concepts généraux de l'IoT, en suite nous présenterons son fonctionnement et son architecture la plus élémentaire.

1.2. Objets connectés

1.2.1. Définition de l'objet connecté

L'IoT repose avant tout sur les objets connectés, qu'ils soient matériels ou logiciels. A cette dichotomie, il faut ajouter les humains qui associent les capacités matérielles et logicielles. L'objet connecté est d'abord un objet qui a une fonction mécanique et/ou électrique propre, il peut soit être conçu directement connectable, soit il est déjà existant et la connectivité est rajoutée à postériori. Il a la capacité de capter une donnée grâce aux capteurs qu'il possède et de l'envoyer, via le réseau Internet ou d'autres technologies, pour que celle-ci soit analysée et visualisée sur des tableaux de bord dédiés [1].

Un objet connecté peut effectuer généralement deux rôles [2] :

- Un rôle de capteur pour surveiller l'apparition d'un événement ou récupérer des informations (capteur de température, capteur de présence, mesure de la distance…).

- Un rôle d'actionneur pour réaliser une action suite à un événement spécifique mesuré au détecté (alerte via SMS en cas de danger, allumage du ventilateur à distance…).

FIGURE 1.1 : Objets connectés [3].

Il existe plusieurs types d'objet connecté :

- **Electroniques :** comme les véhicules connectés en 4G pour optimiser les performances.
- **Electriques :** tout ce qui est de la domotique, allumage à distance...
- **Non électriques :** vêtements, équipements…
- **Capteurs environnementaux :** chaleur, proximité...

1.2.2. Caractéristiques d'un objet connecté

Généralement, un objet connecté est caractérisé par [4] :

- **Identité :** pour que les objets soient gérables il est essentiel que chaque objet connecté possède une identité unique, qu'il lui propre et qui le distingue des autres objets du système.
- **Interactivité :** Un objet n'a pas besoin d'être connecté à un réseau à tout moment.
- **Programmable :** l'objet connecté doit être programmé et piloté à distance via un ordinateur, une tablette ou un Smartphone.

- **Sensibilité :** un objet a la capacité de percevoir son environnement et peut collecter ou transmettre des informations à celui-ci.

- **Autonomie :** cette caractéristique est peut-être la caractéristique la plus importante pour l'objet connecté. On désigne par cette caractéristique la capacité de l'objet d'agir sans l'intervention d'un tiers.

1.2.3. Cycle de vie d'un objet connecté dans l'IoT

Dans l'IoT, les objets intelligents passent par trois étapes représenter dans la « figure 1.2 » : la phase préparatoire, la phase opérationnelle et la phase de maintenance [5].

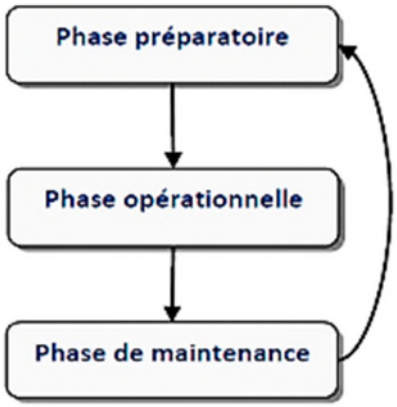


FIGURE 1.2 : Cycle de vie d'un objet connecté [5].

- **La phase préparatoire :** déploiement des objets (capteurs, tags), leur configuration avec les informations nécessaires, par exemple les identificateurs, les clés de sécurité, etc.

- **La phase opérationnelle :** dans la phase opérationnelle, l'objet connecté se met à réaliser sa mission qui diffère d'une application à une autre.

- **La phase de maintenance :** effectuer des mises à jour, régler les problèmes en faisant d'éventuelles et réparations des objets en cas de défaillances par exemple. Il est même possible de remplacer carrément des objets et redémarrer à nouveau à partir de la phase préparatoire.

1.2.4. Exemples d'objets connectés

- **WeMo Switch Smart :**

Une prise intelligente montré dans la « figure 1.3 ». Elle se branche dans une prise régulière, accepte le cordon d'alimentation de n'importe quel appareil et peut être utilisé pour l'allumer et l'éteindre sur un calendrier programmé ou lorsque vous appuyez sur un bouton sur votre smart phone. Il Surveille également la quantité d'énergie utilisée par vos appareils, en vous aidant à rendre votre maison plus économe en énergie. Vous pouvez voir quand les fiches sont activées, combien d'énergie elles utilisent [6].

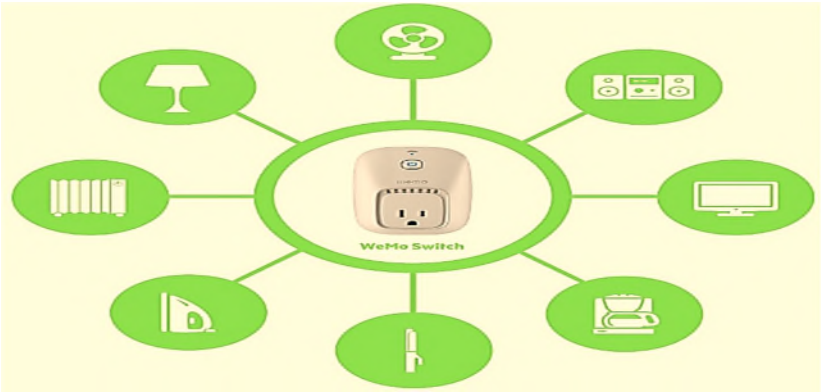


FIGURE 1.3: Wemo Switch Smart [7].

- **La lentille de contact intelligente**

En ce moment, Google et Microsoft travaillent sur des lunettes intelligentes. D'autres ont décidé de créer des lentilles de contact intelligentes. Les chercheurs de l'Université de Gand se sont attelés à la tâche et viennent de sortir un prototype. Cette lentille abrite un écran LCD capable de vision des images directement sur votre lentille [6].

FIGURE 1.4 : Lentille de contact intelligente [8].

- **Le t-shirt connecté**

Le t-shirt de sport connecté, composé de nombreux capteurs, qui permet à son utilisateur de recueillir une multitude d'informations (altimètre, cardio-fréquence mètre, accéléromètre) et qui lui sert également de GPS. Reliées au Smartphone, les données sont ensuite sauvegardées à l'aide d'une application qui permet également de suivre ses performances [9].

FIGURE 1.5 : T-shirt connecté [10].

1.3. Internet of things

1.3.1. Définition de IoT

Il existe plusieurs définitions de l'internet des objets, qu'ils sont :

FIGURE 1.6 : Internet des objets [11].

« La technologie IoT est considérée comme l'émergence de l'internet du futur, certaines la définissent comme des objets ayant des identités et des personnalités virtuelles, opérant dans des espaces intelligents et utilisant des interfaces intelligentes pour se connecter et communiquer au sein de contextes d'usages variés » [12].

« L'Internet des objets est un réseau qui relie et combine les objets avec l'Internet, en suivant les protocoles qui assurent leurs communications et échange d'informations à travers une variété de dispositifs » [13].

« L'IoT peut se définir aussi comme un réseau de réseaux qui permet via des systèmes d'identification électroniques normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi, de pouvoir récupérer, stocker, transférer et traiter les données sans discontinuité entre les mondes physiques et virtuels. L'IoT est une combinaison d'innovations technologique récent et de solutions déjà existantes. Chaque objet est muni d'une identification électronique unique capable de lire et transmettre à travers un protocole dans les réseaux internet. Il est nécessaire cependant de définir la nature de l'objet, ses fonctionnalités, sa position dans l'espace, l'historique de ses déplacements, etc. pour effectuer ce lien entre physique et virtuel, le dispositif technique doit donc modéliser des contextes réels et les rendre virtuel » [14].

1.3.2. Historique de IoT

Dans cette section, nous citons les évènements les plus marquants sur le chemin de la concrétisation de l'IoT. Le concept d'un réseau de dispositifs intelligents a été évoqué pour la première fois en 1982, avec le premier appareil connecte à Internet à l'Université Carnegie Melon capable de signaler à son inventaire si les boissons nouvellement chargées sont bien froides. Ainsi, en 1991, Mark Weiser a introduit l'informatique omniprésente à travers son papier intitulé : "L'ordinateur du 21ème siècle " et a présente d'avance la vision contemporaine de l'Internet des objets. Ensuite, en 1998, l'informatique ubiquitaire a commencé d'attirer l'attention par le fait qu'elle permettrait l'incorporation flexible et efficace de l'informatique dans la vie quotidienne. Après, en 2000 la société LG annonce son premier réfrigérateur intelligent connecté à Internet. De plus, la technologie RFID (Radio Frequency Identification) qui est l'une des technologies constitutionnelles de l'IoT, a commencé à être massivement déployée vers les années 2003 et 2004. D'autre part, une initiative très intéressante a été prise

en 2008, un groupe de recherche appelé Ipso Alliance s'est consacré à promouvoir l'utilisation du protocole IP pour les réseaux d'objets miniatures intelligents.

De nombreux travaux de recherches ont été succèdes et se sont tous concentrés autour de la réalisation, dans les meilleures conditions, de la vision de l'Internet des objets et la mener à sa maturité en dépit de tous les défis soulevés. Cela avec la considération des progrès technologiques continus dans le marché des dispositifs intelligents et dans le domaine de technologies de télécommunication [15].

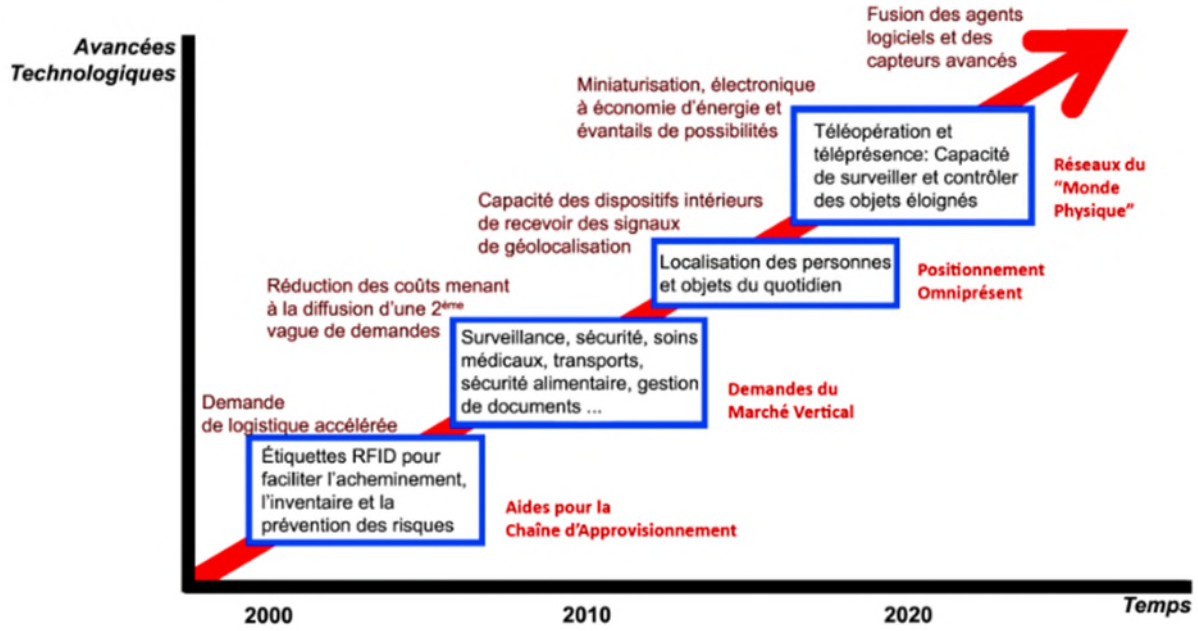


FIGURE 1.7 : Historique de IoT.

1.3.3. Dimension de l'IoT

Cette vision de l'internet des objets introduira une nouvelle dimension aux technologies de l'information et de la communication, qui permettent aux personnes de se connecter à n'importe quel moment depuis n'importe quelle place à n'importe quel objet [16]. La « figure 1.7 » présent une nouvelle dimension pour l'IoT :

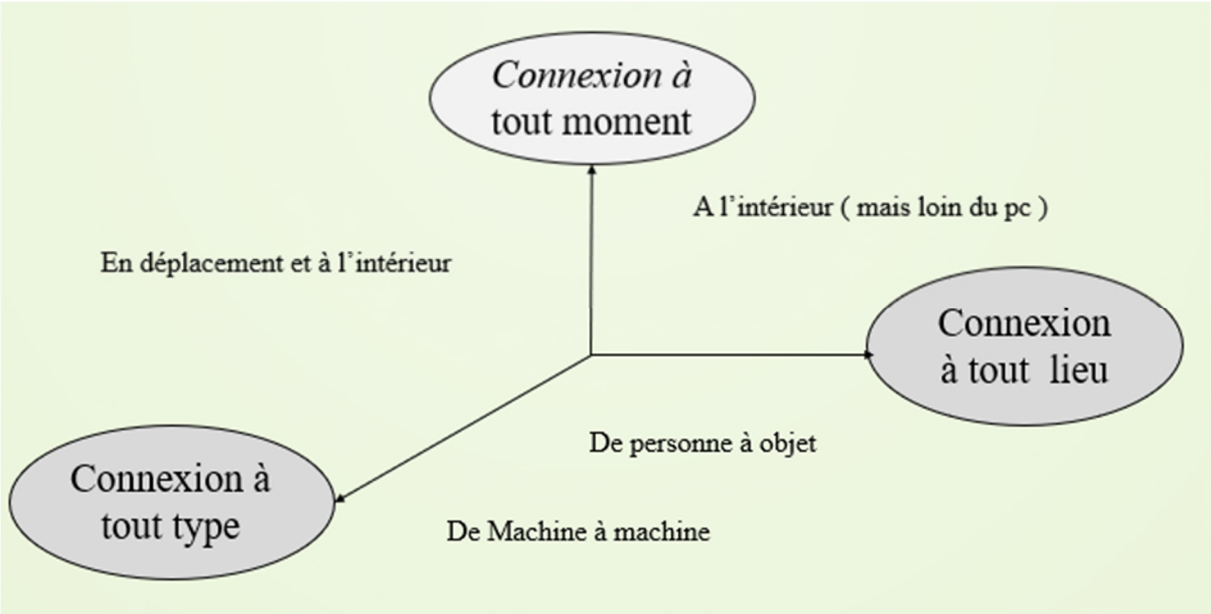


FIGURE 1.8 : Dimensions de l'IoT [16].

1.3.4. Composants de système IoT

Un système IoT est composé :

- D'un réseau d'objets connectés.
- Des passerelles et réseaux de communication sans fil (Wifi et Bluetooth).
- Des protocoles réseau (Sigfox et LoRa).
- D'API et des plateformes pour collecter et traiter les données.
- D'hébergeur ou des fournisseurs de cloud computing pour stoker les informations.
- Des logiciels et applications pour visualiser, trier et afficher plus facilement les données.
- D'une informatique en périphérie (Edge Computing) pour déplacer si nécessaire le traitement et l'analyse de données près de l'utilisateur final afin de réduire toute latence [17].

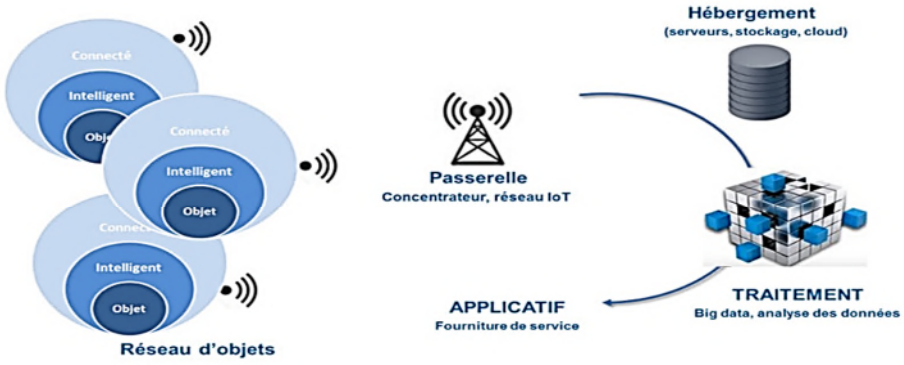


FIGURE **1.**9 : Les composants d'un système IoT [17].

1.3.5. Technologies de Fonctionnement de IoT

L'Internet of Things permet l'interconnexion des différents objets intelligents via l'Internet. Ainsi, pour son fonctionnement, plusieurs systèmes technologiques sont nécessaires. Citons quelques exemples de ces technologies.

L'IoT désigne diverses solutions techniques (RFID, TCP/IP, technologies mobiles, etc.) qui permettent d'identifier des objets, de capter, stocker, traiter, et transfère des données dans les environnements physiques, mais aussi entre des contextes physiques et des univers virtuels.

D'après [19], cinq technologies IoT sont largement utilisées pour le déploiement des produits et des services axés sur l'Internet des objets qui connaissent du succès :

1.3.5.1 Identification par radiofréquence (RFID)

L'identification par radio fréquence (RFID) permet l'automatisation et la saisie de données à l'aide d'ondes radio, d'une balise et d'un lecteur. L'étiquette peut stocker plus données que les codes-barres traditionnels. L'étiquette contient des données sous la forme du Electronic Product Code (EPC), un système mondial d'identification par RFID développé par l'Auto-id Center. Trois types d'étiquettes sont utilisés :

- **Les étiquettes RFID passives** : dépendent de l'énergie radiofréquence transférée du lecteur à l'étiquette pour alimenter l'étiquette, elles ne sont pas alimentées par batterie.

- **Les étiquettes RFID actives** : possèdent leur propre batterie et peuvent favoriser la communication avec un lecteur, elles peuvent contenir des capteurs externes pour surveiller la température, la pression, les produits chimiques et d'autres conditions. Les

étiquettes RFID actives sont utilisées dans la fabrication, les laboratoires hospitaliers et la gestion des biens de TI par télédétection.

- **Les étiquettes semi-passives RFID**: utilise des piles pour alimenter la micro-puce tout en communiquant en tirant l'énergie du lecteur. Les étiquettes RFID actives et semi-passives coûtent plus cher que les étiquettes passives.

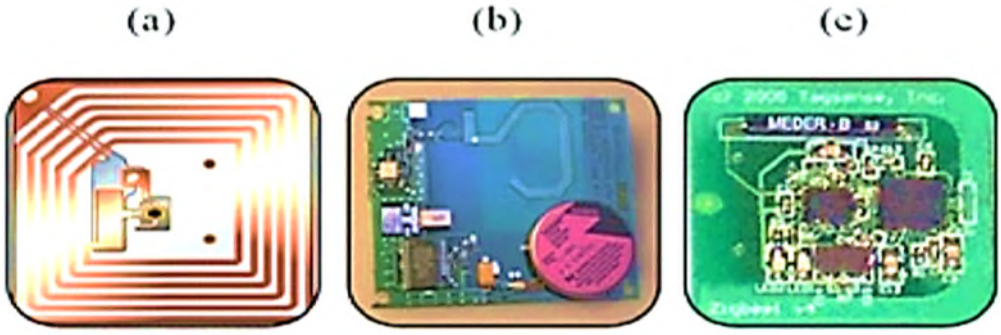


FIGURE 1.10 : (a) étiquette passive, (b) étiquette semi-passive, (c)étiquette active [18].

1.3.5.2 Réseaux de capteurs sans fil (WSN)

Les réseaux de capteurs sans fil (WSN) se composent de dispositifs autonomes spatialement distribués équipés de capteurs pour surveiller les conditions physiques ou environnementales et peuvent coopérer avec les systèmes RFID pour mieux suivre l'état des objets telles que leur emplacement, leur température et leurs mouvements. WSN permettent différentes topologies de réseau et de communication multi-hop.

Les récents progrès technologiques dans le domaine des circuits intégrés de faible puissance et des communications sans fil ont rendu disponibles des appareils miniatures efficaces, peu coûteux et de faible puissance pour les applications WSN. L'IoT existe à un niveau supérieur au WSN. En d'autres termes, WSN est souvent une technologie utilisée dans un système IoT. Une grande collection de capteurs, comme dans un réseau maille, peut être utilisée pour collecter individuellement des données via un routeur vers internet dans un système IoT. Par exemple, WSN utilisé dans la logistique de la chaîne du froid qui utilise des méthodes d'emballage thermique et réfrigéré pour transporter des produits sensibles à la température. WSN sont également utilisés pour les systèmes de maintenance et de suivi.

1.3.5.3 Middleware

Middleware est une couche logicielle interposée entre les applications logicielles pour faciliter la communication et l'entrée/sortie pour les développeurs de logiciels. Sa

caractéristique de cacher les détails des différentes technologies est fondamentale pour libérer les développeurs de l'IoT de services logiciels qui ne sont pas directement pertinents pour les applications spécifiques de l'IoT. Middleware a gagné en popularité dans les années 1980 en raison de son rôle majeur dans la simplification de l'intégration des anciennes technologies dans les nouvelles. Il a également facilité le développement de nouveaux services dans l'environnement d'informatique répartie. Une infrastructure distribuée complexe de l'IoT avec de nombreux dispositifs hétérogènes nécessite de simplifier le développement de nouvelles applications et de services, de sorte que l'utilisation de middleware est un ajustement idéal avec le développement d'applications IoT. Par exemple, Global Sensor Networks (GSN) est une plate-forme intermédiaire de capteurs open source qui permet le développement et le déploiement de services de capteurs avec un effort de programmation presque nul. La plupart des architectures de middleware pour l'IoT suivent une approche orientée service afin de soutenir une topologie réseau inconnue et dynamique.

1.3.5.4 Cloud computing

L'informatique en cloud est un modèle d'accès à la demande à un ensemble partagé de ressources (configurables) qui peuvent être fournis en tant qu'infrastructure "infrastructure as a service" (IaaS) ou en tant que logiciel "software as a service"(SaaS). L'un des résultats les plus importants de l'Internet des objets est l'énorme quantité de données générées par les appareils connectés à Internet. De nombreuses applications de l'Internet des objets nécessitent un stockage massif de données, une vitesse de traitement énorme pour permettre la prise de décisions en temps réel, et des réseaux à large bande à haute vitesse pour diffuser des données, de l'audio ou de la vidéo. L'informatique en cloud offre une solution idéale pour gérer d'énormes flux de données et les traiter pour le nombre sans précédent d'appareils IoT et d'humains en temps réel.

1.3.5.5 Applications

L'Internet des objets facilite le développement d'une myriade d'applications de l'Internet des objets axées sur l'industrie et propres aux utilisateurs. Alors que les appareils et les réseaux assurent la connectivité physique, les applications de l'Internet des objets permettent des interactions entre dispositifs et entre humains de manière fiable et robuste. Les applications de l'Internet des objets sur les appareils doivent s'assurer que les données et les messages ont été reçus et mis en œuvre correctement et en temps opportun. Par exemple, les applications de transport et de logistique surveillent l'état des marchandises transportées (comme les fruits frais coupés, la viande et les produits laitiers). Pendant le transport, l'état de conservation

(température, humidité, choc) est surveillé en permanence et des mesures appropriées sont prises automatiquement pour éviter la détérioration lorsque la connexion est hors de portée. Par exemple, Fedex utilise Senseaware pour surveiller la température, l'emplacement et d'autres signes vitaux d'un colis, y compris quand il est ouvert et s'il a été altéré en cours de route. Il est important que les applications de l'Internet des objets soient construites avec de l'intelligence afin que les dispositifs puissent surveiller l'environnement, identifier les problèmes, communiquer entre eux et potentiellement résoudre les problèmes sans intervention humaine.

