REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

OOREDOO

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

Mise en place d'un réseau IP/MPLS et l'intégration d'une plateforme Web pour la gestion et l'exploitation du réseau d'accès 2G 3G 4G

ZAIDI Mohamed Arysse

Présenté et soutenu publiquement le (04/07/2022)

Composition du jury :

Président	**Mme.Bouadjenek**	**MCB.**	**ENP**
Promoteur	**Mme.Lani**	**MAA.**	**ENP**
Examinateur	**Mr.Taghi**	**MAA.**	**ENP**
Promoteur	**Mr.Saghir**	**DGA.**	**OOREDOO**

ENP 2022

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

OOREDOO

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'ingénieur d'état en électronique

Mise en place d'un réseau IP/MPLS et l'intégration d'une plateforme Web pour la gestion et l'exploitation du réseau d'accès 2G 3G 4G

ZAIDI Mohamed Arysse

Présenté et soutenu publiquement le (04/07/2022)

Composition du jury :

Président	**Mme.Bouadjenek**	**MCB.**	**ENP**
Promoteur	**Mme.Lani**	**MAA.**	**ENP**
Examinateur	**Mr.Taghi**	**MAA.**	**ENP**
Promoteur	**Mr.Saghir**	**DGA.**	**OOREDOO**

ENP 2022

ملخص

قد كان المجتمع المدني منذ القرن التاسع عشر موضوع إحالة في الخطابتتكون إدارة الشبكة من التحكم في موارد الشبكة ، وتنسيق خدماتها ، ومراقبة حالاتها ، والإبلاغ عن حالة الشبكة والشذوذ

.

مع عدد لا يحصى من التقنيات التمكينية التي ظهرت في السنوات الأخيرة ، وكلها تقدم درجات متفاوتة من التوزيع وفوائد الشبكات ، فليس من الواضح ما هي هذه التقنيات ومتى وأين تكون أكثر قابلية للتطبيق؟ وما هو نوع الشبكة الأمثل لاستخدام مشغلي الهاتف؟

يعرض هذا المستند تنفيذ شبكة MPLS / IP OOREDOO ، وإنشاء قاعدة بيانات لشبكات VLAN المختلفة وعنوان IP للواجهات المختلفة لكل تقنية بالإضافة إلى إنشاء واجهة WEB لإدارة الشبكة مع جمع البيانات تلقائيًا باستخدام مكتبات شبكة Python.

كلمات مفتاحية : الوصول إلى الشبكات ، ومحاكيات الشبكة ، وقاعدة البيانات ، وواجهة الويب IP/MPLS، Python ,

Abstract

Network management is about controlling network resources, coordinating its services, monitoring network status and reporting anomalies. With the co-existence of various access technologies that are constantly evolving, all offering many degrees of distribution and network management benefits, it is not clear what, when and where these technologies are most applicable? And what is the optimal type of network for telephone operators to use?And what is the challenge of network automation? This paper presents the implementation of the OOREDOO IP/MPLS network, the establishment of the database of the various VLANs and the IP address of each access network as well as the realisation of a WEB interface for network management with automatic data update using the Python network libraries.

Key words : IP/MPLS, access networks, Python, network emulators : GNS3 Eve-NG, databases, Web interfaces.

Résumé

La gestion du réseau consiste à contrôler les ressources du réseau, à coordonner ses services, surveiller ses états et à signaler l'état du réseau et les anomalies. Avec la cohexistence de diverses technologies d'accès qui ne cessent d'évoluer, toutes offrant à des degrés divers une distribution et des avantages en matière de gestion de réseau, il n'est pas évident de savoir à quoi, quand et où ces technologies sont le plus applicables? Et quel est le type de réseau optimal à utiliser pour les opérateurs téléphoniques? Et quel est l'enjeu de l'automatisation des réseaux? Ce document présente l'implémentation du réseau IP/MPLS de OOREDOO, l'établissement de la base des données des divers VLAN et l'adresse IP de chaque réseau d'accès ainsi que la réalisation d'une interface WEB pour la gestion du réseau avec une mise à jour automatique des données grâce aux librairies réseau de Python.

Mots clés : IP/MPLS, réseau d'accès, Python, émulateurs réseaux : GNS3 Eve-NG, Base de données, interface Web.

Dédicace

Je dédie ce travail à la mémoire de mon défunt grand-père,

À la mémoire de mon défunte grand-mère,

À ma source d'inspiration,

À ma merveilleuse mère, mon père et mes soeurs,

À ma famille,

À mes amis,

Et à tous ceux qui m'ont soutenu dans mon parcours.

Remerciemennts

Je voudrais dans un premier temps remercier, mon encadrante ***Mme.LANI****, pour sa confiance, elle m'a toujours soutenu dans mes projets et a toujours été à l'écoute.*

Je remercie également ***M. HASSANI Kamel*** *et* ***M. SAGHIR Moncef*** *ainsi que toute l'équipe de* ***OOREDOO*** *pour leur disponibilité, leurs précieux conseils et pour m'avoir introduit au monde professionnel.*

Je tiens à témoigner toute ma reconnaissance envers mes parents et mes soeurs pour m'avoir offert l'environnement adéquat afin de me focaliser sur mon travail.