1.3.6 Domaines d'application de l'Internet des objets

Les progrès technologiques, la diversité des besoins des utilisateurs potentiels et les potentialités offertes par l'IoT rendent possible le déploiement d'un nombre considérable d'applications. Il existe de nombreux domaines et environnements dans lesquels de nouvelles applications seraient susceptibles d'améliorer le sort de nos vies personnelles, de nos entreprises et de nos communautés. Ces environnements sont désormais équipés d'objets intelligents. Permettre à ces objets dans notre environnement de communiquer entre eux et de traiter les informations collectées permettra de déployer une vaste gamme d'applications imprévues. Dans ce qui suit, nous présentons les domaines d'application de l'internet des objets les plus pertinents :

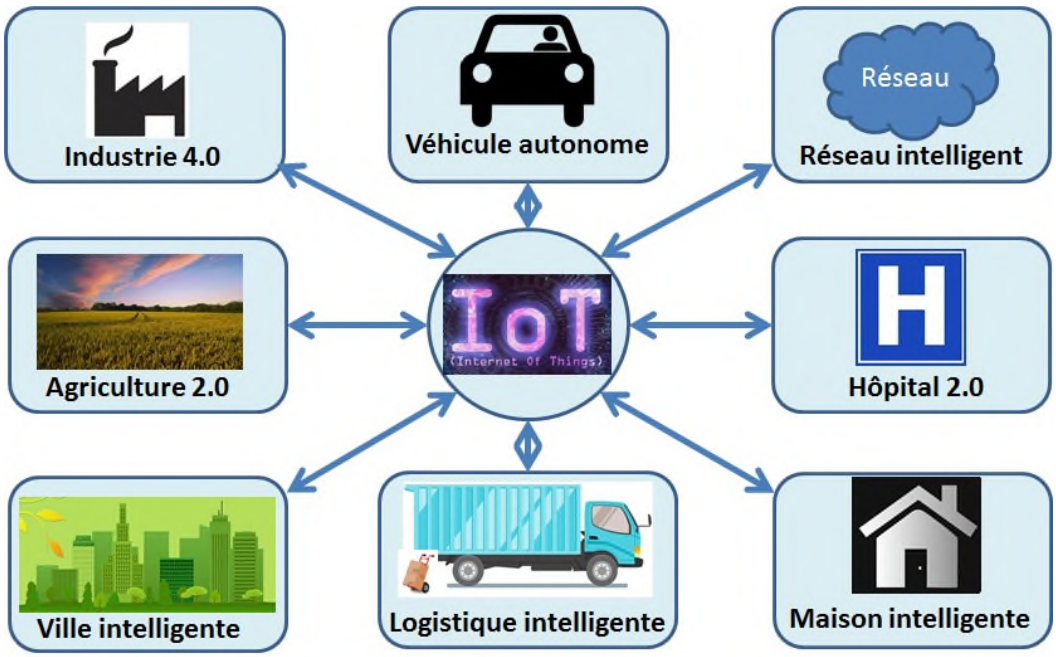


FIGURE 1.11 : Domaines d'application de IoT [20].

1.3.6.1 Domaine du transport et de la logistique

Plusieurs moyens de transport (les voitures, les trains, les bus, etc.) sont actuellement équipés de capteurs, d'étiquettes, d'actionneurs et de puissance de traitement. Les routes, les rails et les espaces publics sont également équipés RFID et de capteurs afin de collecter des informations utiles et vitales pour une bonne gestion de trafic. L'utilisation de domaine du transport vise à optimiser l'utilisation des infrastructures physiques de la ville et améliorer la qualité de vie des citoyens. Les voitures et les routes peuvent échanger des informations importantes par le biais d'un service approprié. Ces informations seront envoyées aux conducteurs et aux transports privés pour les aider à trouver un meilleur itinéraire et éviter les accidents et les embouteillages, ce qui leur permettre d'économiser de l'énergie, du temps et de l'argent, et de réduire le nombre de kilomètres parcourus. Ainsi, il contribue à réduire les émissions et à une moindre pollution. En autre, plusieurs applications concernant les services de transport Peuvent être équipées d'une étiquette NFC. Ainsi l'utilisateur peut obtenir des informations sur plusieurs prestations sur le web en utilisant son téléphone portable, comme des cartes touristiques qui lui permettent d'économiser beaucoup de temps et d'argent [20].

FIGURE 1.12 : Système de transport basée sur l'IoT [21].

Il existe de nombreux scénarios de logistique intelligente, notamment le transport logistique, l'entreposage, le chargement/déchargement, le transport, l'emballage, le traitement de la distribution, et de l'information. Par exemple, le transport logistique est l'activité économique la plus importante parmi les composants des systèmes logistiques. Il consiste à expédier des articles d'un endroit à un autre en utilisant des installations et des outils [22]. D'autre part, l'entreposage logistique est un composant important de la chaine

d'approvisionnement logistique. Il s'agit d'activités qui contrôlent, classent et gèrent l'inventaire.

1.3.6.2 Domaine de santé

Les applications IoT jouent un rôle essentiel dans le domaine de santé, ou elles sont utilisées pour aider les patients à des fins de surveillance et de suivi. Les patients portent des capteurs médicaux pour suivre leurs paramètres de santé vitaux, tels que la pression artérielle et la température corporelle. Ces capteurs collectent et analysent les données, en cas d'activités inhabituelle, le capteur déclenche l'alarme et la transmet à des centres médicaux distants afin d'évaluer l'état du patient, puis le personnel médical prendra les mesures appropriées pour le patient. Ces informations peuvent être utiles à l'assistance médicale en cas d'urgence tel que l'hôpital le plus proche sera alerté et l'intervention sera rapide, ce qui signifie que la vie du patient et sauvée et que les frais d'hospitalisation sont réduits [23].

FIGURE 1.13 : Exemple de système de santé intelligent [23].

1.3.6.3 Domaine de l'environnement

Un environnement intelligent facilite l'utilisation de notre environnement et rend notre vie meilleure et confortable à la maison, au bureau et dans les usines, etc. Il est utilisé l'intelligence de la technologie des captures intégrés intelligents pour surveiller les paramètres critiques du domaine, ce qui rend la vie plus détendue et confortable sous plusieurs aspects. Par exemple, la gestion de l'énergie via le contrôle des équipements domestiques tels que les machines à laver et les climatiseurs, contrôle de l'éclairage des pièces en fonction de l'heure de

la journée et ils économisent de l'énergie en éteignant les équipements électriques lorsqu'ils ne sont pas nécessaires [20].

FIGURE 1.14 : Maison intelligent [24].

1.3.6.4 Domaine personnel et social

Plusieurs applications IoT sont utilisées dans le domaine personnel et social. Ces applications permettent de construire des relations sociales entre les personnes d'une maniéré différent. Elles sont équipées de dispositifs RFID et NFC. Parmi ces applications utilisent des sites web réseautage social, tels que Twitter et Facebook. Ces applications mettent automatiquement à jour toutes informations qui concernent les activités sociales des personnes. Dans ce concept, divers objets peuvent tweeter périodiquement les lectures qui peuvent être rapidement suivies par les amis de n'importe où en temps réel [20].

1.3.6.5 Domaine de l'agriculture

Le domaine de l'agriculture est considéré comme un paradigme complexe de l'IoT, qui utilise les solutions et technologies de l'IoT pour permettre une gestion efficace des ressources. Il peut aider à surveiller le développement des plantes. Ces plantes sont équipées d'étiquettes RFID et de capteurs pour être contrôlées et gérées en cas de changement drastique ou inattendu dans l'évolution des plantes en raison de la température/humidité, en envoyant ces anomalies

au le lecteur pour être partagées sur internet. L'agriculteur ou le scientifique peut accéder à ces informations à distance et prendre les mesures nécessaires [25].

FIGURE 1.15 : IoT dans le domaine de l'agriculture [26].

Ainsi, cette agriculture a pour principaux objectifs :

- L'optimisation des apports aux cultures (semences, eau d'irrigation, fertilisants et produits phytosanitaires) dans le but d'améliorer le rendement.
- La lutte contre l'insuffisance et l'insécurité alimentaire surtout dans les pays en développement.
- La réduction des effets néfastes de l'agriculture sur l'environnement.

1.3.7 Architecture de l'Internet des objets

L'architecture d'un système IoT est composée de plusieurs niveaux qui communiquent entre eux pour relier le monde physique des objets au monde virtuel des réseaux. Tous les projets n'adoptent pas une architecture formellement identique, néanmoins il est possible de schématiser le parcours de la donnée. Il existe un modèle à 4 couches pour l'IoT [27, 28] comme le montre la « figure 1.16 » :

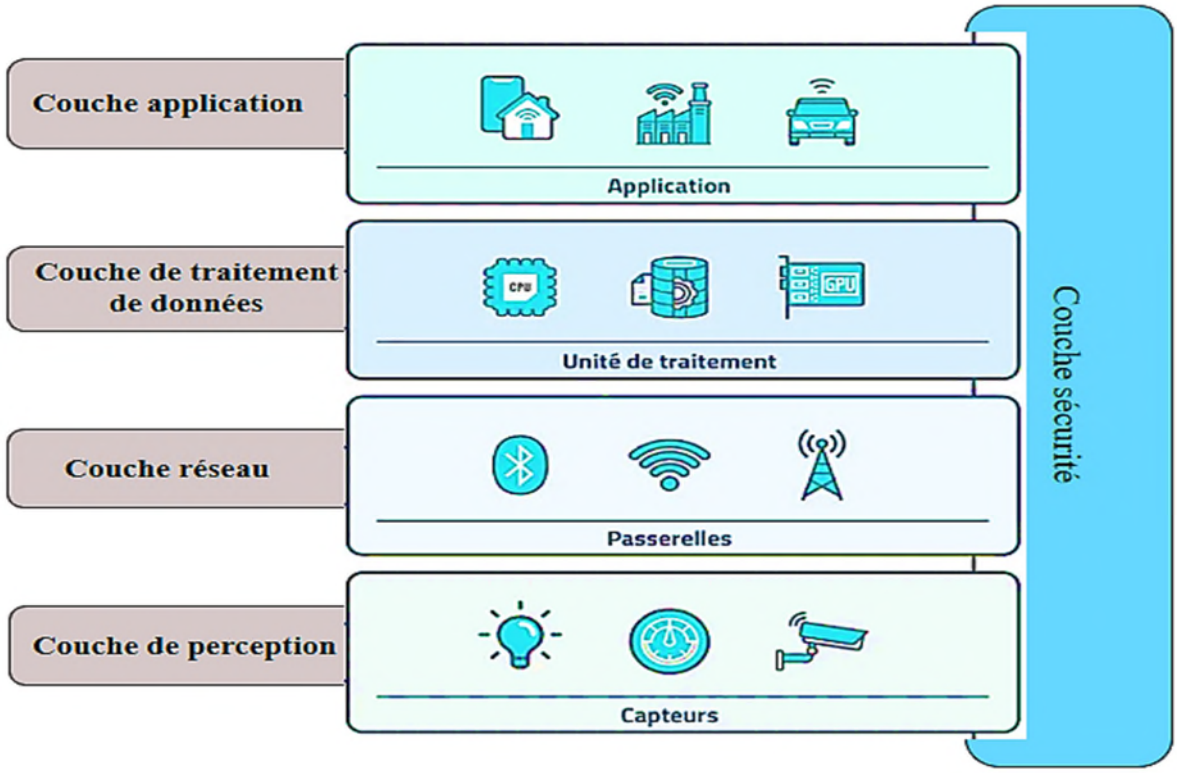


FIGURE 1.16 : Architecture de l'internet des objets [29].

1.3.7.1 Couche perception

Est une couche périphérique /objet. Elle possède des capteurs et actionneurs qui détectent et recueillent des informations sur l'environnement, ces informations collectées sont envoyées à la couche suivante, c'est-à-dire la couche réseau, pour un traitement ultérieur des données.

1.3.7.2 Couche réseau

Est également appelée « couche de transmission », responsable de la connexion, du transport et du traitement des données en toute sécurité issue des capteurs et actionneurs. Les communications entre les appareils et les services cloud ou les passerelles impliquent différentes technologies (Ethernet, Réseaux cellulaires, LPWAN, Wifi).

1.3.7.3 Couche traitement de données

Appelée également couche middleware. Il stocke, analyse et traite d'énormes quantités de données provenant de la couche transport. Il peut gérer et fournir un ensemble diversifié de services aux couches inférieures. Il utilise de nombreuses technologies telles que les bases de données, le cloud computing et les modules de traitement des méga données.

1.3.7.4 Couche application

Offre la gestion des applications basées sur les données traitées dans la couche middleware et est chargée de fournir à l'utilisateur des services spécifiques et applications intelligentes.

1.3.7.5 Couche sécurité

Cette couche est transverse à toutes les couches précédentes car la sécurité de l'IoT est primordiale.

1.3.8 Les Avantages de IoT

Avantages L'IoT devra faire partie de notre quotidien proche et sera appliqué dans divers domaines. Il présente de nombreux points positifs [30, 31] :

- Accès ubiquitaire à l'information pour un monde plus intelligent et un mode vie sophistiqué et confortable.

- Améliorer la productivité et l'expérience-client : les objets connectes envoient des rapports à leurs constructeurs indiquant les préférences et les habitudes des clients aidant davantage les entreprises à agir de manière proactive et adaptée qui satisfait la demande et les exigences de la clientèle.

- Automatisation : l'idée générale de l'IoT implique une communication directe entre des dispositifs, appareils et autres matériels distincts, sans intervention humaine.

- Dans certaines applications, l'IoT nous permet même de rationaliser nos dépenses et faire des économies car on ne consomme qu'en cas de besoin, que ça soit pour les achats ou la consommation énergétique ou autre.

- Connectivité : des connexions améliorées au sein d'un même réseau à l'échelle mondiale permettent d'accéder facilement à diverses informations.

- Télésurveillance : dans le domaine médical, les fabricants d'équipements médicaux équipent de plus en plus leurs appareils avec des capteurs IoT pour collecter les données des patients et permettre la surveillance de la santé et l'administration des médicaments à distance et même de réaliser une chirurgie à distance où le chirurgien et le patient se trouveraient à deux endroits différents.

- Possibilité d'exploitation des ressources géantes de l'Internet pour le stockage et le traitement des données écoulées de l'IoT.

- Gain du temps : l'IoT aide les gens à accomplir leurs tâches quotidiennes. Cela permet de gagner un temps précieux. Au lieu de faire des tâches monotones tous les jours, on peut les remplacer par une simple navigation sur le web pour commander des produits, contrôler l'état des objets et/ou endroits connectas.

1.3.9 Enjeux et les défis de l'Internet des objets

Bien que l'Internet des objets soit un concept qui est à la fois avantageux et prometteur, et qui pourra apporter des solutions efficaces des problèmes du suivi et de télésurveillance dans différents domaines. En contrepartie, l'IoT soulève certaines :

1. Sécurité : la sécurité des personnes, des communications, des données, des services, des réseaux et des équipements était et continue à être un problème sévère observé par l'internet courant. Il est donc important d'instaurer des politiques de sécurité claires, adaptatives selon le contexte d'utilisation et d'établir les responsabilités des faits et des actions sur l'environnement physique des objets [32].

2. Consommation d'énergie : l'énergie est considérée comme une ressource cruciale pour les appareils intelligents, car la plupart des applications sont alimentées par batterie ou utilise des techniques de récupération d'énergie. L'optimisation énergétique est donc l'aspect majeur de l'IoT [33].

3. Connectivité et réseaux : la connectivité est la caractéristique principale de l'IoT car elle permet au système d'envoyer des données et de rester connecté à d'autres appareils. Il fournit l'accessibilité au réseau du système et fonctionne en collaboration. Il s'agit de favoriser le développement de réseaux adaptés aux besoins de l'internet des objets, et leur large disponibilité, Parmi les défis auxquels nous sommes confrontés à notre époque liés à la connectivité et de réseau d'internet des objets sont :

- Interopérabilité : dans l'IoT, de nombreux objets intelligents sont connectés et chaque objet intelligent a sa propre capacité de collecte d'informations, de communication et de traitement. Pour la communication et la coopération entre les objets intelligents de différents types, il est nécessaire d'avoir une norme commune et d'assurer

l'interopérabilité de systèmes hétérogènes et répartis afin de permettre la mise à disposition et l'utilisation d'un large éventail d'informations et de services [34].

- Gestion des données : Un des défis critique de l'IoT est l'inter connectivites des objets intelligents et l'échange constant de tous les types d'informations, le volume des données générées et les processus impliqués dans le traitement de ces données. Donc, le stockage doit être attribué en fonction de la quantité de données [35].

- Scalabilité : En raison de connectivité spontanée de plusieurs objets ou appareils intelligents au réseau, l'Internet des objets devrait prendre en charge le nombre croissant d'appareils connectés, d'utilisateurs et d'analyses, et aussi être capable de résoudre les problèmes tels que l'adressage, la gestion de l'information et la gestion des services.

- Le déploiement du protocole IPv6 : Nous avons atteint le nombre maximal d'adresses IPv4. Le développement de l'IoT pourrait s'en trouver ralenti, puisque chacun de nouveaux capteurs potentiels devra avoir sa propre adresse IP. En outre, le protocole IPv6 facilite la gestion des réseaux grâce à des fonctions de configuration automatiques, et propose des fonctions de sécurité améliorées.

1.4. Conclusion

L'Internet des objets en tant qu'une évolution de l'Internet actuel permet une amélioration considérable de notre mode de vie et la façon dont les objets intelligents dans notre entourage interagissent entre eux et avec leurs utilisateurs.

Dans ce premier chapitre nous avons exposé l'Internet des objets d'une manière générale, nous avons cité brièvement les domaines d'application de l'internet des objets, ses technologies et son architecture. Ensuite nous avons cité les avantages et les défis. Parmi les défis majeurs de l'IoT c'est la connectivité qui est l'une des techniques assurent la sûreté de fonctionnement est le plus gros problème rencontré par la déployer d'IoT.

Dans le prochain chapitre, nous allons présenter les notions relatives aux solutions de la connectivité de IoT.

Chapitre 2 : La Connectivité

Chapitre 2 : La Connectivité

2.1. Introduction

L'IoT fait référence à la connectivité de divers appareils intelligents et d'objets physiques, qui peuvent communiquer entre eux pour fournir des données et des informations en temps réel. Les questions relatives à la connectivité font l'objet de nombreuses études techniques, notamment l'évaluation des technologies de réseau, l'essai des objets IoT et les études relatives à sa plateforme. Dans l'environnement connecté, il est très important de faire en sorte que tous les appareils se connectent en permanence avec les méthodes de connexion les plus efficaces. Par conséquent, les études sur la technologie de connectivité de l'IoT tendent à se concentrer sur la recherche de moyens d'interconnecter, n'importe quel produit du monde physique avec le monde virtuel par l'intermédiaire de n'importe quel réseau.

Dans ce chapitre nous allons donner une vue globale sur les réseaux de communications sans fil avec ses technologies. Par la suite nous allons présenter quelques protocoles les plus couramment utilisés pour alimenter les appareils et les applications de IoT et à la fin nous allons voir les défis liés à la connectivité avec ses solutions.

2.2. Communication dans les réseaux sans fil

Le réseau sans fil offre deux modes de fonctionnement : le mode infrastructure et le mode sans infrastructure (ad hoc).

2.2.1. Réseaux avec infrastructures

Les réseaux sans fil que nous utilisons aujourd'hui, tels que le GSM / GPRS / UMTS ++ ou le Wi-Fi, sont ce que l'on appelle des réseaux sans fil à stations de base. En mode avec infrastructure, également appelé le mode BSS (Basic Service Set) certains sites fixes, appelés stations support mobile (Mobile Support Station) ou station de base (SB) sont munis d'une interface de communication sans fil pour la communication directe avec des sites ou unités mobiles (UM), localisés dans une zone géographique limitée, appelée cellule. A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé [36].

1.3.10 Réseaux sans infrastructures

Le réseau mobile sans infrastructure également appelé réseau Ad hoc ou IBSS (Independent Basic Service Set) ne comporte pas l'entité « site fixe », tous les sites du réseau

se communiquent d'une manière directe en utilisant leurs interfaces de communication sans fil. L'absence de l'infrastructure ou du réseau filaire composé des stations de base, oblige les unités mobiles à se comporter comme des routeurs qui participent à la découverte et la maintenance des chemins pour acheminer les paquets vers les hôtes destinataires dans le réseau [36].

2.3. Les technologies de communication utilisée dans l'IoT

1.4.5 Technologies à courte portée

Sont très utilisés comme boucles locales par les objets connectés du grand public. Ils ont tendance à avoir besoin uniquement de petites batteries et leur fonctionnement est généralement peu onéreux. Le type de réseaux dits personnels WPAN (Wireless Personal Area Networks), et possède différents standards qui proposent, souvent seulement les couches physique et liaison de donnée. Quelque unes de ces technologies sont :

- **Bluetooth :** Très largement utilisé dans le monde. Quasiment tous les smartphones sont équipés de cette techno, fréquemment utilisée pour faire communiquer les wear ables. Il a une portée de 60 mètres en terrain dégagé et "consomme environ 20 fois moins d'énergie que le Wi-Fi". La dernière version de cette techno, le Bluetooth 5, est plus adaptée à l'IoT [37].

- **Z-Wave :** Protocole de communication dédié à la domotique. Sans fil, il est facile à installer dans la maison. Il a une portée de base de 30 mètres. C'est un réseau maillé, c'est-à-dire que chaque appareil connecté au système est émetteur de données, mais peut aussi relayer celles qui sont émises par ses voisins, cela permet d'élargir sa portée [37].

- **Wireless Hart :** Wireless Hart est une technologie open source dont la couche physique est basée sur le IEEE802.15.4, qui travaille à développer le protocole HART (Highway Addressable Remote Transducer Protocol). Celui-ci englobe, au-delà des couches physique et MAC (Media Access Control), les couches réseau, transport et application. Il ambitionne de garantir une robustesse face aux interfaces et un délai de bout-en-bout meilleur que Zigbee et Bluetooth. Ce qui fait de cette technologie une meilleure candidate pour les applications de mesure et de contrôle industriel [38].

- **ZigBee et ZigBee PRO :** Développés par ZigBee Alliance, ZigBee est destiné aux applications de domotique et ZigBee PRO pour les réseaux industriels de contrôle et de télémesure. Leur couche physique, basée aussi sur le standard IEEE 802.15.4, scanne et

sélectionne la fréquence qui a le moins d'interférence. Les couches supérieures (réseau et application) sont spécifiées et de la sécurité est introduite dans ce technologie [38].

1.4.6 Technologies à moyenne portée

- **Wi-fi :** Permet de transférer un grand nombre de données rapidement, jusqu'à 600 mégabits par seconde (mbps). Ce protocole de communication peut être utilisé pour connecter des caméras de surveillance qui filment des images lourdes 24 heures sur 24 heures. Il est bidirectionnel, ce qui permet de mettre facilement à jour les appareils, sauf qu'il est très énergivore et ne peut être utilisé que pour des appareils branchés au secteur, dans la maison par exemple et a une portée de 250 mètres en extérieur et de 35 mètres en intérieur [37].

- **ANT :** Conçu pour fonctionner dans la bande ISM de 2,4 GHz, ANT a une efficacité énergétique supérieure aux technologies 802.15.4 [Raw+14]. Elle supporte les topologies point-à-point, MeSH, étoile et arbre. Elle est utilisée pour les applications de santé et le débit peut atteindre 1 Mbit/s [38].

- **EnOcean :** EnOcean est une technologie qui émerge. Elle n'utilise pas de batterie comme les autres technologies et les capteurs utilisent l'énergie de leur environnement immédiat (en utilisant des panneaux solaires par exemple). EnOcean utilise les fréquences autour de 868 MHz et 315 MHz et a une portée de 300 mètres en communication libre [38].

1.4.7 Technologies à longue portée

Communément appelées LPWAN pour Low Power Wide Area Networks, ces technologies sont développées pour faciliter l'Internet des Objets en offrant des possibilités de communication à très basse consommation d'énergie et très longue distance. Parmi ces technologies IoT on distingue celles qui sont basées sur les réseaux mobiles cellulaires et celles qui ne le sont pas. Les technologies basées sur les réseaux mobiles telle que la 5G par exemple, offrent de large couverture, mais ont une consommation excessive d'énergie. Par contre les technologies telles que Sigfox, LoRa et NB-IoT sont les leaders de la longue distance avec une faible consommation d'énergie. Dans ce paragraphe, nous allons faire une étude comparative des technologies LPWAN.

- **Sigfox :** Développée en 2010 à Toulouse, Sigfox développe ses propres solutions IoT à travers 31 pays. C'est un réseau propriétaire, les stations de base sont équipées de la technologie SDR (Software-Defined Radio) qui les connecte au réseau par de connexion de

type IP. Les débits offerts atteignent quelques centaines de bits/s pour des distances allant jusqu'à 40 km [38].

- **NB-IoT :** NB-IoT (Narrow Band-Internet of Things) est une technologie LPWAN basée sur la radio bande étroite (narrow band) Un canal radio est dit bande étroite lorsque le message transmis dans la bande passante ne dépasse pas excessivement la bande de cohérence du canal. Elle a été développée par le groupe de travail 3GPP (3rd Generation Partnership Project) en 2016. Le protocole de communication de NB-IoT est basé sur LTE (Long Term Evolution). Les débits offerts sont de l'ordre de 200 kbits/s dans le sens descendant tandis qu'ils sont de seulement 20 kbits/s dans le sens ascendant [38].

- **LoRaWAN :** LoRaWAN est le protocole MAC de la technologie LoRa. Elle opère dans la bande ISM et est réputée être un standard ouvert, pourtant, la couche physique utilise une modulation à étalement de spectre qui est propriétaire au consortium LoRa Alliance. Les débits théoriques sont compris de 300 bits/s à 50 kbits/s et les distances en visibilité directe atteignent les 30 km [38].

- **LTE-M :** le standard LTE-M (Long Term Evolution, category M) extension des normes 4G monte progressivement en puissance avec la déclinaison de premières solutions opérationnelles. C'est une technologie mobile dédiée à l'IoT économiquement rationnelle et consomme moins d'énergie déployée par le biais d'équipements déjà en place et ne nécessite pas l'achat de nouveaux modems compatible comme pour le NB-IoT. La principale différence du LTE-M contrario du NB-IoT, c'est qu'il propose les échanges voix sur le réseau et surtout gère la mobilité des objets. Les appareils ne sont pas contraints de rester fixes mais ont la possibilité de se déplacer ce qui est indispensable pour les véhicules connectés ou applications pour travailleurs isolés.

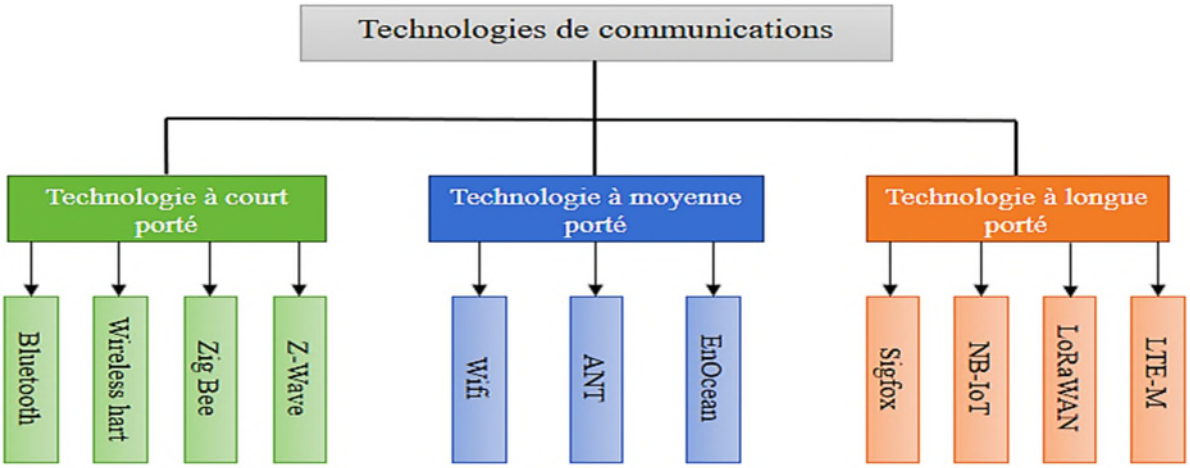


FIGURE 2.1 : Technologies de communications.

2.4. Les protocoles de IoT

2.4.1. Les protocoles IP

Dans l'internet d'aujourd'hui, la plupart des communications entre les nœuds de bout en bout utilisent le protocole IP. Ce protocole attribue une adresse unique à tous les nœuds connectés à l'internet et fournit les mécanismes de transport des données entre deux nœuds.

1.4.7.1 Internet Protocol version 4 (IPv4)

Est comme son nom l'indique un protocole qui identifie les adresses dites IP sur internet. Défini dès le début des années 1970 par les créateurs d'Internet. Il utilise un adressage 32 bits et permet d'obtenir 4 294 967 296 adresses uniques. La version 4 du protocole IP a été la première à être largement utilisée dans le TCP/IP moderne, il a prouvé sa qualité à l'usage sur une période de plus de 20 ans. A son origine, on pensait que ce serait amplement suffisant pour gérer tous les ordinateurs connectés sur internet mais depuis quelques années, un nouveau protocole IPv6 codé sur 128 bits est en cours d'élaboration car Ipv4 commence à montrer ses limites [39] :

- **Épuisement des adresses :** La principale limite d'IPv4 est l'épuisement des adresses IPv4 publiques disponibles, en particulier avec la croissance des services mobiles et domestiques qui nécessitent un grand nombre d'adresses IP publiques pour adresser tous les périphériques depuis l'extérieur.

- **Absence de type de données :** L'absence d'un type de données clairement défini dans IPv4 limite la capacité de gérer les priorités, ce qui affecte la qualité de service pour le trafic multimédia.

- **Configuration IP** : Actuelle d'IPv4 nécessite souvent une configuration manuelle ou l'utilisation de DHCP, mais avec la croissance des appareils utilisant des adresses IP, une configuration plus simple et automatisée est nécessaire, indépendante de l'infrastructure DHCP.

- **Faible niveau de sécurité :** IPv4 présente un faible niveau de sécurité, malgré l'existence de normes comme IPSec, qui sont facultatives et peuvent nécessiter des frais de licence supplémentaires pour être utilisées.

- **Explosion des tables de routage :** La demande croissante d'adresses IPv4 et d'accès à Internet entraîne une explosion des tables de routage, ce qui pose un défi majeur en raison de l'espace de stockage limité et de la combinaison d'informations de routage plates et hiérarchiques dans le réseau IPv4.

- **Fragmentation :** dans IPv4 permet de diviser les paquets de données pour les envoyer sur des réseaux de tailles différentes, mais cela peut entraîner des problèmes de performance et de sécurité lors de la reconstitution des paquets.

- **Qualité de service :** IPv4 ne prend pas en charge les mécanismes de qualité de service, ce qui signifie que tous les paquets sont traités de la même manière, quelle que soit leur importance ou leur priorité [40].

1.4.7.2 Internet Protocol version 6 (IPv6)

Les limites imposées par le protocole IPv4 se font sentir : le stock d'adresses IP autorisé par ce protocole n'est pas infini. Rapidement, un nouveau protocole l'IPv6 de substitution est défini par la communauté internationale afin de se substituer à l'IPv4 et de proposer un stock d'adresses. Les objectifs principaux de ce nouveau protocole sont [41] :

- L'IPv6 permet l'intégration de nouvelles fonctionnalités, telles qu'une sécurité améliorée, une qualité de service, alors que l'IPv4 ne fournit que le meilleur effort, et des mécanismes de mobilité.

- Facilitation de la qualité de service (QoS) : IPv6 facilite la mise en œuvre de la qualité de service (QoS) en permettant aux applications de marquer leurs paquets avec des priorités de trafic spécifiques.

- Simplification de l'en-tête : IPv6 a simplifié l'en-tête de paquet par rapport à IPv4. L'en-tête IPv6 est conçu pour être plus efficace à traiter par les routeurs et les autres équipements réseau.

- L'évolutivité offerte par IPv6 permettra donc d'interconnecter n'importe quel équipement et de concevoir de nouveaux services que nous ne pouvions pas trouvée par IPv4.

- Réduire la taille des tables de routage.

- Simplifier le protocole, pour permettre aux routeurs de router les datagrammes plus rapidement.

- Donner la possibilité à un ordinateur de se déplacer sans changer son adresse.

1.4.7.3 Intégration d'IPv6 dans l'internet des objets

L'intégration d'IPv6 dans l'Internet des objets fait référence à l'utilisation de la version 6 du protocole Internet (IPv6) pour connecter et gérer les appareils IoT. IPv6 a été développé pour remédier à l'épuisement des adresses IP dans la version précédente, IPv4, et offre un espace d'adressage beaucoup plus vaste. Cela permet de résoudre le problème de pénurie d'adresses IP dans un environnement où un grand nombre d'appareils IoT doivent être connectés. Avec IPv6, chaque appareil IoT peut être attribué une adresse IP unique, ce qui simplifie l'acheminement des données et facilite l'identification et la communication directe entre les appareils [42]. De plus, IPv6 offre des fonctionnalités avancées telles que la prise en charge native de la sécurité (IPsec) et la qualité de service (QoS), ce qui contribue à renforcer la fiabilité et la confidentialité des communications dans l'IoT [43].

L'IoT repose sur des réseaux personnels sans fil à faible puissance (LoWPAN), qui sont des dispositifs intelligents dont les performances sont faibles et limitées. À cette fin, l'IETF a défini l'IPv6 pour les réseaux LoWPAN afin d'étendre ces appareils intelligents à l'internet. Cette définition a donné naissance au protocole : 6LoWPAN, intégrant ainsi IPv6 dans les réseaux de capteurs sans fil (WSN). Pour l'IoT, les défis à relever pour offrir une communication IPv6 ont imposé aux WSN d'appliquer IPv6.

1.4.8 Les Protocoles de communication

Le protocole de communication dans l'Internet des objets est une couche de communication qui permet aux appareils connectés de communiquer entre eux et d'échanger des informations. Il s'agit d'un ensemble de règles et de normes qui définissent comment les dispositifs IoT interagissent et partagent des données. Ces protocoles sont conçus pour répondre aux défis spécifiques de l'IoT, tels que la consommation d'énergie réduite, la faible bande passante, les ressources limitées et la gestion des objets hétérogènes. Ils fournissent des fonctionnalités telles que la découverte de périphériques, l'échange de messages, la sécurité des données et la gestion de la connectivité [44]. La « figure 2.2 » illustre la pile du protocole IP Smart Object.

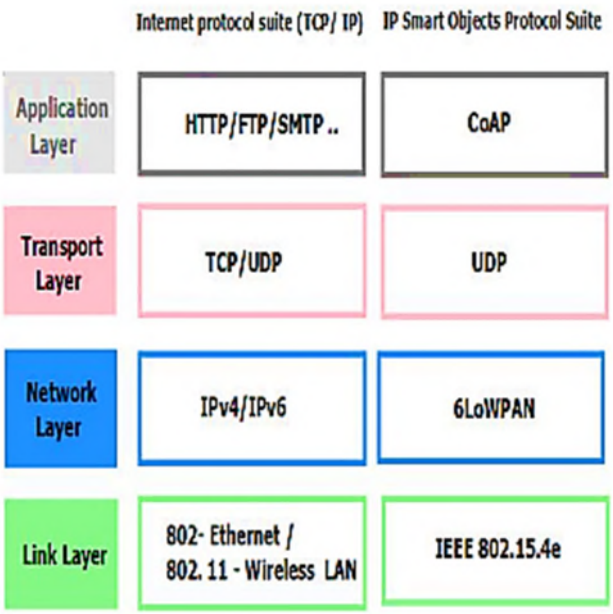


FIGURE 2.2 : Pile du protocole IP Smart Object [45].

2.4.2.1 Protocole d'application contraint (CoAP)

CoAP est un protocole de communication léger et basé sur le web, conçu spécifiquement pour les appareils IoT qui fonctionnent avec des ressources limitées en termes de puissance de calcul, de mémoire et de bande passante. Il permet aux dispositifs connectés d'échanger des informations de manière efficace et fiable, en utilisant le modèle client-serveur. CoAP utilise l'architecture REST (Representational State Transfer) pour l'accès aux ressources, et il est basé sur le protocole UDP (User Datagram Protocol) pour minimiser l'overhead. Il offre des fonctionnalités telles que la découverte automatique de ressources, l'observation de ressources pour détecter les changements, et la gestion des erreurs de manière légère. CoAP joue un rôle

essentiel dans la connectivité des appareils de l'IoT et favorise l'interopérabilité entre différents systèmes [46].

Le CoAP gère deux sous-couches : la sous-couche transaction et la sous-couche demande/réponse [48], comme l'illustre la « figure 2.3 ».

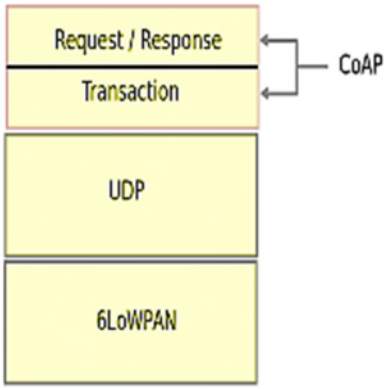


FIGURE 2.3 : Pile de protocoles CoAP [47].

La sous-couche transactionnelle traite le message unique transmis entre les nœuds communicants. Elle assure une communication fiable sur la couche de transport UDP (User Datagram Protocol) en utilisant le backoff exponentiel. Elle détecte également les duplications de messages.

La sous-couche demande/réponse gère les communications REST. La fiabilité de la CoAP est assurée par les messages confirmables et non confirmables.

2.4.2.2 User Datagram Protocol (UDP)

User Datagram Protocol (UDP) est un protocole de communication sans connexion et orienté vers les datagrammes. Il fait partie des protocoles de transport utilisés dans l'Internet des objets (IoT) pour permettre la communication entre les appareils connectés.

UDP est apprécié dans l'IoT pour sa simplicité et sa faible surcharge en comparaison avec le protocole TCP (Transmission Control Protocol), et pour cela il est utilisé dans l'IoT pour les applications où la latence est privilégiée et où la perte occasionnelle de paquets est acceptable. Il offre une alternative légère à TCP, tout en nécessitant des mesures de sécurité supplémentaires pour assurer la protection des données dans un environnement IoT [48].

2.4.1.3. 6LoWPAN

L'idée fondamentale derrière la couche d'adaptation de 6LoWPAN est d'utiliser la compression pour réduire la taille des en-têtes de la couche réseau et de la couche transport. Ainsi, l'ensemble des trois couches peut être compressé en seulement quelques octets. Cela permet d'économiser de la bande passante et de l'énergie, en particulier dans les réseaux IoT qui ont des ressources limitées. L'adoption d'IPv6 au-dessus d'un réseau personnel sans fil (WPAN) à faible puissance est inapplicable puisque les WPAN à faible consommation ont une faible bande passante, des paquets de petite taille et des dispositifs alimentés par batterie avec une capacité limitée. [47].

1.4.8.1 IEEE 802.15.4

Le groupe de travail IEEE 802.15 a créé la norme IEEE 802.15.4, qui spécifie la sous-couche pour le contrôle d'accès au support (MAC) et une couche physique (PHY) pour Réseaux personnels (PAN). Il met également l'accent sur une faible consommation d'énergie et un faible dispositif de coût. Il fournit une communication fiable avec un débit de messages élevé et a peut fonctionner sur différentes plates-formes. Il réalise une authentification de haut niveau, un cryptage et services de sécurité [49].

1.4.8.2 Message Queuing Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) est protocole de messagerie qui a été introduit par Andy StanfordClark en 1999 [50]. MQTT vise à connecter des dispositifs et des réseaux embarqués avec des applications et des intergiciels. L'opération de connexion utilise un mécanisme de routage (one-to-one, one-to-many, many-to-many) et fait de MQTT un protocole de connexion optimal pour l'IoT. MQTT utilise le modèle de publication/abonnement pour offrir une flexibilité de transition et une simplicité de mise en œuvre. Par conséquent, le protocole MQTT représente un protocole de messagerie idéal pour les communications IoT et M2M, d'assurer le routage de dispositifs de petite taille, bon marché, à faible mémoire dans des réseaux vulnérables et à faible bande passante.

2.5. La connectivité

2.5.1. Les défis liés à la connectivité massive de IoT :

Bien que les technologies de l'IoT sans fil existantes permis succès dans la prise en charge de diverses applications IoT, il reste des problèmes et des difficultés subsistent pour répondre aux besoins prévisibles des futures applications de l'IoT avec des centaines de milliards d'objets ou de choses à connecter. L'un des principaux défis consiste à est de prendre

en charge la connectivité massive des appareils IoT avec des charges utiles de transmission de petite taille et des caractéristiques sporadiques [51, 52]. En fait, les protocoles RA des technologies existantes sont principalement basés sur ALOHA ou CSMA/CA, ce qui est très susceptible de provoquer :

- Graves collisions d'accès.
- Une latence accrue.
- Une surcharge de signalisation élevée pour les dispositifs IoT.
- Les ressources sans fil allouées à la connectivité IoT sont limitées et inefficace pour la connectivité massive.

Pour résoudre ces problèmes, des efforts continus ont été déployés pour développer de nouvelles technologies qui permettent de remédier aux lacunes des technologies existantes tout en conservant leurs bonnes caractéristiques. Dans la « figure 2.4 », nous résumons les principales technologies.

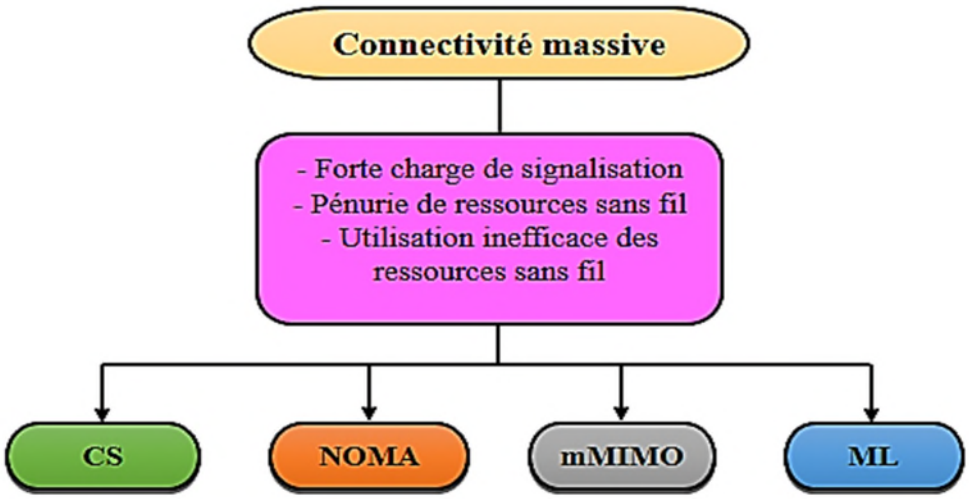


FIGURE 2.4 : Diagramme du défi à relever pour les technologies existantes et les solutions prometteuses.

1.4.9 Les solutions des problèmes de la connectivité dans IoT

Dans cette étude, il existe de quatre technologies prometteuses telles que CS (Compressive Sensing), NOMA (Non-Orthogonal Multiple Access), Mmimo (Multiple

Input Multiple Output) et ML (Machine learning) : qui peuvent résoudre efficacement la pénurie de ressources sans fil et améliorer l'efficacité de l'utilisation du spectre

1.4.9.1 Connectivité IoT basée sur le CS

En général, la RA sans subvention permet aux dispositifs IoT de se confronter directement à leurs charges utiles de liaison montante en transmettant un préambule avec les données. En utilisant la caractéristique naturelle des trafics sporadiques dans les MTC (Machine Type Communications), comme le montre la « figure 2.5 ». Des séquences éparses ont été utilisées à la place des séquences binaires pour l'étalement du signal de données afin d'augmenter le nombre de dispositifs MTC et de permettre l'identification des dispositifs [53].

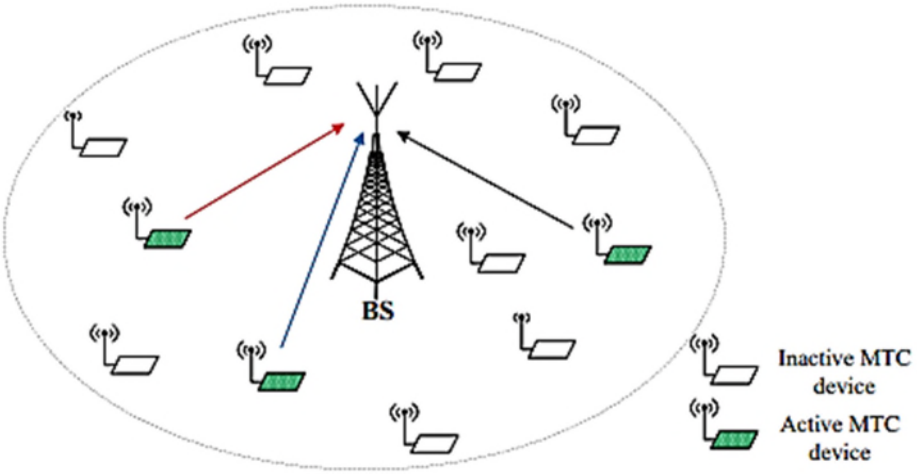


FIGURE 2.5: Illustration de l'activité éparse des utilisateurs dans un MTC massif [54].

1.4.9.2 Connectivité IoT basée sur NOMA

Le NOMA a récemment été identifié comme une technologie prometteuse pour une utilisation plus efficace des ressources sans fil. L'idée principale de la NOMA pour l'accès massif est de permettre le chevauchement des signaux sur la même ressource temps-fréquence via le multiplexage dans le domaine de la puissance (PDM) ou le multiplexage dans le domaine du code (CDM) et d'utiliser des interférences successives, de code (CDM) d'utiliser l'annulation successive des interférences (SIC) au niveau d'une station de base. La « figure 2.6 » illustre le principe de base de la NOMA dans le domaine de la puissance pour la transmission sur la liaison montante. Le principal avantage de la NOMA dans le domaine de la puissance pour le MTC est de permettre à plusieurs dispositifs d'effectuer simultanément un accès libre à la même ressource temps-fréquence sans étalement de la bande passante [55].

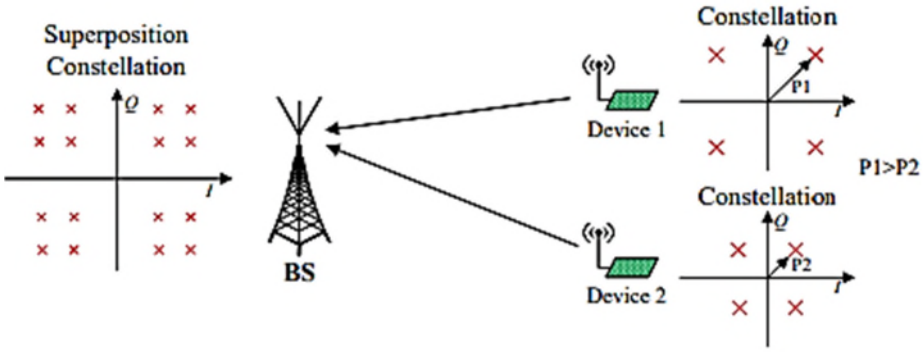


FIGURE 2.6 : Systèmes NOMA simplifiés dans le domaine de la puissance [54].

Bien que tous ces travaux indiquent que le NOMA est une technologie prometteuse pour permettre un accès massif sans concession pour les normes MTC émergentes, il reste des défis à relever pour permettre sa mise en œuvre.

1.4.9.3 Connectivité IoT basée sur mMIMO

Outre le NOMA, le mMIMO est une autre technologie prometteuse mMIMO exploite les ressources sans fil dans le domaine spatial, ce qui permet d'accueillir un grand nombre de dispositifs MTC, comme le montre la « figure 2.7 ». Dans un système mMIMO typique, un grand nombre d'antennes sont utilisées à la station de base. Grâce à cela, les réponses des canaux entre les différents dispositifs ont tendance à être orthogonales les unes par rapport aux autres. En tirant parti de cette propriété, un grand nombre de dispositifs dans la même ressource temps-fréquence peuvent être simultanément accommodés de manière efficace. Ces travaux valident l'efficacité du mMIMO dans la résolution des collisions, la réduction des délais et l'amélioration de la qualité de service d'accès.

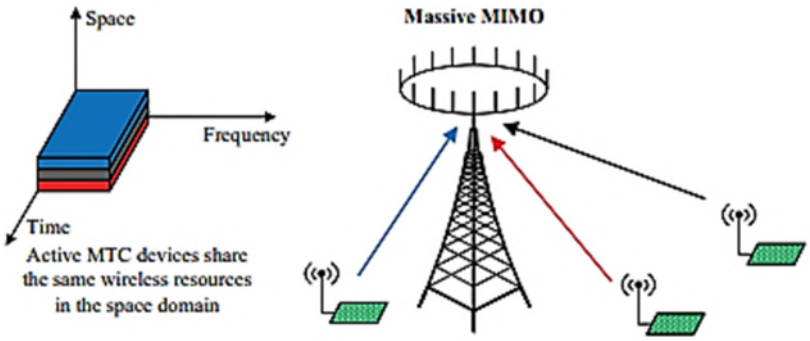


FIGURE 2.7 : Illustration des systèmes mMIMO [54].

Bien que tous ces travaux de recherche aient confirmé que le mMIMO est un outil viable et efficace pour les applications MTC émergentes dans l'IoT. D'un autre côté, étant donné que

le nombre de dispositifs MTC qui pourraient être pris en charge par mMIMO croît avec le nombre d'antennes, on s'attend à ce que des centaines ou des milliers d'antennes soient utilisées pour prendre en charge l'accès massif dans diverses applications IoT. Toutefois, compte tenu des dimensions du réseau et du coût du matériel, il pourrait être difficile de rassembler des antennes massives de manière centralisée. Par ailleurs, le mMIMO distribué pourrait être un candidat viable pour l'IoT du futur [56, 57].

1.4.9.4 Connectivité IoT assistée par l'apprentissage automatique

Récemment, les algorithmes ML ont attiré beaucoup d'attention pour résoudre divers problèmes dans les communications sans fil, notamment l'adaptation des liens, le contrôle du trafic et l'allocation des ressources.

La ML est un outil très puissant qui peut être utilisé pour améliorer l'utilisation inefficace des ressources sans fil dans l'IoT puisque les problèmes liés à l'optimisation de l'allocation des ressources sont généralement trop complexes pour être modélisés en raison des environnements sans fil dynamiques. Cependant, les modèles dynamiques de l'environnement sans fil pourraient être explorés efficacement par ML avec une complexité beaucoup plus faible que l'utilisation de technologies d'optimisation. C'est la raison pour laquelle plusieurs travaux (prédire la qualité de service (QoS) pour les applications de l'Internet des objets basées sur la mobilité des dispositifs, optimiser l'accès massif dans les réseaux IoT et la prédiction de la consommation d'énergie des dispositifs IoT…etc.) ont appliqué la ML pour relever les défis de l'accès massif pour les nouveaux réseaux mobiles. Défis de l'accès massif pour les applications MTC émergentes [58].

Toutes les technologies susmentionnées ont le potentiel d'être employées dans les futures normes de connectivité de l'IoT. Néanmoins, leur mise en œuvre soulève également des problèmes et des limites. Le « tableau 2.1 » met en évidence leurs points forts et leurs limites.

Technologies	Points forts	Limites
CS	Des schémas efficaces de détection de signaux multiples Peuvent être développés en exploitant l'activité éparse des dispositifs dans le MTC.	- Complexité élevée de l'accès massif avec un grand nombre de dispositifs MTC. - expansion de la bande passante de bande passante nécessaire pour augmenter le nombre de dispositifs MTC pouvant être pris en charge simultanément.
NOMA	Permettre le chevauchement des signaux sur la même ressource temps-fréquence par le biais de PDM ou CDM.	- Propagation des erreurs au stade de la SIC dans NOMA dans le domaine de la puissance. - compromis entre facteur de surcharge et la complexité du récepteur dans le domaine du code NOMA doit être optimisé.
Mmimo	Grâce à une propagation favorable, les ressources sans fil dans le domaine spatial peuvent être exploitées pour prendre en charge simultanément un grand nombre de dispositifs.	- Grand espace de préambule mais une faible corrélation mutuelle doit être conçu. - les dimensions du réseau et le coût du matériel doivent être matériel doivent être pris en compte lorsque le nombre d'antennes est important.
ML	Les modèles dynamiques de l'environnement sans fil qui sont trop complexes pour être modélisés pourraient être explorés efficacement.	- Le compromis entre les exigences de calcul des algorithmes et la précision des modèles appris doit être bien conçu. - il peut prendre du temps, ce qui peut ne pas être pas adapté aux environnements hautement dynamiques.

TABLEAU 2.1: Résumé des points forts et des limites des technologies prometteuses pour la connectivité massive [54].

En outre, toutes ces technologies émergentes peuvent non seulement être employées pour assurer une connectivité massive, mais aussi pour fournir des transmissions à haute fiabilité et à faible latence. À l'avenir, on s'attend à ce que des technologies de plus en plus avancées soient développées pour relever divers défis critiques de l'IoT. Sur le site Dans le même temps, des efforts doivent être faits pour fusionner intelligemment les technologies existantes et émergentes afin de réaliser leur plein potentiel et de maximiser les performances du système.

2.6. Conclusion

Enfin, il est important de noter que la technologie de l'IoT est encore en évolution et que de nouveaux développements continuent d'émerger, notamment dans le domaine de la connectivité. La connectivité est un élément essentiel de l'IoT, permettant aux dispositifs de collecter et d'échange des données de manière efficace. Dans ce chapitre nous avons détaillé en premier lieu les réseaux de communication sans-fil, en second lieu nous avons expliqué les protocoles de communication les plus utilisés dans le domaine de IoT et enfin nous avons illustré les défis liés à la connectivité et ses solutions.

Dans le chapitre suivant, nous allons présenter la technologie SDN, qui est l'une des solutions utilisées pour gérer la connectivité entre les différents objets connectés.

Chapitre 3 : SDN dans l'IoT

Chapitre 3 : SDN dans l'IoT

3.1. Introduction

Le Software Difined Networking (SDN) est considéré aujourd'hui comme une innovation majeure permettant de gérer plus efficacement les objets connectés et permet également une grande flexibilité et évolutivité, qui peuvent répondre aux besoins actuels de l'IoT en matière de connectivité et de sécurité. Dans ce chapitre, nous allons présenter l'entreprise SONATRACH et l'organisme d'accueil qui est une étape importante pour représenter les contraintes sous lesquelles se réalisera notre projet, par la suite nous allons détaille la solution SDN proposée et l'architecture IoT dans cette solution.

3.2. Présentation de l'organisme d'accueil

Afin de mettre en pratique les différentes connaissances que nous avons acquises en notre formation de l'administration et sécurité des réseaux et pour réaliser notre mémoire, nous avons obtenu un accord avec l'entreprise SONATRACH pour la réalisation d'un stage pratique d'une durée de 45 jour dont le but principal est l'adoption du solution SDN (Software-Defined Networking) pour optimiser son infrastructure.

3.2.1. Présentation de l'entreprise SONATRACH

SONATRACH, avant d'avoir ce nom, était la société pétrolière de gérance (SOPEG) fondée le 12 mars 1956 par la compagnie française des pétroles Algérie (CFPA) et la société nationale de recherche et exploitation des pétroles en Algérie (SNREPAL). Après l'indépendance, et grâce au décret n 36/491 de la nationalisation des hydrocarbures, la SOPEG est devenue sonatrach.

SONATRACH est un acronyme de « Société Nationale de Transport et de Commercialisation des Hydrocarbures », c'est une société Algérienne créée le 31/12/1963. Ses activités principales étaient le transport et la commercialisation des hydrocarbures, et à partir de 1966, son champ d'action s'élargit et englobe la recherche et la transformation des hydrocarbures. En 1967, singé inventa son logo de couleurs orange, rouge et noire.

3.2.2. Organigramme de l'entreprise

Pour la réalisation de ses objectifs, SONATRACH est divisée en cinq branches différents représentées par l'organigramme suivant :

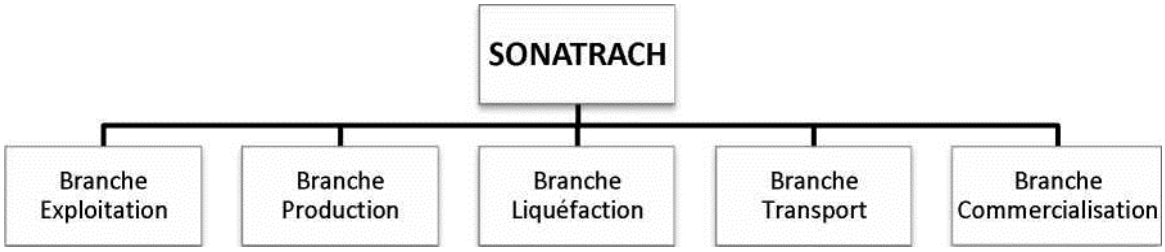


FIGURE 3.1 : Organigramme de Sonatrach.

2.3.7 Présentation de centre informatique

Le centre informatique est chargé du développement et de l'exploitation des applications informatiques de gestion pour le compte de la direction régionale de Bejaia (DRGP) et des autres régions. Le centre informatique s'organise en trois services tels qu'ils sont schématisés dans la figure suivante :

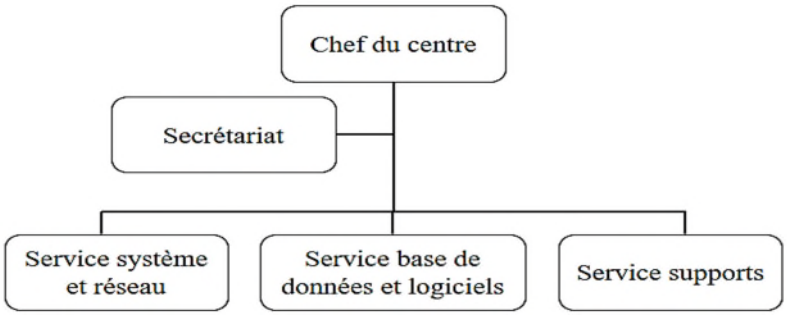


FIGURE 3.2 : Organigramme du centre informatique

Chaque service a sa propre fonction, nous allons définir et citer les différentes taches de chacune ci-dessous :

- **Service système et réseau**

Ce service est divisé en deux section

- ✓ **Système** : son rôle est :
  - Choix des équipements informatique et logiciel de base.

- Mise en œuvre les solutions matériels et logiciels retenues.
- Installation et configuration des systèmes.
- Orientation des travaux de l'équipe de développement par une abonne utilisation des ressources de l'ordinateur.
- Mise en œuvre des nouvelles versions de logiciels.

✓ **Réseau :** a pour rôle :

- Assure le bon fonctionnement, la fiabilité des communications, l'administration du réseau et organiser l'évolution de sa structure.
- Conduite de l'étude pour le choix de l'architecture du réseau à installer.
- Participer à la mise en place des réseaux.
- Définir les droits d'accès à l'utilisation du réseau.
- Assure le surveillance permanente pour détecter et prévenir les pannes.
- Traitement des dysfonctionnements et incident survenant sur le réseau.

➢ **Service base de données et logiciels**

✓ **Base de données :** son rôle est :

- Conçoit les bases de données et assure l'optimisation et le suivi de la gestion des données informatique.
- Installe, configure et exploite le SGBD et ses bases.
- Met en œuvre et gère les procédures de sécurité (accès, intégrité).
- Gère la sauvegarde, la restauration et la migration des données.
- Assure la cohérence et la qualité des données introduites par les utilisateurs.

✓ **Logiciels :** a pour rôle :

- Etude et conception des systèmes d'information.
- Développement et maintenance de l'application informatique pour TRC.

- **Service supports :** son rôle est :

  - Assistance aux utilisateurs en cas de problèmes software et hardware.
  - Installation des logiciels, technique et bureautique.
  - Formation aux nouveaux produits installés.

3.3. Cahier des charges

Un cahier des charges est un document qui doit être respecté lors de la réalisation d'un projet, cet outil est indispensable pour définir les spécifications d'un projet.

2.4.5 Présentation du sujet

Le projet que nous présentons dans ce mémoire traite le thème de « la solution SDN dans le réseau IoT » qui consiste à fournir une gestion centralisée et simplifiée de réseau d'entreprise.

2.4.6 Problématique

Lors de notre stage chez Sonatrach, nous avons identifié un enjeu majeur lié à leur dépendance à un ancien réseau qui présente de nombreux problèmes en termes de performance, de flexibilité et de gestion.

2.4.7 Objectifs de notre travail

Nous avons proposé une solution basée sur le SDN (Software-Defined Networking) qui permettrait de résoudre les problèmes de dépendance à l'ancien réseau de Sonatrach de manière efficace.

Objectifs visés par ce travail sont :

- Déploiement rapide et une configuration flexible des appareils connectés.
- Mettre en œuvre nos connaissances théoriques au sein d'une entreprise.
- Surveiller en temps réel les performances et détecter les problèmes potentiels de manière proactive**.**
- La sécurité des données IoT sera renforcée grâce à des politiques de contrôle d'accès et à une segmentation précise du réseau.
- Améliorer l'efficacité opérationnelle, la flexibilité et la connectivité de réseaux IoT dans l'entreprise.

3.4. Software Difined Networking SDN

2.5.5 Définition du SDN

Le SDN est un nouveau concept émergent de gestion des réseaux. C'est une nouvelle façon de concevoir, de construire, d'exploiter et de sécuriser les réseaux. Il est basé sur une gestion centralisée des flux réseaux par le découplage du plan de contrôle, responsable d'associer une décision de routage aux paquets du plan de données représentant l'infrastructure physique ou virtuelle et qui s'occupe uniquement de l'acheminement des paquets rendant les réseaux flexibles et programmables. SDN est une architecture dynamique, gérable, rentable et adaptable, ce qui la rend idéale pour la nature dynamique et les applications à large bande passante d'aujourd'hui.

Selon l'ONF (Open Network Fondation) SDN centralise toute l'intelligence de réseau dans une entité programmable appelé « Contrôleur », afin de gérer plusieurs éléments du plan de données (switches ou routeurs, etc.) via des APIs (Application Programming Interface) [59].

On peut dire qu'une architecture réseau suit le paradigme SDN si, et seulement si, elle vérifie les points suivants :

- Le plan de contrôle est complètement découplé du plan de données.
- Toute l'intelligence du réseau est externalisée dans un point logiquement centralisé appelé contrôleur SDN, ce dernier offre une vue globale sur toute l'infrastructure physique.
- Le contrôleur SDN est un composant programmable qui expose une API (NorthboundAPI) pour spécifier des applications de contrôle [60].

2.5.6 Historique de SDN

Le concept de SDN a été proposé en 2005 par des chercheurs de l'université de Stanford, notamment Nick McKeown et Martin Casado, dans le but de simplifier la gestion des réseaux informatiques complexes. Ils ont présenté une approche innovante qui consistait à séparer le plan de contrôle du plan de données dans les réseaux, permettant ainsi aux administrateurs de contrôler et de configurer les réseaux de manière centralisée, via un contrôleur SDN. Cette approche a également permis aux développeurs d'applications de programmer des réseaux via des API, facilitant ainsi l'innovation et la création de nouveaux services de réseau. En 2008, des membres de l'Open Networking Foundation (ONF) ont publié

un document fondateur sur SDN, qui définit une architecture de réseau basée sur la séparation du plan de contrôle et du plan de données. Cette publication a été le catalyseur de la formation de l'ONF, une organisation à but non lucratif créée en 2009 pour promouvoir et développer la technologie SDN, en rassemblant des acteurs clés de l'industrie des réseaux, tels que Cisco, Google, Microsoft et Hewlett-Packard. Depuis lors, SDN a connu une adoption croissante dans les réseaux informatiques, avec des applications dans divers secteurs, notamment les centres de données, les réseaux d'entreprise et les réseaux de télécommunications. En 2011, l'Open Networking Foundation (ONF) a été créée pour promouvoir et développer la technologie OpenFlow. Par la suite, d'autres protocoles de contrôle ont été développés, tels que NETCONF et OpenDaylight, et l'utilisation de l'architecture SDN s'est étendue à d'autres domaines, tels que les centres de données et les réseaux de télécommunications. Aujourd'hui, le SDN continue d'évoluer, offrant des avantages tels que la flexibilité, l'efficacité, la connectivité et la réduction des coûts pour la gestion des réseaux modernes [61,62].

2.5.7 Principe du SDN

Un élément clé du SDN est l'introduction d'une abstraction entre le plan de transmission (plan de données) et le plan de contrôle, qui sont traditionnellement liés, dont leurs rôles sont les suivants : le plan de transmission transfère le trafic en fonction des décisions prises par le plan de contrôle, qui a son tour, décide comment gérer le trafic du réseau. Cette abstraction permet de séparer et de fournir aux applications les moyens nécessaires pour contrôler le réseau, par programme ou par requête. L'objectif est de tirer avantage de cette séparation, afin de réduire la complexité et de permettre une innovation plus rapide sur les deux plans [63].

2.5.8 Architecture de SDN

Un réseau traditionnel est composé généralement des équipements d'interconnexions tels que des switches et des routeurs. Ces équipements incorporent à la fois la partie transmission et la partie de contrôle de réseau. Dans ce modèle d'architecture, il est difficile de développer de nouveaux services, en raison du fort couplage qui existe entre le plan de contrôle et le plan de transmission. La « figure 3.3 » donne un aperçu sur la différence entre l'architecture traditionnelle et l'architecture SDN.

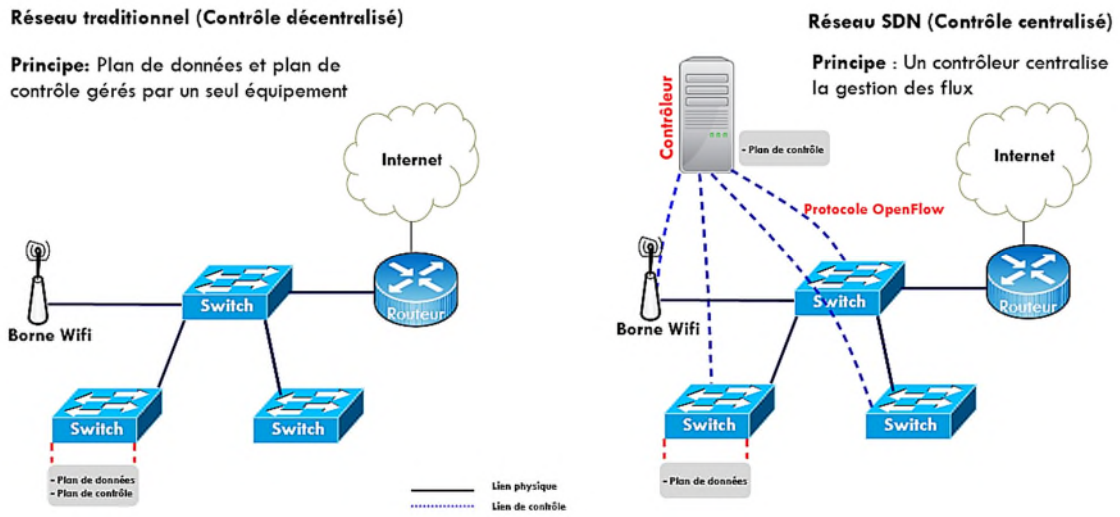


FIGURE 3.3 : Comparaison entre un réseau traditionnel et un réseau SDN [64]

L'architecture SDN permet de découpler la partie de contrôle de la partie transmission des équipements d'interconnexions. Elle est divisée en trois couches à savoir la couche de transmission, la couche de contrôle et la couche application, La communication entre ces différentes couches est faite à travers les interfaces de communication (APIs) comme l'indique la « figure 3.4 » [60] :

- **La couche de transmission :** appelée aussi « plan de données », elle est composée des équipements d'acheminement tels que les switches ou les routeurs, son rôle principal est de transmettre le trafic réseau sur une base d'un certain ensemble de règles de transmission ordonnée par le plan de contrôle. La communication entre le plan de données et le plan de contrôle est assurée par des interfaces Sud qu'on appelle API Southbond.Ces interfaces permettent au plan de contrôle d'instruire la couche de transmission sur la manière dont le trafic doit être acheminé en fonction des politiques définies.

- **La couche de 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'. La configuration du réseau, via le plan de contrôle, donne la possibilité de manipuler en temps réel les tableaux de flux de chaque élément du réseau, en fonction des performances du réseau et des exigences du service. Le contrôleur donne une vue claire et centralisée du réseau sous-jacent, ce qui donne un outil de gestion du réseau flexible et puissant, qui peut affiner les règles de performance.

- **La 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 construites moyennant une interface de programmation appelée 'North-bound API.

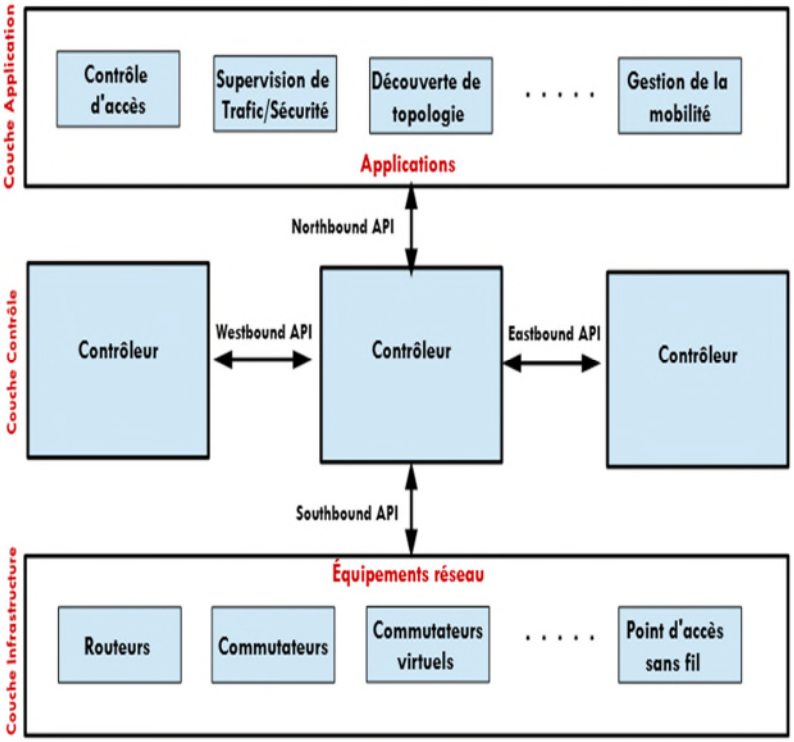


FIGURE 3.4 : Architecture d'un réseau SDN [60].

2.5.9 Les interfaces de programmation du SDN

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

- **Interfaces Sud (Southbond API) :** Les interfaces Sud constituent un protocole entre le contrôleur SDN et le plan de donnée. Elles contrôlent les opérations de transfert, les notifications d'événements, les rapports statistiques et annoncent également les capacités du réseau. Essentiellement, elles permettent à un contrôleur de définir le comportement du matériel dans le réseau. Des types d'interfaces Sud sont connus comme OpenFlow ou ForCES.

- **Interfaces Nord (Northbound API) :** Les interfaces Nord représentent une abstraction de fonctions réseau avec une interface programmable. En d'autres termes, elles permettent aux applications de consommer les services du réseau, et dynamiquement, changer leur comportement. L'architecture et les interfaces Nord des contrôleurs SDN varient selon les fournisseurs. Certains vendeurs incorporent des contrôleurs SDN dans leurs applications, tandis que d'autres définissent des interfaces Nord pour faciliter le protocole de communication entre les contrôleurs et leur propres services SDN au niveau de la couche application.
- **Interfaces Est /Ouest :** Les interfaces Est /Ouest sont des interfaces de communications qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleur pour synchroniser l'état de réseau. Ces architecteurs sont très récents et aucun standard de communication inter-contrôleur n'est actuellement disponible.

2.5.10 Les avantages de SDN

Le SDN offre de nombreux avantages par rapport aux réseaux traditionnels, notamment [65] :

- Réseau programmable : le contrôle de réseau est directement programmable grâce au découpage des fonctions de transfert, ainsi, il est simple de modifier les stratégies réseau car il suffit de changer une politique de haut niveau et non de multiples règles dans divers équipements de réseau.

- Management centralisé : l'intelligence du réseau est centralisée dans le contrôleur SDN, qui maintient une vue globale du réseau.

- Infrastructure réseau personnalisable : avec un réseau software-defined, les administrateurs peuvent configurer les services réseau et allouer des ressources virtuelles pour modifier l'infrastructure 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.

- Flexibilité : SDN apporte également une grande flexibilité dans la gestion du réseau. Il devient facile de rediriger le trafic, d'inspecter des flux inattendus.

- Simplification matérielle : la puissance du calcul n'est requise qu'au niveau du contrôleur. Ainsi, les équipements de réseau deviendront des produits à bas prix offrant des interfaces standard. Il serait également simple d'ajouter de nouveaux périphériques.

- 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 cloud tels que l'évolutivité, l'adaptation, ou des mouvements de machines virtuelles.

2.5.11 Les protocoles SDN

2.5.11.1 OpenFlow

OpenFlow est le protocole utilisé pour la communication entre la couche transmission et la couche contrôle, il a été initialement proposé et implémenté par l'université de Stanford, et défini par l'ONF comme étant un protocole de communication qui permet de contrôler le comportement du plan de données d'une façon centralisée, dynamique et programmable.

OpenFlow sert de lien entre le plan de contrôle et le plan de données, sa dernière version est 1.5 [66].

- **Architecture OpenFlow :**

L'architecture OpenFlow est l'implémentation réelle des réseaux SDN, Cette architecture est basée principalement sur trois composantes : le plan de données, qui est composée des switches OpenFlow, le plan de contrôle, constitué par des contrôleurs OpenFlow et une chaîne sécurisée qui permettent aux commutateurs de se connecter au plan de contrôle. La spécification d'un commutateur OpenFlow est standardisée par l'ONF. Selon la spécification d'ONF, un commutateur OpenFlow doit contenir un ou plusieurs tables de flux, ces tables de flux contiennent plusieurs d'entrées qui correspondent à des règles, où chacune est constituée principalement des trois champs suivants « Tableau 3.1 » [60] :

- **L'En-tête de paquet :** il définit le flux de données, il contient les informations nécessaires pour déterminer le paquet auquel cette règle sera appliquée. L'en-tête de paquet peut identifier différents protocoles tel qu'Ethernet, IPv4, IPv6 ou MPLS, cela dépend de la spécification d'OpenFlow déployée.

- **L'Action :** spécifie comment les paquets d'un flux seront traités. Une action peut être l'une des suivantes : transférer le paquet vers un ou plusieurs ports, supprimer le paquet, transférer le paquet vers le contrôleur, ou modifier le champ d'entête de paquet.

- **Les Compteurs :** sont réservés à la collecte des statistiques de flux. Ils enregistrent le nombre de paquets et d'octets reçus de chaque flux, et le temps écoulé depuis le dernier transfert de flux.

Champs d'en-tête	Compteurs	Actions

TABLEAU 3.1 : structure d'une entrée de table de flux d'un commutateur openflow 1.0[60].

La figure ci-dessous montre le processus de communication entre les switches OpenFlow et le contrôleur dans ce cas nous avons pris le cas d'un seul switch.

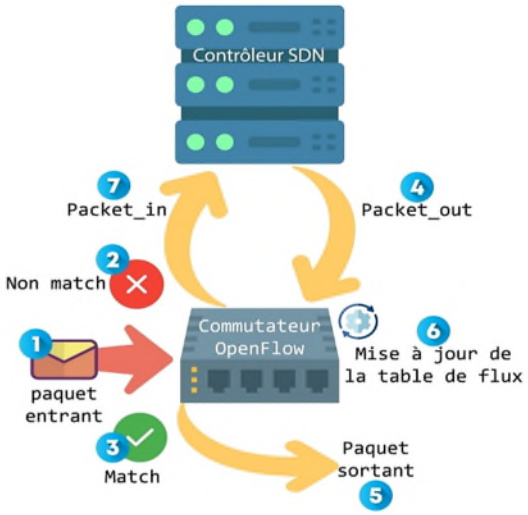


FIGURE 3.5: processus de communication entre le switch et le contrôleur [67].

Quand un paquet est reçu par le commutateur (1), son champ d'entête 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 l'action 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 à effectuer soit le paquet est détruit 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 d'un Packet_out et Flow-mod au commutateur (4). Enfin, la table de flux du commutateur est actualisée (6).

- **Fonctionnalités des différentes spécifications d'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.

Version / **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
Contrôleur multiples	Non	Non	Oui	Oui

TABLEU 3.2 : Fonctionnalités des différentes spécifications d'OpenFlow.

- **Les messages OpenFlow :**

Les messages OpenFlow entre le contrôleur et le commutateur sont transmis via un canal sécurisé, implanté via une connexion TLS sur TCP. Le commutateur initie la connexion TLS lorsqu'il connait l'adresse IP de contrôleur. Le protocole OpenFlow supporte trois types de messages (Messages symétriques, Messages asynchrones et Messages contrôleur-commutateur) [68].

- **Messages symétriques :** Ce sont des messages bidirectionnels qui sont envoyés par le contrôleur ou le commutateur entre eux sans aucune sollicitation. Les principaux messages sont décrits ci-dessous :

a. **HELLO :** échangés entre le contrôleur et le commutateur une fois le canal sécurisé a été établi.

b. **ECHO :** utilisés par les deux entités pour s'assurer que la connexion est toujours en vie et afin de mesurer la latence et le débit courant de la connexion. Chaque message ECHO-REQUEST doit être acquitté par un message ECHO-REPLAY.

- **Messages asynchrones :** Ces messages sont initiés par le commutateur et utilisés pour informer le contrôleur des évènements du réseau et des changements de l'état du commutateur. Ils sont utilisés pour informer le contrôleur des arrivées de paquets, des changements d'état au commutateur et des erreurs. Les quatre principaux messages asynchrones sont décrits ci-dessous :

  a. **PACKET-IN :** utilisé par le commutateur pour passer les paquets de données au contrôleur pour leur prise en charge (lorsqu'aucune entrée de flux ne correspond au paquet entrant ou lorsque l'action de l'entrée correspondante spécifie que le paquet doit être relayé au contrôleur). Si le commutateur dispose d'une mémoire suffisante pour mémoriser les paquets qui sont envoyés au contrôleur, les messages PACKET-IN contiennent une partie de l'en-tête (par défaut 128 octets), les commutateurs qui ne supportant pas de mémorisation interne émettent le paquet entier au contrôleur dans le message PACKET-IN.

  b. **FLOW-REMOVED :** le commutateur peut informer le contrôleur qu'une entrée de flux a été supprimée de la table de flux via le message FLOW-REMOVED.

  c. **PORT- STATUS :** utilisé afin de communiquer un changement d'état d'un port (le lien est hors service).

  d. **ERREUR :** ce message est utilisé pour notifier les erreurs au contrôleur.

- **Messages contrôleur-commutateur :** les messages contrôleur-commutateur sont initiés par le contrôleur et utilisés pour gérer et examiner l'état du commutateur OpenFlow. Les principaux messages sont décrits ci-dessous :

  a. **FEATURES :** le contrôleur peut demander l'identité et les capacités du commutateur en envoyant un message FEATURES-REQUEST. Le commutateur

doit répondre par FEATURE-REPLAY. Ceci est généralement effectué lors de l'établissement du canal sécurisé OpenFlow.

b. **CONFIGURATION** : le contrôleur émet le message unidirectionnel SET-CONFIG afin de positionner les paramètres de configuration du commutateur. La paire de message GET-CONFIG-REQUEST et GET-CONFIG-REPLAY est utilisé afin d'obtenir la configuration du commutateur.

c. **MODIFY-STATE :** ces messages sont envoyés par le contrôleur pour gérer l'état du commutateur, leur objectif est d'ajouter, supprimer ou modifier les entrées des tables de flux et de positionner les propriétés des ports du commutateur.

d. **PACKET-OUT :** utilisé par le contrôleur afin d'émettre des paquets de données au commutateur pour leur acheminement.

e. **BARRIER :** le messages BARRIER-REQUEST est utilisé par le contrôleur pour s'assurer que tous les messages OpenFlow émis par le contrôleur et qui ont précédé cette requête ont été reçus et traités par le commutateur. La confirmation est retournée par le commutateur via la réponse BARRIER-REPLAY.

2.5.11.2 ForCES

ForCES est un protocole standard définit par Internet Engineering Task Force (IETF) depuis 2004 et qui propose de séparer le contrôle IP de l'acheminement des paquets dans les équipements réseaux. Dans cette approche, les dispositifs d'acheminement des paquets sont modélisés à l'aide de blocs de fonctions logiques appelés Logical Fonction Blocks (LFB), pouvant être composés de manière modulaire pour former des mécanismes de transmission complexes. Chaque LFB fournit une fonctionnalité telle que le routage IP. Les LFB modélisent un dispositif de transfert et coopèrent pour former encore plus de périphériques réseaux complexes. Les éléments de contrôle utilisent le protocole ForCES pour configurer les LFB interconnectés, pour modifier le comportement des dispositifs d'acheminement des paquets. ForCES n'est pas adopté par les acteurs des réseaux en raison du manque de langage clair de communication entre le contrôleur et l'équipement réseau [69].

2.5.11.3 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 défini 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.

2.5.12 Les Contrôleurs SDN

Plusieurs contrôleurs ont été développés, dont la majorité sont open source et supportent le protocole openflow. Le « tableau 3.3 » présente les contrôleurs SDN les plus connus [60] :

Contrôleur	Organisation	Langage	Fonctionnalités
NOX	Nicira	C++	Le premier contrôleur OpenFlow.
POX	Nicira	Python	Amélioré les performances de NOX.
Ryu	NTT, OSRG groupe	Python	Support l'OpenStack.
Beacon	Standford	Java	Basé sur le multithreading.
Floodlight	Big Switch	Java	Testé avec des commutateurs OpenFlow physique et virtuels.
Opendaylight	Linux Foundation	Java	Support le Framework OSGi et le REST API

TABLEAU 3.3 : Contrôleurs SDN [60].

3.5. 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 contrôler et sécuriser une énorme quantité de données et d'informations qui nécessite beaucoup d'efforts en matière de traitement pour les transformer en informations utiles. Ces mégas donnés sont considérés 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) [70]. Pour les solutions SDN, l'architecture IoT la plus couramment utilisé d'après [71], est comme le montre la figure 3.6 :

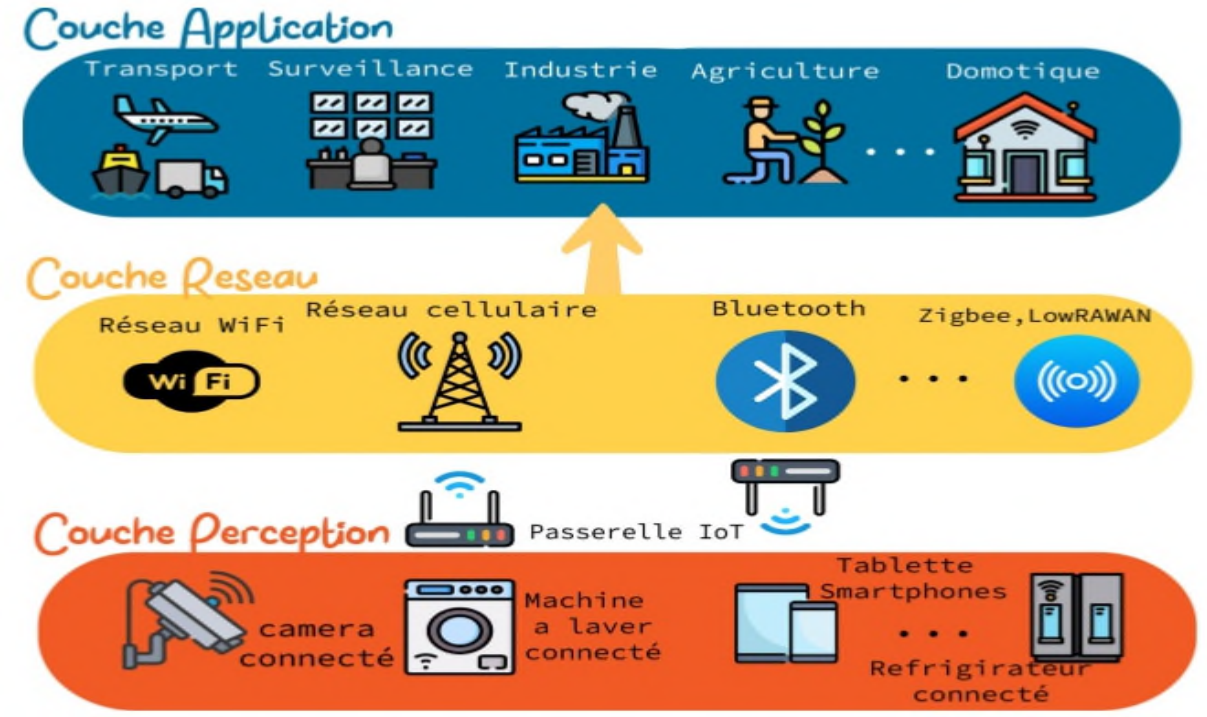


FIGURE 3.6 : Architecture IoT pour les solutions SDN [67].

✓ **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. Donc, 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.

- ✓ **La couche Réseau**

La couche réseau est responsable de la transmission des données vers la couche perception et vers la couche application. Pour assurer le transfert de données, la couche réseau utilise différents types de technologies de mise en réseau et de protocoles. Ces solutions de connexions réseaux sont, d'une part, les technologies déjà largement déployées, capables de transporter de gros volumes d'informations, mais gourmandes en énergie, telles que Bluetooth, WiFi, 3G, 4G et LTE, et d'autre part, les technologies connes¸ Cues spécifiquement pour le monde de l'IoT mais encore moins déployées comme les ZigBee, LoRa, Sigfox etc. La couche réseau est responsable de la connexion des objets intelligents, des périphériques réseau et des serveurs. Elle est également utilisée pour la distribution et l'analyse des données des capteurs.

- ✓ **La couche Application**

La couche application permet l'interaction direct avec les utilisateurs finaux. Les applications peuvent être déployées pour différents domaines de la vie quotidienne tels que la surveillance sanitaire, l'agriculture intelligents, le transport intelligent etc. elle comprend également des infrastructures serveur et cloud partageant du contenu et fournissant des services en temps réel. Le traitement des données et la fourniture de services sont deux des fonctions les plus importantes de cette couche. La couche application fournit une gestion globale du système IoT. Il reçoit des informations de la couche réseau qui permet aux développeurs de créer diverses applications en utilisant une abstraction et des interfaces ouvertes et de haut niveau. Et c'est dans cette couche que toutes les décisions de contrôle, de sécurité et de gestion des applications sont prises.

3.6. Comparaison entre l'architecture de l'IoT et l'architecture IoT dans le SDN :

La différence clé entre l'architecture traditionnelle de l'IoT et l'architecture IoT pour les solutions SDN réside dans leur objectif principal. L'architecture de l'IoT se concentre sur la collecte et le traitement des données, tandis que l'architecture IoT pour les solutions SDN se concentre sur la gestion et l'optimisation du réseau en utilisant les principes de SDN. Cette distinction est importante car elle peut aider à déterminer l'architecture la mieux adaptée à un projet IoT spécifique, Le « tableau 3.4» résume les différences entre eux [72, 73] :

Caractéristiques	Architecture traditionnelle de IoT	Architecture IoT pour les solutions SDN
Définition	Une architecture qui permet aux objets connectés de communiquer entre eux et avec le cloud.	Architecture réseau qui utilise les technologies SDN pour faciliter la gestion et le contrôle des dispositifs IoT connectés.
Fonctionnement	Communication sans fil entre les dispositifs IoT.	Utilisation d'un contrôleur SDN pour gérer la connectivité entre les dispositifs IoT.
Caractéristiques principales	Faible coût, grande flexibilité et évolutivité.	Gestion centralisée des dispositifs IoT, facilité de contrôle et évolutivité.
Technologies de communication	Bluetooth, Wi-Fi, Zigbee, LoRaWAN.	Les protocoles SDN (OpenFlow, ForCES, OpFlex, etc.)
Sécurité	La sécurité est un défi majeur dans l'architecture traditionnel de l'IoT en raison du grand nombre d'appareils et de nœuds connectés.	Offre une sécurité améliorée grâce à la technologie SDN, qui permet une segmentation efficace du réseau.
Gestion du trafic	Mise en œuvre pour gérer efficacement le trafic en utilisant des protocoles de communication appropriés.	Utilise les protocoles SDN pour gérer le trafic de manière efficace et flexible.
Fiabilité	L'architecture de l'IoT peut être affectée par des problèmes tels que les interférences radio, la distance de communication, etc.	L'architecture IoT pour les solutions SDN peut être conçue pour offrir une meilleure fiabilité en utilisant la technologie SDN pour optimiser la communication entre les appareils et les nœuds.

TABLEAU 3.4 : Comparaison entre l'architecture traditionnel de l'IoT et l'architecture IoT pour les solutions SDN.

3.7. Inconvénients de SDN

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

1. Besoin de formation du personnel : L'adoption de l'approche SDN peut nécessiter une formation approfondie du personnel existant ou même le recrutement de nouveaux employés afin de se familiariser pleinement avec son utilisation.

2. Réorganisation des échanges d'informations : L'échange d'informations entre les sociétés d'applications, les serveurs et les équipes réseau doit être fondamentalement réorganisé pour s'aligner avec l'approche SDN. Cela implique des modifications de l'infrastructure réseau existante pour mettre en œuvre les protocoles et les contrôleurs SDN, ce qui entraîne des reconfigurations complètes du réseau et des coûts supplémentaires [74].

3.8. Conclusion

La solution SDN offre des avantages significatifs pour l'IoT, en permettant une gestion plus efficace et sécurisée des réseaux d'objets connectés, une bonne connectivité et une meilleure gestion de la bande passante. Ce chapitre a introduit le concept de SDN, en expliquant son architecture et les protocoles utilisées. Enfin nous avons illustré l'architecture IoT dans le SDN.

Dans le chapitre suivant, nous allons simulée le réseau SDN avec les équipements IoT.

Chapitre 4 : Mise en place d'une la solution et Simulation

Chapitre 4 : Mise en place d'une la solution et Simulation

4.1. Introduction

Après avoir introduit les réseaux définit par logiciel, l'Internet des objets, puis 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. Par la suite, nous allons montrer quelques captures d'écran de notre application.

4.2. Présentation de l'environnement de travail

3.3.5 Open vSwitch (OVS)

Open vSwitch est un commutateur logiciel multicouche open source qui a été conçu pour gérer des environnements virtuels à grand échelle. Il fonctionne comme un commutateur virtuel assurant la connectivité entre les machines virtuelles et les interfaces physiques. Son objectif est de fournir une fonction de routage niveau 2 et 3 du modèle OSI pour les environnements virtualisés supportant différents protocoles et standards. L'OVS supporte le protocole OpenFlow dans un environnement virtuel basé sur linux. En plus d'OpenFlow, il prend en charge de nombreuses autres techniques de commutation traditionnelles. Il est constitué d'un service (**ovs-vswitchd**) et d'un module kernel (**openvswitch_mod**). Le service permet de capturer le trafic provenant des interfaces réseau [75].

- **Composantes d'Open vSwitch**

Comme le montre la « figure 4.1 », OVS est principalement composé des éléments suivants [76] :

- 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 cœur du système d'exploitation et la partie qui commute les paquets.

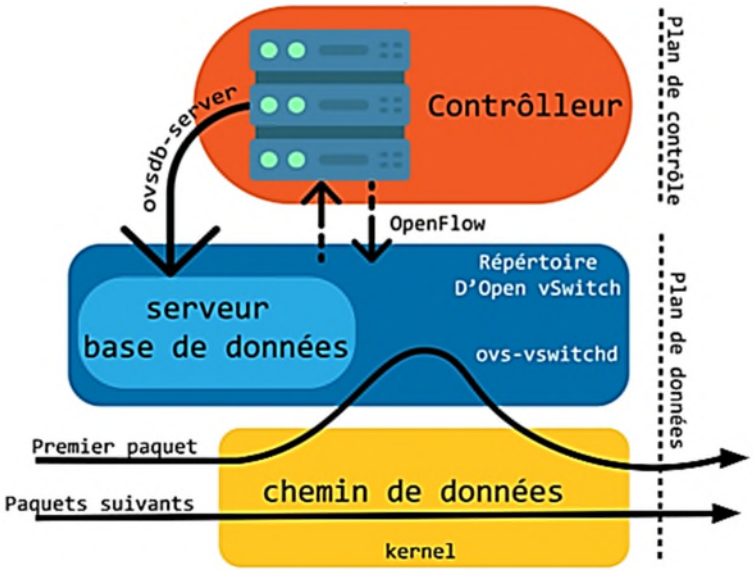


FIGURE 4.1 : Les composants et les interfaces d'Open vSwitch [71].

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

- **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 [76]. 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'OpenvSwitch utilisé dans une architecture OpenFlow [77].

— 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 estovs-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

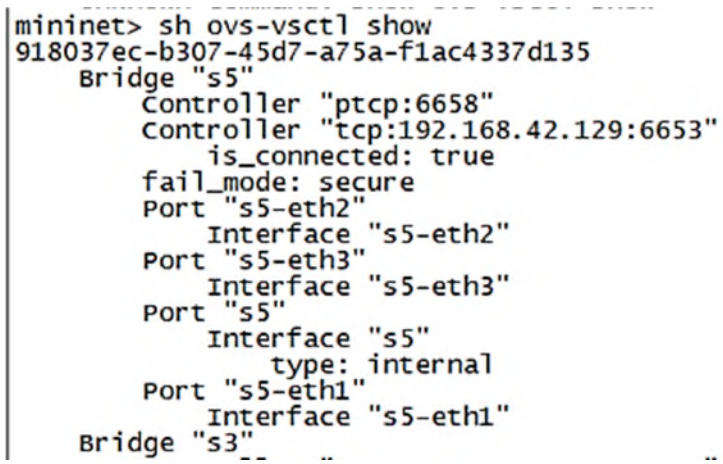


FIGURE 4.2 : Affichage de la commande show Dovs.

Pour l'architecture SDN, nous devons ensuite connecter le switch à un contrôleur, pour ce faire on introduit ovs-vsctl set-controller br0 tcp :192.168.42.129 :6633 ou « 192.168.42.129 » est l'adresse IP du contrôleur et 6633 le port d'écoute du protocole OpenFlow. A partir de là pour communiquer avec OpenFlow l'outil privilégié estovs-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 l'exemple présent on ajoute un simple fluxovs-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.

3.3.6 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. Il 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 [78].

- **Choix de Mininet**

Mininet fournit un outil simple pour reproduire le comportement d'un système réel et tester avec différentes topologies. Il permet :

- Un prototypage rapide pour le SDN
- Tester des topologies complexes sans qu'il soit nécessaire de mettre en place un réseau physique.
- Sa manipulation se fait via :
- Ligne de commande
- Interface interactive
- Scripts python

Pour mettre en place des topologies SDN, nous avons choisi d'utiliser une machine virtuelle que nous avons téléchargée. Cette machine virtuelle utilise le système d'exploitation linux et est accessible via n'importe quel environnement de virtualisation tels que : VMwar ou Virtualbox.

- **Fonctionnement de Mininet :**

Mininet permet avec un simple jeu de commande de réaliser des réseaux virtuels en utilisant la commande mn. Cette commande permet de créer deux hôtes et un switch. En plus de cette topologie, mininet permet de créer les topologies tree (Arbre), linear (linéaire) et single (étoile), il suffit d'utiliser l'option –topo de la commande mn. Par exemple : $ sudo mn –topo single. Pour connecter notre topologie à un contrôleur l'option adéquate est --controller. On peut également ajouter un chiffre come argument indiquant le nombre d'hôtes à créer : $ sudo mn –topo single, 4.

Si nous souhaitons personnaliser une topologie déjà créée, nous appliquons les commandes suivantes :

- Ajouter un switch: self.addSwitch('s1')
- Ajouter un hôte : self.addHost('h1')
- Ajouter un lien : self.addLink('h1,s1')

- **Créer une topologie avec Mininet**

Pour créer une topologie personnalisée différents de celles proposée par mininet, on utilise des scripts Python. Le fichier python (test4.py) doit être sauvegardé dans le répertoire mininet/custom de la machine virtuelle Mininet.

```
GNU nano 2.2.6                                          File: test4.py

from mininet.topo import Topo

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

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

        # Add hosts and switches
        h1 = self .addHost( 'h1', ip='10.0.10.1/24' )
        h2 = self .addHost( 'h2', ip='10.0.10.2/24' )
        h3 = self .addHost( 'h3', ip='10.0.11.1/24' )
        h4 = self .addHost( 'h4', ip='10.0.11.2/24' )
        h5 = self .addHost( 'h5', ip='10.0.12.1/24' )
        h6 = self .addHost( 'h6', ip='10.0.12.2/24' )
        h7 = self .addHost( 'h7', ip='10.0.10.3/24' )
        h8 = self .addHost( 'h8', ip='10.0.10.4/24' )
        s1 = self .addSwitch( 's1' )
        s2 = self .addSwitch( 's2' )
        s3 = self .addSwitch( 's3' )
        s4 = self .addSwitch( 's4' )
        s5 = self .addSwitch( 's5' )
        s6 = self .addSwitch( 's6' )
        s7 = self .addSwitch( 's7' )
        # Add links
        self.addLink( s4, h1 )
        self.addLink( s4, h2 )
        self.addLink( s4, s2 )
        self.addLink( s5, h3 )
        self.addLink( s5, h4 )
        self.addLink( s5, s2 )
        self.addLink( s6, h5 )
        self.addLink( s6, h6 )
        self.addLink( s6, s3 )
        self.addLink( s7, h7 )
        self.addLink( s7, h8 )
        self.addLink( s7, s3 )
        self.addLink( s2, s1 )
        self.addLink( s3, s1 )
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

FIGURE 4.3: Fichier test4.py.

Après avoir créé la topologie, nous devons tester si les paquets sont routés correctement avec la commande ping qui permet de vérifier la connectivité : Mininet> pingall.

- Pour accéder à toutes les commandes disponibles sous Mininet, il suffit de taper la commande : help.

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mininet>
```

FIGURE 4.4 : Affichage de la commande show help.

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

sudo mn --mac --custom test4.py --topo mytopo –switch=ovs - controller=remote,ip=192.168.42.129 Celle-ci généra les différents éléments de la couche physique qui sera connectée au contrôleur OpenDaylight à distance.

```
mininet@mininet-vm:~/mininet/custom$ sudo mn --mac --custom test4.py --topo mytopo --switch=ovs --controller=remote,ip=192.168.42.129
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.42.129:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s2, s1) (s3, s1) (s4, h1) (s4, h2) (s4, s2) (s5, h3) (s5, h4) (s5, s2) (s6, h5) (s6, h6) (s6, s3) (s7, h7) (s7, h8) (s7, s3)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
```

FIGURE 4.5: Création des éléments du réseau dans Mininet.

3.3.7 OpenDaylight

Le contrôleur OpenDaylight est un logiciel de système d'exploitation réseau open source développé en Java et géré par la fondation Linux. C'est l'un des contrôleurs SDN 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 réseaux traditionnels ou les réseau software-defined.

Il est basé sur une architecture modulaire et échange avec les applications du SDN en utilisant les Northbound API. Opendaylight communique avec les équipements réseaux avec sa Southbound API. La Southbound API la plus souvent utilisée dans le SDN est OpenFlow.

3.3.8 Openflow Manager

Le Software Defined Networking (SDN) implique une application interagissant avec un réseau 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 (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 [79].

- **Fonctionnalités d'OFM**

Les fonctions de bases proposées par l'application affichée en haut de l'écran de l'interface graphique sont subdivisées en 4 onglets :

- Basic view : Vue globale de la topologie sous-jacente. OFM schématise la structure durés eau 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.

3.3.9 Wireshark

Wireshark est un analyseur de paquets libres utilisé dans l'analyse et le dépannage de réseaux informatiques. Dans notre contexte, nous l'avons utilisé pour faire de l'investigation

sur les messages OpenFlow échangés entre le commutateur OpenFlow et le contrôleur OpenDaylight.

4.3. Simulation du réseau

La simulation de notre réseau repose sur l'utilisation de deux machines virtuelles distinctes :

La première machine virtuelle est dédiée à l'exécution d'OpenDaylight, une plateforme logicielle open source conçue pour le contrôle et la gestion de réseaux définis par logiciel (SDN). La deuxième machine virtuelle est réservée à l'exécution de Mininet, un émulateur de réseau virtuel. Mininet permet de créer un environnement de réseau virtuel réaliste et flexible, en émulant plusieurs hôtes, commutateurs et liens réseau.

En combinant OpenDaylight et Mininet, nous avons la possibilité de simuler et d'expérimenter des scénarios réseau complexes. Ceci nous permettra d'ouvrir des terminaux SSH pour une meilleure manipulation des VMs avec surtout la possibilité d'exécuter toutes sortes de programmes linux et de pouvoir lancer les interfaces graphiques d'ODL et OFM sur un navigateur local.

4.4. 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 appel à l'émulateur (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 à l'API OpenFlow Manager, nous effectuerons nos différentes configurations et politiques du réseau.

3.5.5 Topologie du réseau traditionnelle de sonatrach

La figure suivante illustre l'architecture du réseau traditionnelle pour l'entreprise Sonatrach Béjaia sous le logiciel GNS3.

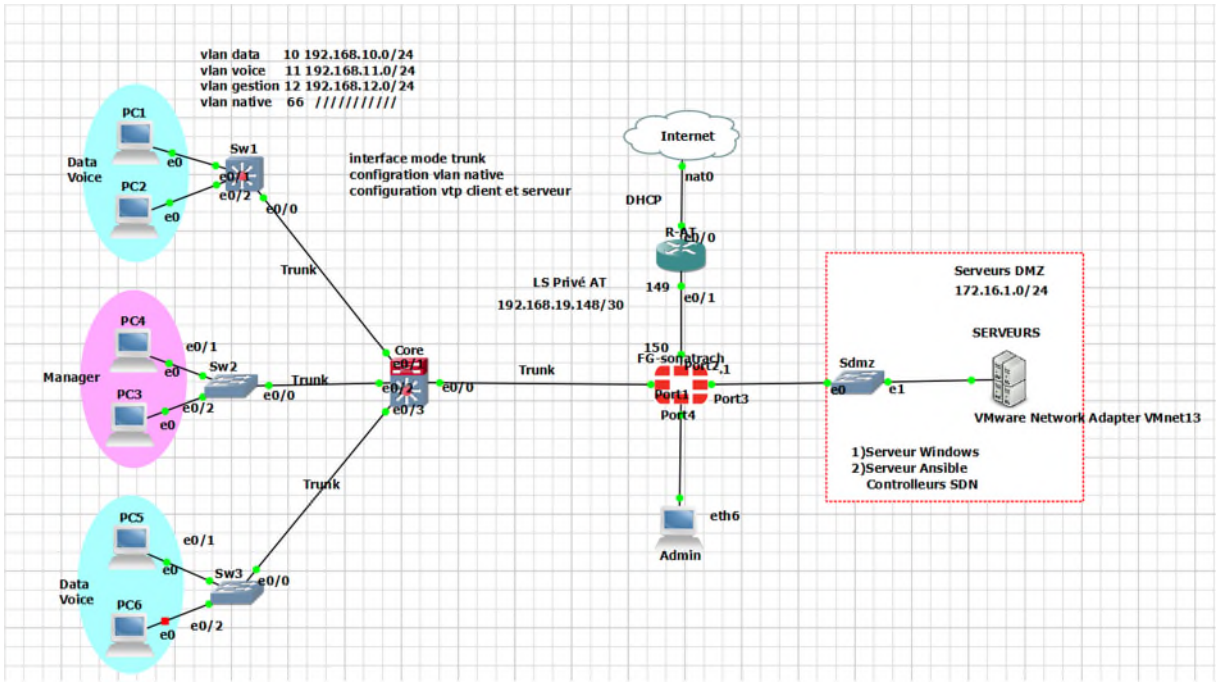


FIGURE 4.6: Topologie du réseau pour sonatrach.

3.5.6 Topologie proposée sur ODL

Dans notre déploiement réseau, nous avons mis en place un système composé de cinq régions. Chaque région est organisée de manière hiérarchique, avec des niveaux d'accès, de distribution et de cœur. La région au milieu joue un rôle central dans le réseau, car c'est là que le contrôleur SDN est connecté au commutateur OpenFlow 1. Cependant, pour des raisons de sécurité, le contrôleur SDN doit rester invisible dans le réseau, car il est considéré comme l'élément le plus critique.

Il est important de noter que notre réseau prend en compte la présence d'appareils IoT (Internet of Things), tels que les appareils sans fil avec capteurs connectés, équipement de contrôle et de surveillance, etc. qui collectent des données et communiquent entre eux. Le contrôleur détecte les switchs Openflow, et après un ping global détecte les hôtes et parvient à afficher la stucture du réseau sous-jacent comme l'atteste la « figure 4.7 ».

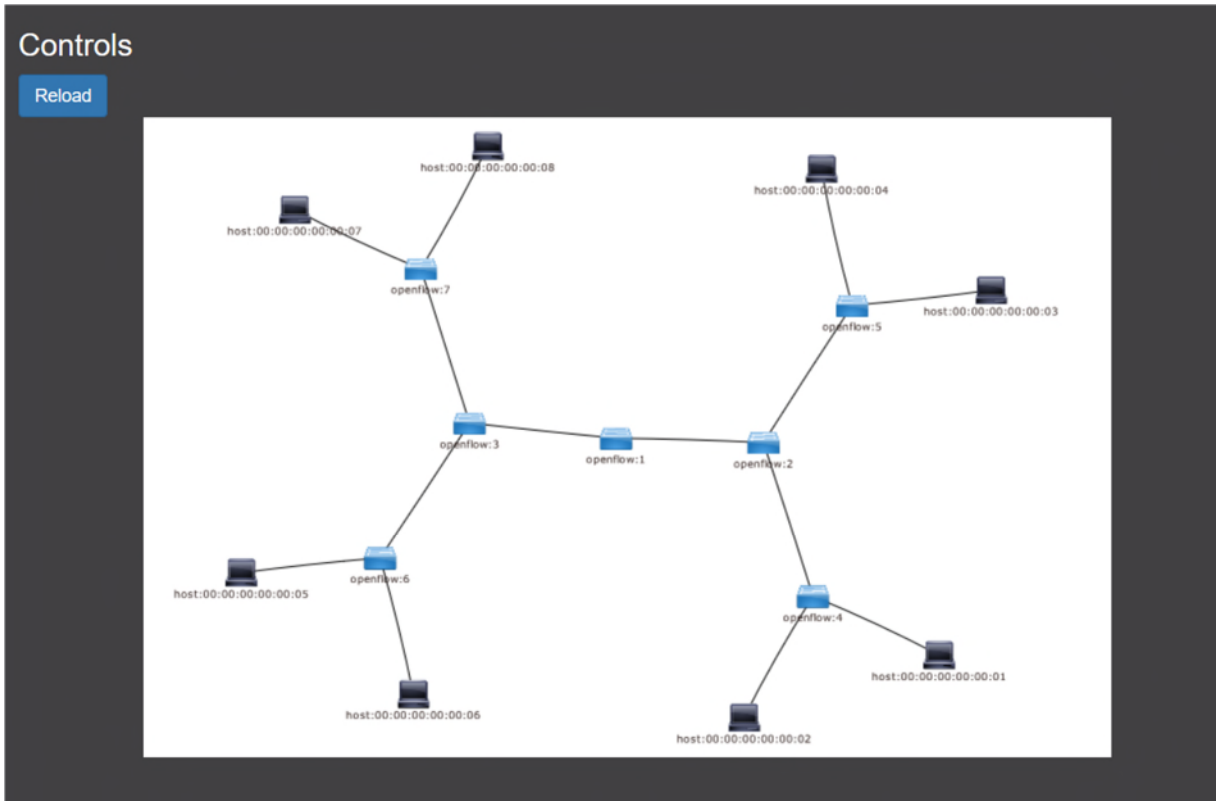


FIGURE 4.7 : Topologie du scénario affichée sur ODL.

- **Plan d'adressage**

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

VLANs	Hots	Adresse MAC	Adresse IP
Vlan 10	h1	00 :00 :00 :00 :00 :01	10.0.10.1/24
	h2	00 :00 :00 :00 :00 :02	10.0.10.2/24
Vlan 11	h3	00 :00 :00 :00 :00 :03	10.0.11.1/24
	h4	00 :00 :00 :00 :00 :04	10.0.11.2/24
Vlan 12	h5	00 :00 :00 :00 :00 :05	10.0.12.1/24
	h6	00 :00 :00 :00 :00 :06	10.0.12.2/24
Vlan 10	h7	00 :00 :00 :00 :00 :07	10.0.10.3/24
	h8	00 :00 :00 :00 :00 :08	10.0.10.4/24

TABLEU 4.1 : Plan d'adressage.

3.5.7 Topologie proposée sur OFM

Le contrôleur SDN effectue une détection des commutateurs OpenFlow et, grâce à un ping global, il identifie également les hôtes connectés au réseau. Ensuite, il utilise l'application

OpenFlow Manager déployée sur l'interface nord du contrôleur, qui écoute sur le port 9000. À travers cette application, la topologie du réseau est instantanément extraite et affichée dans l'interface graphique. Cela peut être observé dans la « Figure 4.8 ».

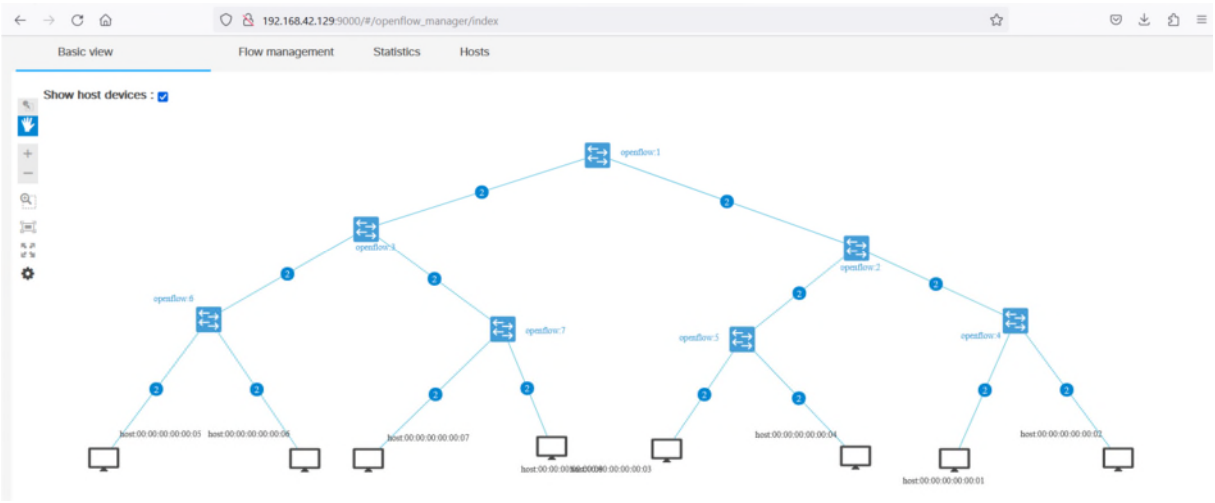


FIGURE 4.8 : Topologie du scénario affichée sur OFM.

On peut créer le flux, il suffit d'aller sur l'onglet Flow Management, et de cliquer sur le bouton 'show window for managing flows' comme suit :

FIGURE 4.9 : Accès à la gestion de flux.

Après cela, il suffit de compléter les cases choisies afin de générer l'entrée de flux sur le commutateur sélectionné, puis d'envoyer la requête en cliquant sur le bouton "envoyer la demande".

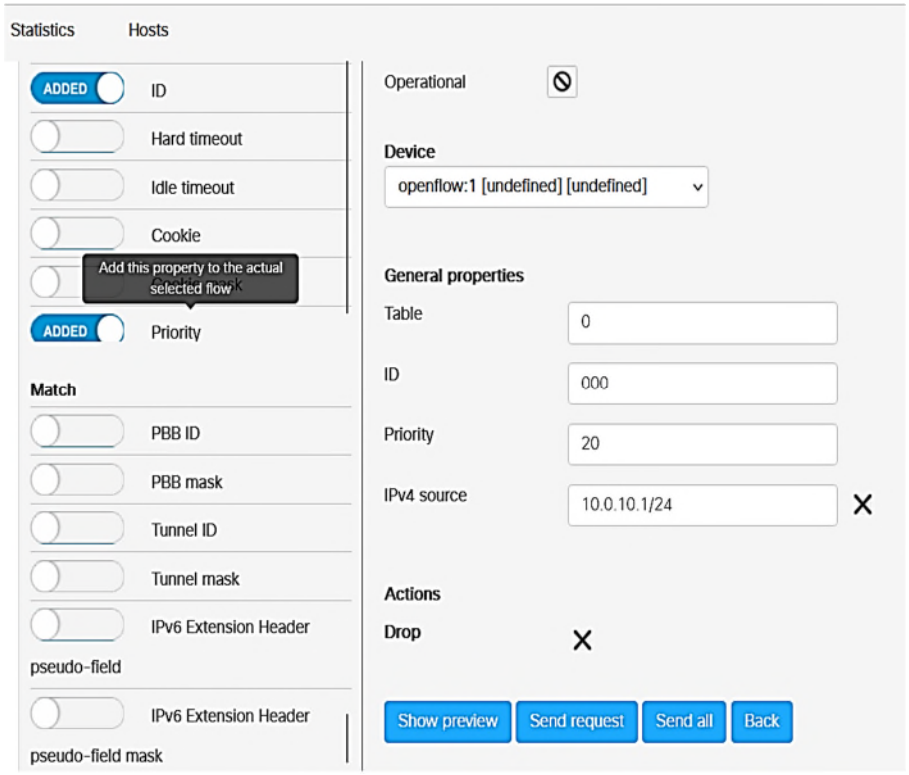


FIGURE 4.10 : Programmation de flux sur OFM.

- Voici Quelques flux introduit

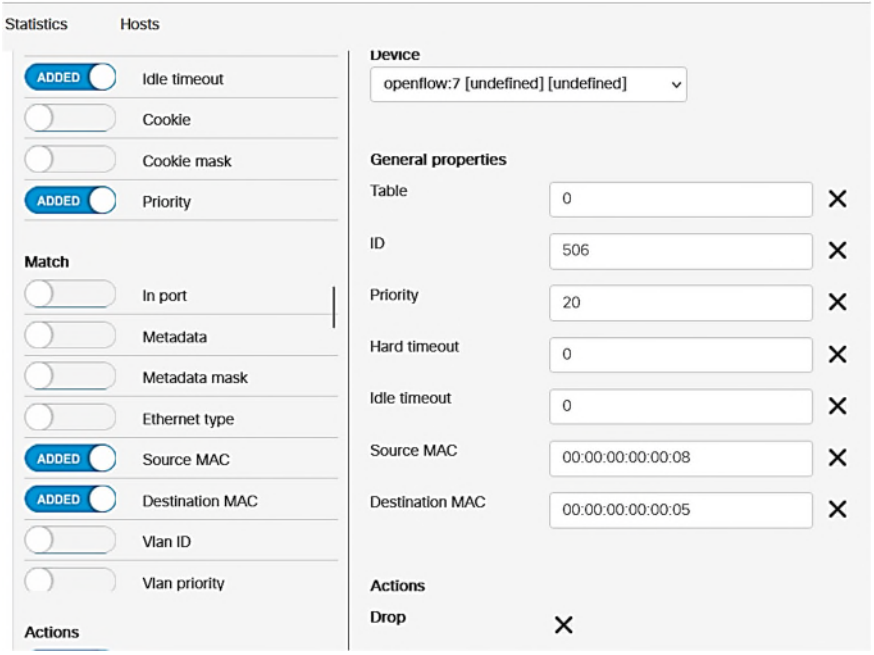


FIGURE 4.11 : Installation d'un filtre de niveau 2 sur s7.

- **Méthode d'implémentation des flux**

Pour faciliter la gestion des flux installés et offrir une interface conviviale, nous avons décidé d'utiliser l'application OFM (OpenFlow Manager). Cette application nous offre une visibilité complète via son interface graphique, ce qui simplifie l'édition et la suppression des flux précédemment installés.

Il est important de signaler que nous devons informer le contrôleur du type d'Ethernet utilisé pour l'identification des paquets. Par exemple, la valeur 2048 (0x0800) représente le type Ethernet IPv4. Cette information est cruciale pour que le contrôleur puisse reconnaître les paquets correspondant au flux spécifié et les traiter en conséquence.

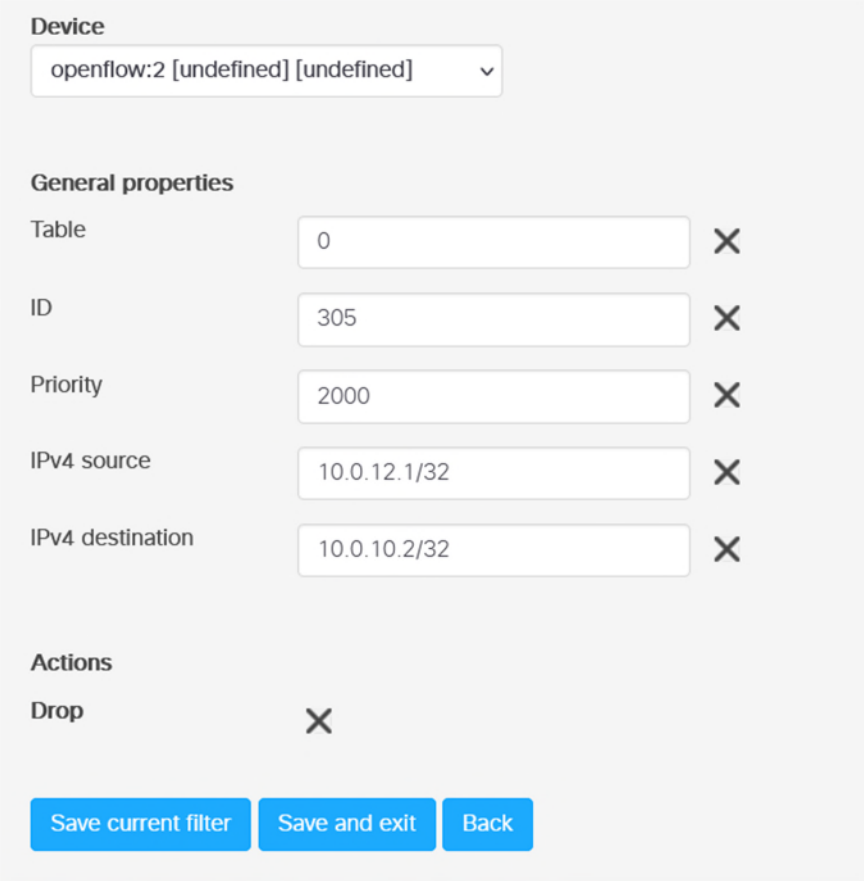
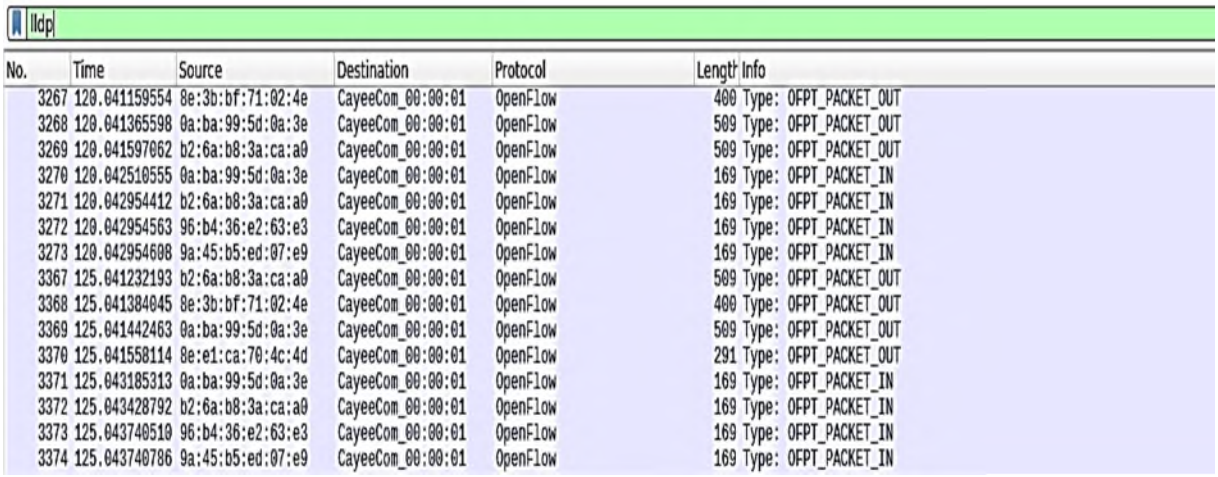


FIGURE 4.12 : Installation de l'entrée flux 305 sur OFM.

En utilisant OFM, nous avons pu bénéficier d'une approche conviviale pour la gestion des flux et nous assurer que notre réseau SDN est configuré de manière optimale pour prendre en charge les protocoles Ethernet spécifiques que nous utilisons.

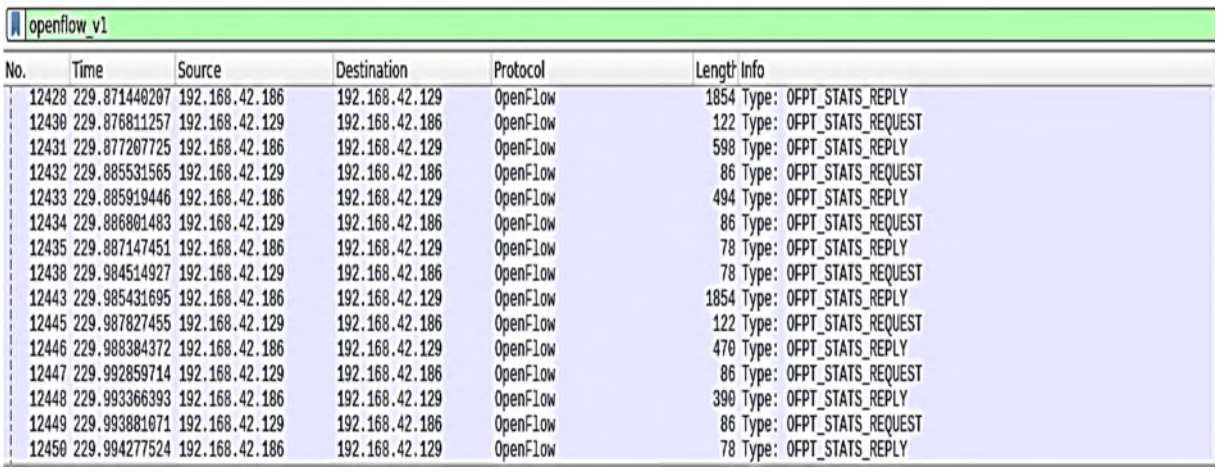
- **Le protocol OpenFlow**

Lorsque nous avons lancé la topologie, nous avons observé sur Wireshark que des messages OpenFlow circulent entre le contrôleur et Mininet. Ces paquets sont envoyés régulièrement pour gérer la topologie, accompagnés de paquets LLDP (Link Layer Discovery Protocol). Le LLDP est un protocole utilisé pour découvrir les topologies réseau et faciliter les échanges d'informations entre les équipements réseau et les utilisateurs finaux, comme le montrent les « figures 4.13 et 4.14».


lldp

No.	Time	Source	Destination	Protocol	Length	Info
3267	120.041159554	8e:3b:bf:71:02:4e	CayeeCom_00:00:01	OpenFlow	400	Type: OFPT_PACKET_OUT
3268	120.041365598	0a:ba:99:5d:0a:3e	CayeeCom_00:00:01	OpenFlow	509	Type: OFPT_PACKET_OUT
3269	120.041597062	b2:6a:b8:3a:ca:a0	CayeeCom_00:00:01	OpenFlow	509	Type: OFPT_PACKET_OUT
3270	120.042510555	0a:ba:99:5d:0a:3e	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3271	120.042954412	b2:6a:b8:3a:ca:a0	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3272	120.042954563	96:b4:36:e2:63:e3	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3273	120.042954608	9a:45:b5:ed:07:e9	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3367	125.041232193	b2:6a:b8:3a:ca:a0	CayeeCom_00:00:01	OpenFlow	509	Type: OFPT_PACKET_OUT
3368	125.041384045	8e:3b:bf:71:02:4e	CayeeCom_00:00:01	OpenFlow	400	Type: OFPT_PACKET_OUT
3369	125.041442463	0a:ba:99:5d:0a:3e	CayeeCom_00:00:01	OpenFlow	509	Type: OFPT_PACKET_OUT
3370	125.041558114	8e:e1:ca:70:4c:4d	CayeeCom_00:00:01	OpenFlow	291	Type: OFPT_PACKET_OUT
3371	125.043185313	0a:ba:99:5d:0a:3e	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3372	125.043428792	b2:6a:b8:3a:ca:a0	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3373	125.043740510	96:b4:36:e2:63:e3	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN
3374	125.043740786	9a:45:b5:ed:07:e9	CayeeCom_00:00:01	OpenFlow	169	Type: OFPT_PACKET_IN

FIGURE 4.13: Capture wireshark d'un Openflow.


openflow_v1

No.	Time	Source	Destination	Protocol	Length	Info
12428	229.871440207	192.168.42.186	192.168.42.129	OpenFlow	1854	Type: OFPT_STATS_REPLY
12430	229.876811257	192.168.42.129	192.168.42.186	OpenFlow	122	Type: OFPT_STATS_REQUEST
12431	229.877207725	192.168.42.186	192.168.42.129	OpenFlow	598	Type: OFPT_STATS_REPLY
12432	229.885531565	192.168.42.129	192.168.42.186	OpenFlow	86	Type: OFPT_STATS_REQUEST
12433	229.885919446	192.168.42.186	192.168.42.129	OpenFlow	494	Type: OFPT_STATS_REPLY
12434	229.886801483	192.168.42.129	192.168.42.186	OpenFlow	86	Type: OFPT_STATS_REQUEST
12435	229.887147451	192.168.42.186	192.168.42.129	OpenFlow	78	Type: OFPT_STATS_REPLY
12438	229.984514927	192.168.42.129	192.168.42.186	OpenFlow	78	Type: OFPT_STATS_REQUEST
12443	229.985431695	192.168.42.186	192.168.42.129	OpenFlow	1854	Type: OFPT_STATS_REPLY
12445	229.987827455	192.168.42.129	192.168.42.186	OpenFlow	122	Type: OFPT_STATS_REQUEST
12446	229.988384372	192.168.42.186	192.168.42.129	OpenFlow	470	Type: OFPT_STATS_REPLY
12447	229.992859714	192.168.42.129	192.168.42.186	OpenFlow	86	Type: OFPT_STATS_REQUEST
12448	229.993366393	192.168.42.186	192.168.42.129	OpenFlow	390	Type: OFPT_STATS_REPLY
12449	229.993881071	192.168.42.129	192.168.42.186	OpenFlow	86	Type: OFPT_STATS_REQUEST
12450	229.994277524	192.168.42.186	192.168.42.129	OpenFlow	78	Type: OFPT_STATS_REPLY

FIGURE 4.14: Capture wireshark d'un Openflow et LLDP.

- **Tables de flux introduites**

Dans le cadre de notre topologie, nous avons implémenté des tables de flux personnalisées dans les commutateurs de notre réseau à l'aide du contrôleur SDN. Ces tables de flux ont été configurées pour définir le comportement des commutateurs en fonction des paquets qu'ils reçoivent. Chaque table de flux a été spécifiquement créée pour répondre à nos besoins en matière de routage et de gestion du trafic. Grâce à cette configuration, nous avons pu contrôler la manière dont les paquets sont traités et acheminés à travers notre réseau. Cela nous a permis d'optimiser les performances et de personnaliser le comportement de notre réseau SDN selon nos exigences spécifiques.

Nous avons mis un exemple de configuration des tables de flux que nous avons introduites dans le Switch 7 de la topologie via le contrôleur SDN, décrivant son comportement en fonction des paquets reçu.

Openflow7						
N-packet	N-bytes	Src mac	Dst mac	In port	Action	Priority
2	140	: 07	: 08			10
3	182	: 08	: 07			10
4	280			3	Output : 1	2
14	644			1	Output : 3	2
8	392			2	Output : 3	2
19	1615				Controller : 65535	100
0	0				Drop	0

FIGURE 4.15 : Table de flux SW.OF.7.

4.5. Vérification de fonctionnement

Les tests pour vérifier que le flux est bien opérationnel sont multiples. D'abord vérifions que le flux a bien été introduit dans le switch avec la commande dump-flows :

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x2b000000000000c, duration=2217.835s, table=0, n_packets=41, n_bytes=3178, idle_age=208, priority=2,in_port=1 actions=output:2,CONTROLLER:65535
 cookie=0x2b000000000000d, duration=2217.835s, table=0, n_packets=22, n_bytes=1708, idle_age=2217, priority=2,in_port=2 actions=output:1,CONTROLLER:65535
 cookie=0x2b00000000000005, duration=2221.776s, table=0, n_packets=890, n_bytes=75650, idle_age=2, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000005, duration=2221.776s, table=0, n_packets=0, n_bytes=0, idle_age=2221, priority=0 actions=drop
```

FIGURE 4.16 : Résultat de la commande dump-flows s2.

Ensuite à partir de h1 dont l'adresse mac est 00:00:00:00:00:01 nous allons ping h2 pour vérifier que les paquets en provenance de l'adresse mac en question sont bel et bien bloqués :

```
mininet> h1 ping h2
PING 10.0.10.2 (10.0.10.2) 56(84) bytes of data.
64 bytes from 10.0.10.2: icmp_seq=1 ttl=64 time=0.175 ms
64 bytes from 10.0.10.2: icmp_seq=2 ttl=64 time=0.040 ms
^C
--- 10.0.10.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
```

FIGURE 4.17 : Résultat du test ping.

4.6. Tests de fonctionnement

3.7.5 Test d'accessibilité avant la création des VLANs :

Avant de créer les VLAN, nous avons effectué un test d'accessibilité. Lorsque nous avons lancé notre topologie sur Mininet, nous avons constaté que tous les hôtes étaient capables de communiquer entre eux.

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

FIGURE 4.18 : Test Avant la création des VLANs.

3.7.6 Test d'accessibilité après la création des VLANs :

La « figure 4.19 » représente le test de connectivité entre les différents hôtes de réseaux. On remarque que les hôtes qui appartiennent au même VLAN arrivent à se joindre tandis que le test de connectivité échoue entre deux hôtes qui appartiennent à deux VLANs différents.

Nous concluons que notre réseau local virtuel est fonctionnel.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X X X h7 h8
h2 -> h1 X X X X h7 h8
h3 -> X X h4 X X X X
h4 -> X X h3 X X X X
h5 -> X X X X h6 X X
h6 -> X X X X h5 X X
h7 -> h1 h2 X X X X h8
h8 -> h1 h2 X X X X h7
*** Results: 71% dropped (16/56 received)
mininet>
```

FIGURE 4.19 : Test après la création des VLANs.

4.7. Discussion

Dans cette partie, nous avons mis en œuvre une gestion centralisée dans la simulation de réseau SDN, avec un objectif d'optimiser la connectivité des hôtes. En utilisant un contrôleur SDN centralisé, nous avons réussi à améliorer la connectivité des applications IoT.

L'approche de gestion centralisée a offert plusieurs avantages, notamment une implémentation simplifiée et un gain de temps considérable dans la programmation du comportement des réseaux. Les résultats ont démontré la fiabilité de l'architecture, celui-ci a confirmé la conformité des communications envers ce qui a été préétabli.

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.

4.8. Conclusion

Dans ce quatrième chapitre, nous avons exposé les différentes opérations d'administration largement utilisées. En plus de leur facilité de mise en œuvre, ces opérations permettent d'économiser un temps considérable leur de la programmation du comportement des réseaux, ouvrant ainsi des nouvelles perspectives pour la mise en place des topologies complexes, notamment dans le contexte de l'Internet des objets. Bien que nous n'ayons pas encore identifié de véritable déploiement d'IoT au sein de Sonatrach, nous avons toutefois proposé des scénarios

qui pourraient être adopté à l'avenir. Les résultats obtenus ont démontré la fiabilité de cette architecture en termes de conformité aux politiques préétablies. De plus, Un test d'accessibilité a révélé le comportement du réseau en matière de restrictions, confirmant ainsi la conformité des communications par rapport à ce qui avait été préalablement établi.

Conclusion générale

Conclusion générale

L'émergence de l'Internet des objets a ouvert de nouvelles perspectives en matière de communication. Cette technologie permet de connecter différents objets, aux caractéristiques variées, dans une zone géographique restreinte, générant ainsi une densité de trafic réseau élevée et une quantité considérable de données. Cependant, la gestion et le contrôle de cette infrastructure réseau, avec les technologies de commutation et de routage traditionnelles, peuvent se révéler complexes, notamment en raison de l'hétérogénéité des dispositifs.

Plusieurs recherches ont été menées et diverses solutions ont été proposées pour faire face à ces défis, telles que Ansible, Node-red, OpenRemote, Flutter, etc. Toutefois, ces solutions se limitent généralement à des tâches spécifiques, contrairement aux réseaux définis par logiciel (SDN). Le SDN permet une gestion et un contrôle centralisés de toute l'infrastructure réseau, tout en prenant en charge l'aspect hétérogène des différents nœuds présents. L'un des principaux avantages du SDN réside dans sa couche d'application, qui permet un déploiement instantané de services. Cela ouvre la voie à de nouvelles perspectives et opportunités dans le domaine des réseaux, offrant une flexibilité et une agilité accrues pour répondre aux besoins changeants des utilisateurs et des applications.

Ce projet de fin d’études consiste à proposer une infrastructure réseau programmable, et mettre en pratique les connaissances théoriques à travers des méthodes professionnelles utilisées Dans l’entreprise SONATRACH. Pour la réalisation de ce travail nous avons examiné en détail le concept des réseaux définis par logiciel (SDN) et mis en œuvre cette technologie pour évaluer son impact sur le comportement des réseaux, ainsi que sa fiabilité. Nous avons également abordé l'un des principaux problèmes de l'Internet des objets et les raisons qui nous poussent à adopter de nouvelles approches telles que les réseaux SDN, qui visent à simplifier la gestion et le contrôle des infrastructures réseau. Nous avons réussi à mettre en place un réseau SDIoT (Software Defined Internet of Things) programmable grâce à l'interface OpenFlow Manager. À travers ce travail, nous avons utilisé divers outils de test qui ont confirmé la fiabilité et la connectivité de cette architecture. Les résultats obtenus démontrent que le SDN remet en question le modèle actuel des réseaux traditionnels et offre de nombreuses possibilités de programmation du comportement des flux de données. Cependant, des efforts supplémentaires sont nécessaires en termes de standardisation pour préparer une transition globale inévitable vers ce type de réseaux.

Ce projet a fait l'objet d'une expérience très intéressante et enrichissante, car il nous a permis d'appliquer nos connaissances en informatique plus particulièrement dans le domaine des réseaux et d'acquérir de nouvelles connaissances concernant le SDN et les avantages qu'elle a apporté à l'IoT.

Annexes

Annexe 1

❖ **Installation de mininet**

1. Prérequis

- Mettez à jour votre liste de paquets en exécutant la commande suivante :

```
sudo apt update
```

- Installez les dépendances nécessaires :

```
sudo apt install -y git make autoconf automake libtool build-essential python3-pip
```

- Clonez le dépôt Mininet GitHub en utilisant Git :

```
git clone git://github.com/mininet/mininet
```

- Accédez au répertoire Mininet :

```
cd mininet
```

- Exécutez le script d'installation en tant que super utilisateur : L'option -fnv permet de forcer l'installation et de désactiver les vérifications.

```
sudo util/install.sh -fnv
```

- Une fois l'installation terminée, vous pouvez vérifier si Mininet est correctement installé en exécutant la commande :

```
sudo mn --test pingall
```

Annexe 2

❖ Installation d'OpenDaylight

1. Préparer le système d'exploitation

Mettez à jour votre système d'exploitation Linux, vos applications et vos outils de sécurité à l'aide du gestionnaire de packages et installez le paquet « unzip» nécessaire pour décompresser les archives.

```
$ sudo apt-get -y update

$ sudo apt-get -y upgrade

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

2. Installer le JRE Java

OpenDaylight a été spécifiquement conçu par les architectes d'OpenDaylight pour fonctionner dans l'écosystème Java. Ainsi, pour pouvoir utiliser OpenDaylight, il est nécessaire d'avoir un environnement d'exécution Java (JRE) installé. Vous avez la possibilité d'utiliser le JRE autonome fourni dans un kit de développement logiciel Java, ou vous pouvez installer directement le JRE Java 8 par la commande suivante :

```
$ 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
```

La commande « 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.

3. 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".

```
$ echo 'export 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 source le fichier des ressources par la commande :

```
$ source ~/.bashrc
```

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

```
$ echo $JAVA_HOME
```

4. Téléchagez la version OpenDaylight souhaitée

Collez le lien dans une commande CURL

```
$ curl -XGET –O
https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/karaf/0.8.4/karaf-0.8.4.zip
```

5. 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 karaf-0.8.4.zip
```

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

6. Démarrer OpenDaylight

Maintenait vous pouvez lancer Opendaylight avec la commande

```
$ cd karaf-0.8.4/
```

On retrouve donc l'interface suivante :

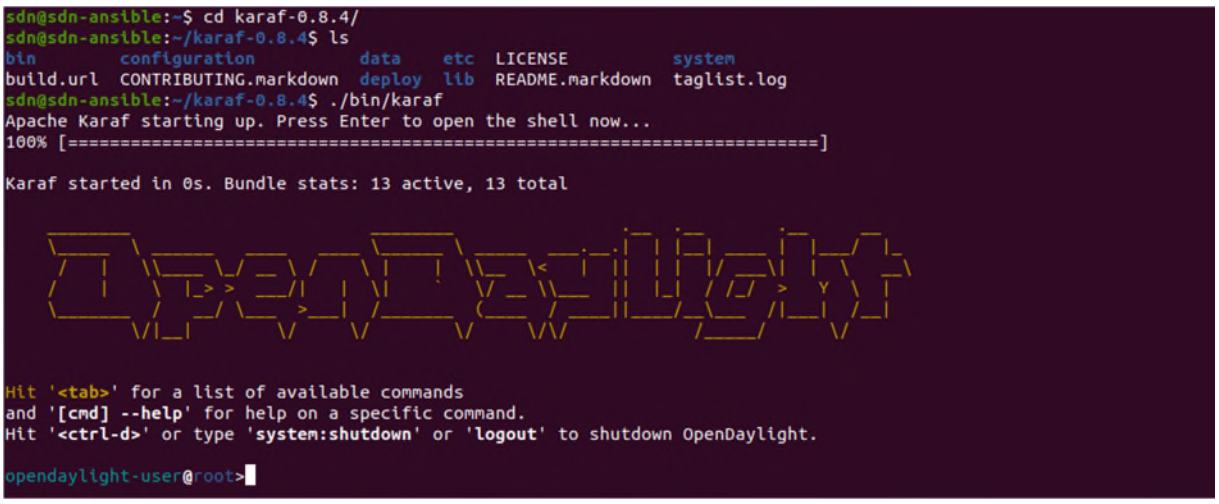


FIGURE 4.20 : lancement de OpenDaylight

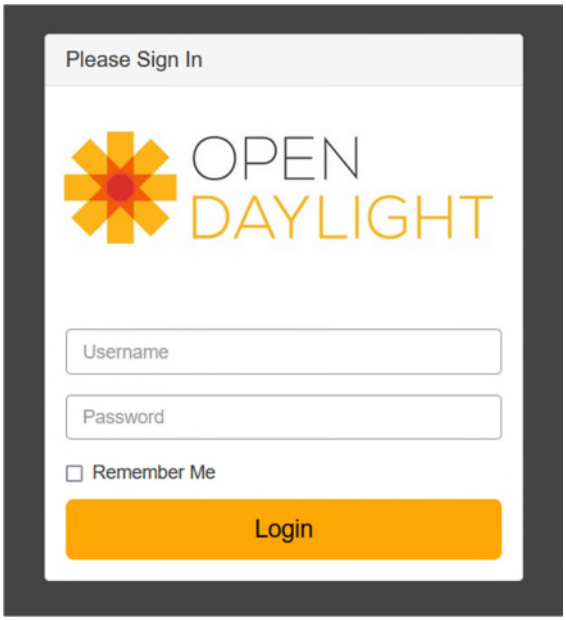
A présent vous devez installer les fonctionnalités suivantes :

```
Opendaylight-user@root>feature:list
```

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

7. Interface graphique d'opendaylight

Pour accéder à l'interface graphique d'OpenDaylight (GUI - Graphical User Interface), vous pouvez utiliser un navigateur web et entrer l'URL suivante dans la barre d'adresse :



http://<adresse IP de la machine virtuelle> : 8181/index.html.

FIGURE 4.21 : Interface de connexion à de OpenDayligh.

ur

- username=admin
- password=admin

Après l'identification, on accède à l'espace utilisateur qui permet d'avoir une vue globale sur les différentes topologies connectées au contrôleur.

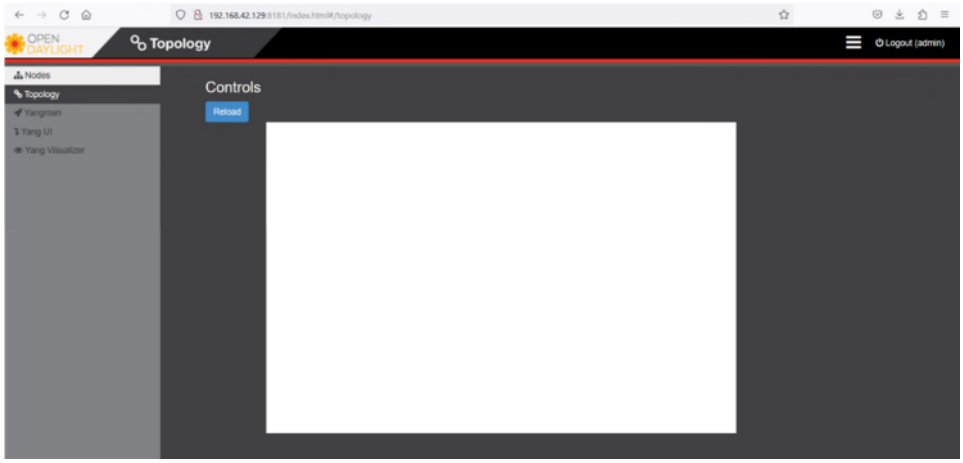


FIGURE 4.22: Interface web d'OpenDayligh

Annexe 3

❖ Installation d'OpenFlow Manager (OFM)

1. Prérequis

Avant de commencer, il est nécessaire d'assurer que Node.js, un environnement d'exécution JavaScript côté serveur, est installé. Cela est essentiel pour pouvoir exécuter Grunt et profiter de toutes ses fonctionnalités.

```
# curl –sl https://deb.nodesource.com/setup_4.x | sudo -E bash –

#apt-get install nodejs
```

2. Téléchargement d'OFM

- Clonage du répertoire github de l'application.

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

3. Configuration de l'adresse IP du serveur web :

```
# sed –i 's/localhost/<adresse IP de la machine>/g'
./opendaylightopenflowApp/ofm/src/common/config/env.module.js
```

```
root@sdn-ansible:/home/sdn/opendaylight-openflow-App/ofm/src/common/config# cat env.module.js
define(['angularAMD'], function(ng) {
    'use strict';

    var config = angular.module('config', [])
        .constant('ENV', {
          baseURL: "http://localhost:",
          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 4.23 : Le fichier env.module.js pour introduire l'adresse IP du contrôleur

- Installation de grunt

npm install –g grunt-cli

4. Exécution du serveur web d'OFM

- La commande grunt démarrera une instance du serveur web, et affichera ceci sur la ligne de commande

```
sdn@sdn-ansible:~/opendaylight-openflow-App$ grunt
Running "connect:def" (connect) task
Waiting forever...
Started connect web server on http://localhost:9000
```

FIGURE 4.24 : Lancement de OpenFlow Manager

5. Interface graphique d'openflow manager

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

FIGURE 4.25 : Interface utilisateur d'OFM

Bibliographie

Bibliographie

[1] A. Fuqaha, M. Guizani, M. Mohammadi et M. Aledhari, "Internet of Things: A survey on enabling technologies, protocols, and applications." IEEE Communications Surveys & Tutorials, 17(4), 2347-2376, 2015.

[4] S. Li, L. D. Xu et S. Zhao, "The Internet of Things: A Survey of Topics and Trends", Information Systems Frontiers, vol. 17, n° 2, pp. 261-274, 2015.

[5] D. E Vans, "The Internet of things: how the next evolution of the internet is changing everything, Cisco Internet Business Solutions Group (IBSG) ", 2011.

[6] A. Rafai and S. Kouah, " Développement d'un système d'iot (internet of things) pour le smart lighting sous la plateforme ibm", 2018.

[9] S. E. Elsakhy, M. A. Saeed, and M. A. Ibrahim, "Digital t-shirt, Ph.D. dissertation, Sudan University of Science and Technology", 2014.

[12] R. Tafazolli, "Technologies For The Wirel", "Ess Future" ESS FUTURE", 2006.

[13] M. Han and H. Zhang, "Business intelligence architecture based on internet of things" Journal of Theoretical & Applied Information Technology, vol.50, no.1, pp.90-95,2013.

[14] P- J. Benghozi, S. Bureau, F. Massit-Folléa, C. Waroquiers, and S. Davidson, "L'internet des objets : quels enjeux pour l'Europe, éd. De la Maison des sciences de l'homme éd ", 2009,66 p.

[15] A. Hakin, A. Gokhale, P. Berthou, D. C. Schmidt, T. Gayraud, "Software-Defined Networking : Challenges and research opportunities for Future Internet, Computer Networks", 2014.

[18] LEE, In et LEE, Kyoochun. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons " vol. 58, no 4, p. 431-440, 2015.

[20] P. Saarika, K. Sandhya, T. Sudha, "Smart transportation system using iot, in: 2017 International Conference On Smart Technologies for Smart Nation (SmartTechCon), IEEE" pp. 1104–1107, 2017.

[22] Y. Tseng, W. L. Yue, M. A. Taylor, "The role of transportation in logistics chain, Eastern Asia Society for Transportation Studies", 2005.

[23] D. Singh, G. Tripathi, A. Jara, "A survey of internet-of-things: Future vision, architecture, challenges and services, in: 2014 IEEE world forum on Internet of Things (WF-IoT), IEEE", 2014, pp. 287–292.

[25] R. Andrew, R. Malekian, D. C. Bogatinoska, "Iot solutions for precision agriculture, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE", pp. 0345–0349, 2018.

[27] K. Patel, M. Sunil, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application et future challenges. International journal of engineering science and computing, 6(5) ", 2016.

[28] I. Saleh, "Internet des objets (ido) : Concepts, enjeux, défis et perspectives. Internet des objets", 1(1) :5, 2017.

[29] L. Antao, R. Pinto, J. Reis, and G. Gon¸calves. "Requirements for testing and validating the industrial internet of things. In 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) ", pages 110–115. IEEE, 2018.

[30] J. Mineraud, "Internet of things: A review and future directions." Journal of Cleaner Production, 195, 85-103, 2018.

[31] F. Skandrani , "IoT industriel : Architecture IoT ",

[32] Y. CHALLAL Y, "Sécurité de l'Internet des Objets : vers une approche cognitive et systémique", HDR, Juin 2012.

[33] L. Farhan, R. Kharel, O. Kaiwartya, M. Quiroz-Castellanos, A. Alissa, and M. Abdulsalam. "A concise review on internet of things (iot)-problems, challenges and opportunities. In 2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP) ", pages 1–6. IEEE, 2018.

[34] S. Vashi, Jyotsnamayee Ram, J. Modi, S. Verma, and C. Prakash. "Internet of things (iot): A vision, architectural elements, and security issues. In 2017 international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) ", pages 492–496. IEEE, 2017.

[35] K. Patel, M. Sunil, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. International journal of engineering science and computing", 6(5), 2016.

[38] D. Inchaadamou. "Réseaux de collecte de données pour les zones blanches étendues. PhD thesis", Université Paris Saclay, 2019.

[39] A. Nizar, A. Ali, "Comparison study between IPV4 & IPV6", Philadelphia University, Jordan, CIS, department IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, 2012.

[40] M. Khairil, R. Hassan, "A Comparative Review of IPv4 and IPv6 for Research Test Bed", Department of Computer Science, Universiti Kebangsaan Malaysia 43600 UKM Bangi Selangor, Malaysia, 2009 International Conference on Electrical Engineering and Informatics.

[42] Y. Jiang, Y. Zhang, "IPv6-based internet of things: technology and applications. International Journal of Distributed Sensor Networks", 2017.

[43] E; Vyncke, "IPv6 Security,Scott Hogg, CCIE No. 5133,Cisco Press 800 East 96th Street Indianapolis", IN 46240 USA.

[44] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions. Future generation computer systems", 29(7):1645–1660, 2013.

[46] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP) ", "Universitaet Bremen TZI", 2014.

[47] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota. Evaluation of constrained application protocol for wireless sensor networks. In Local & Metropolitan Area Networks (LANMAN), 18th IEEE Workshop on, pages 1–6. IEEE, 2011.

[48] L. Chai, R. Reine, "Performance of UDP-Lite for IoT network". In: *IOP Conference Series: Materials Science and Engineering*. T. 495. 1. IOP Publishing. 2019.

[49] Hung-Cuong Le, Violeta Felea, "an overhearing based mac protocol for wireless sensor networks. In sensorcomm" pages 547–553. IEEE, 2007.

[50] D. Locke, "MQ telemetry transport (MQTT) v3. 1 protocol specification", IBM developerWorks, Markham, ON, Canada, Tech. Lib, 2010.

[51] Z. Dawy, W. Saad, A. Ghosh, J. Andrews, and E. Yaacoub, "Toward massive machine type cellular communications", IEEE Wireless Communications, vol. 24, pp. 120–128, 2017.

[52] M. Hasan, E. Hossain, and D. Niyato, "Random access for machineto-machine communication in lte-advanced networks: issues and approaches", IEEE Communications Magazine, vol. 51, pp. 86–93, 2013.

[53] J. Choi, "Two-stage multiple access for many devices of unique identifications over frequency-selective fading channels", IEEE Internet of Things Journal, vol. 4, pp. 162–171, 2017.

[54] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, "IoT Connectivity Technologies and Applications: A Survey", arXiv:2002.12646v1, 2020.

[55] Z. Ding, Y. Liu, J. Choi, Q. Sun, M. Elkashlan, V. Poor, "Application of non-orthogonal multiple access in lte and 5g networks", IEEE Communications Magazine, vol. 55, pp. 185–191, 2017.

[56] J. Ding, H. Jiang, and T. Jiang, "Success probability of grantfree random access with massive mimo", IEEE Internet of Things Journal, vol. 6, pp. 506–516, 2019.

[57] U. Madhow, D. R. Brown, S. Dasgupta, and R. Mudumbai, "Distributed massive mimo: Algorithms, architectures and concept systems", Information Theory and Applications Workshop (ITA), pp. 1–7, Feb, 2014.

[58] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues" IEEE Communications Surveys Tutorials, pp. 1–1, 2019.

[60] I. Choukri, M. Ouzzif, K. Bouragba, "Software Defined Networking (SDN): Etat de L'art. Colloque sur les Objets et systèmes Connectés, Ecole Supérieure de Technologie de Casablanca (Maroc)", Institut Universitaire de Technologie d'Aix-Marseille (France), Jun 2019, CASABLANCA, Maroc. Hal -02298874.

[61] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, et J. Turner, "Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review", 38(2), 69-74, 2008.

[63] Y. Matnee, C. Abooddy, Z. Mohammed, "Analyzing Methods and Opportunities in Software Defined (SDN) Networks for Data Traffic Optimizations", International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 6 Issue: 1,2018.

[64] M. Jaafreh, "Toward integrating software defined networks with the Internet of Things: a review". In : Cluster Computing, p. 1-18, 2021.

[67] M. Abdelaziz, S. Abdelwahab, "Étude et Simulation D'une Solution SDN dans L'internet Des Objets", mémoire de Master, Université de Bejaia, 2021.

[69] B. Astuto, A. Nunes, "A survey of software-defined networking: Past, present, and future of programmable networks". In: IEEE Communications surveys & tutorials 16.3, p. 1617-1634, 2014.

[70] Y. Jararweh, "SDIoT: a software defined based internet of things framework". In: Journal of Ambient Intelligence and Humanized Computing 6.4, p. 453-461, 2015.

[71] F. Olivier, N. Florent, "Gestion dynamique et évolutive de règles de sécurité pour l'Internet des Objets". Thèse de doctorat, Université de Reims, 2019.

[72] A. Bradai, M. Rehmani, I. Haque, M. Nogueira, Syed Hashim Raza Bukhari "Software-Defned Networking (SDN) and Network Function Virtualization (NFV) for a Hyper connected World: Challenges, Applications, and Major Advancements", 2020.

[73] L. YONG, M. CHEN, "Software-Defined Network Function Virtualization: A Survey", Received September 14, 2015, accepted October 2, 2015, date of publication December 9, 2015, date of current version December 16, 2015.

[74] T. Soltana, "Problème de placement du contrôleur dans les réseaux programmables"Thèse de doctorat. Faculté : mathématique et informatique Département : informatique-Option, 2020 : RTIC. Consuler sur. Url: http://dspace.univmsila.dz:8080//xmlui/handle/123456789/21693.

[76] B. Pfaff, "The Design and Implementation of Open (vSwitch)". In: 12th USENIX symposium on networked systems design and implementation (NSDI 15). p. 117-130 2015.

[78] B. Lantz, B. OConnor, Cody Burkhard. Mininet: http://mininet.org.

[79] S. Jamrich, C. Eckel. "OpenDaylight OpenFlow Manager (OFM) ": https://github.com/CiscoDevNet/OpenDaylight-Openflow-App.

Webographie

[2] https://wikimemoires.net/2019/09/internet-des-objets-iot-def-sens-objet/. Consulté le 25/02/2023.

[3] https://www.lagazette des communes.com/395738/les-objets-connectes-mais-de-quoi-parle-t-on/ . Consulté le 20/02/2023.

[7] https://www.objet-connecte.info/avis-electricite/belkin-wemo-switch-interrupteur-domotique/. Consulté le 22/02/2023.

[8] https://www.lenstore.fr/soins-des-yeux/un-regard-sur-lavenir-des-lentilles-de-contact/. Consulté le 22/02/2023.

[10] https://www.usine-digitale.fr/article/la-start-up-chronolife-lance-un-t-shirt-connecte-pour-mesurer-six-facteurs-physiologiques-en-continu.N907929. Consulté le 22/02/2023

[11] https://images.app.goo.gl/54irqk15HtEppgCv7. Consulté le 23/02/2023

[16] https://wikimemoires.net/2019/09/internet-des-objets-iot-def-sens-objet/. Consulté le 25/02/2023.

[17] https://www.ringcentral.com/fr/fr/blog/iot/. Consulté le 25/02/2023.

[19] https://www.researchgate.net/figure/Cloud-computing-scenario. Consulté le 20/03/2023.

[21] https://www.construction21.org/france/amp/articles/h/journ-eacute-e-mobilit-eacute-3-0-quels-syst-egrave-mes-de-transports-intelligents-pour-l-arc-m-eacute-diterran-eacute-en-jeudi-20-juin-agrave-aix-en-provence.html. Consulté le 25/02/2023.

[24] http://visioforce.com/smarthome.html .Consulté le 25/02/2023.

[26] https://hmkfarm.com/ung-dung-iot-trong-nong-nghiep/iot-in-agriculture/. Consulté le 25/02/2023.

[36] https://www.journaldunet.fr/web-tech/dictionnaire-de-l-iot/1181267les-reseaux-iot/?Fbclid=IwAR25u7dl-D-oLbaHbuDGU6nAWNxc1raVvKAIgVBoNYVkxTzOnjTM-XTiR8 . Consulté le 13/03/2023.

[37] http://www.commentcamarche.net/contents/524-le-protocole-ipv6. Consulté le 11/03/2023.

[41] https://www.researchgate.net/figure/IP-smart-objects-protocol-stack_fig5_318474040. Consulté le 21/03/2023.

[45] https://www.memoireonline.com/01/09/1878/m-Les-technologies-sans-fil-Le-routage dans-les-réseaux-ad-hoc-OLSR-et-AODV2.html. Consulté le 16/03/2023.

[59] « Software-Defined Networking (SDN) Definition », Open Networking Foundation: https://www.opennetworking.org/sdn-definition/. Consulté le 07/04/2023.

[62] Open Networking Foundation. (2012). Software-defined networking: the new norm for networks. Retrieved from https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. Consulté le 10/04/2023.

[65] software-defined networking: http://blogs.univ-poitiers.fr/flaunay/2018/01/15/principe-du-sdn-dans-une-architecture-reseau-classique/, consulté le 07/04/2023.

[66] openflow-switch-v1.5.1.pdf: https://www.opennetworking.org/wpcontent/uploads/2014/10/openflow-switch-v1.5.1.pdf. Consulté le 23/03/2023.

[68] OpeFlow Switch Specification: https://www.opennetworking.Org, Consulté le 05/04/2023.

[75] Linux Foundation Collaborative Projec. 2016 Production Quality, Multilayer Open Virtual Switch. Disponible sur. Url : https://www.openvswitch.org/.

[77] Linux Foundation Collaborative Project. 1 April 2013: https://air.imag.fr/index.php/Open%5C_vSwitch .

.

Résumé

L’IoT est Un réseau ouvert d’objets intelligents qui ont la capacité de s’auto-organiser, de partager des informations, des données et des ressources. À cause de l'augmentation de ces objets connectés, la gestion et la coordination de ces dispositifs deviennent de plus en plus complexes. C'est là qu'intervient le SDN, ou Software-Defined Networking qui est un nouveau paradigme réseau permet de simplifier la gestion dans le réseau, en séparant la logique de contrôle du réseau des équipements d’interconnexions et en promouvant la centralisation du contrôle et la capacité de programmer le réseau. Notre objectif principal était d'étudier l'architecture du réseau informatique de Sonatrach, dans le cadre d'un stage réalisé en collaboration avec le département des Systèmes Informatiques de l’entreprise pour d'optimiser la gestion, la connectivité et les performances du réseau IoT de Sonatrach en utilisant l'approche innovante du SDN, nous avons utilisé l’émulateur de réseaux Mininet et le contrôleur OpenDylight.

Mots clés : IoT, objets connectés, SDN, Mininet, contrôleur OpenDylight, Réseaux programmables.

Abstract

The IoT is an open network of intelligent objects that have the ability to self-organize, share information, data and resources. Due to the increase of these connected objects, the management and coordination of these devices are becoming more and more complex. This is where SDN comes in, or Software-Defined Networking, which is a new network paradigm that simplifies management in the network, by separating the network control logic from the interconnection equipment and by promoting centralization of control and the ability to program the network. Our main objective was to study the architecture of Sonatrach's computer network, as part of an internship carried out in collaboration with the company's IT Systems department to optimize the management, connectivity and performance of the network. Sonatrach IoT using the innovative SDN approach, we used the Mininet network emulator and the OpenDylight controller.

Keywords: IoT, connected objects, SDN, Mininet, OpenDylight controller, Programmable networks.