Je remercie sincèrement les membres du jury, ***Mme. BOUADJNEK Nesrine*** *et* ***M.TAGHI Mohamed Oussaid****, qui ont généreusement offert leur temps, leur soutien, leurs conseils et leur bonne volonté durant ces trois dernières années.*

Enfin, je remercie Mélissa LARBI, mes cousins YAHIA AISSA Ilyas et Massil et mes amis CharefEddine, Nadir, Pedro, et les papas qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

Table des matières

Table des figures

Liste des abréviations

Introduction générale **14**

1 Présentation de l'organisme d'accueil **16**

1.1 Introduction 17

1.2 Historique 17

1.3 Présentation de OOREDOO Algérie 17

1.4 Les valeurs de OOREDOO 18

1.5 Le réseau d'OOREDOO 19

1.6 Organigramme 20

1.7 Conclusion 21

2 État de l'art et concepts **23**

2.1 Introduction 24

2.2 L'état de l'art 24

2.3 Généralités sur les réseaux informatiques 25

2.3.1 Définition 25

2.3.2 Les types de réseaux informatiques 25

2.3.3 La topologie des réseaux informatiques 26

2.3.4 Les supports de transmission 28

2.3.4.1 Les câbles à paires torsadées 28

2.3.4.2 Les cables à fibre optique 29

2.3.4.2.a fibre monomode(4) 30

2.3.4.2.b fibre multimode(4) 30

2.3.5 Le modèle OSI 31

2.3.6 Le modèle TCP/IP 32
2.3.6.1 La couche application du modèle TCP/IP 32
2.3.6.2 La couche transport du modèle TCP/IP 32
2.3.6.3 La couche réseau du modèle TCP/IP 33
2.3.6.4 Les couches liaison de données et physique du modèle TCP/IP 33
2.3.7 Comparaison entre le modèle TCP/IP et OSI 33
2.3.8 Les éléments physiques d'un réseau 34
2.3.8.1 Le hub ou concetrateur 34
2.3.8.2 Le switch ou commutateur 35
2.3.8.3 Le routeur 35
2.3.8.4 Le fonctionnement d'un routeur 35
2.3.9 Les techniques de routage 36
2.3.9.1 OSPF 36
2.3.10 Les VLAN 37
2.4 Généralités sur les télécommunications 39
2.4.1 Global System for Mobile Communication 39
2.4.1.1 L'architecture des réseaux GSM 40
2.4.1.2 La station de base BTS(26) 41
2.4.1.3 Le contrôleur de station de base BSC(26) 41
2.4.1.4 L'interface Abis 42
2.4.2 Universal Mobile Telecommunications System 43
2.4.2.1 L'architecture des réseaux UMTS 43
2.4.2.2 La node B 44
2.4.2.3 Le RNC 44
2.4.2.4 L'interface IuB 45
2.4.3 Long Term Evolution 45
2.4.3.1 L'architecture des réseaux 4G (LTE) 46
2.4.3.2 L'eNode-B 47
2.4.3.3 Les interfaces S1/X2 47
2.5 L'IP/MPLS 47
2.5.1 l'en-tête MPLS 48
2.5.2 Le protocole LDP 49
2.6 Les opérations de l'MPLS 49

2.7 L'architecture MPLS 50
2.8 L'ingénierie du trafic 50
2.9 Conclusion 52

3 Les bases de données, les interfaces WEB et l'interaction entre les deux 53
3.1 Introduction 54
3.2 Les bases de données 54
3.2.1 La modélisation Entité-Association 55
3.2.1.1 Le type entité 55
3.2.1.2 Le type association 55
3.2.1.3 Attribut et Valeur 55
3.2.1.4 MySQL(20) 56
3.3 Les interfaces Web 56
3.3.1 HTML 56
3.3.1.1 Quelques exemples de HTML tag 57
3.3.2 CSS 57
3.3.3 Jinja 58
3.4 La relation entre les bases de données et les interfaces Web 58
3.4.1 Python 58
3.4.2 Flask 59
3.4.2.1 Render template 59
3.4.2.2 La fonction request 60
3.4.3 Flask-MySQLAlchemy(9) 60
3.5 Conclusion 60

4 Mise en oeuvre du réseau 61
4.1 Introduction 62
4.2 Outils d'implémentation réseau 62
4.2.1 Présentation des émulateurs 62
4.2.1.1 GNS3 62
4.2.1.2 Eve-NG 63
4.3 Telnet et SSH 63
4.4 Programmation réseau Python 63
4.4.1 telnetlib 63

4.4.2 Netmiko et Parmiko 64
4.5 But de l'implémentation 64
4.6 Réalisation sur GNS3 64
4.6.1 Découpage réseau et tests 66
4.7 Réalisation sur Eve-NG 68
4.8 Tests des performances 71
4.9 Comparaison entre GNS3 et Eve-NG 72
4.10 Conclusion 72

5 Intégration de la base de données et de la plateforme Web, tests et résultats 73
5.1 Introduction 74
5.2 Développement de la problématique 74
5.3 Solution proposée 76
5.4 Développement et réalisation de la solution 77
5.4.1 Réalisation de la base de données 77
5.4.2 Réalisation de l'interface Web 83
5.4.2.1 L'affichage de l'IP PLAN et la recherche de sites 83
5.4.3 Liaison entre la base de données et l'interface Web 85
5.4.3.1 Connexion avec la base de données 86
5.4.4 Récupération des données en étant connecté au matériel 89
5.4.5 Ajout des données récupérées à notre base de données 95
5.5 Conclusion 100

Conclusion générale 101

Bibliographie 103

Annexes 106
.1 Annexe A : Code des pages Web en HTML et jinja 107
.2 Annexe B : Code de la stylisation des pages Web en CSS 110
.3 Annexe C : Code utilisée pour lier la base de données et nos pages Web en utilisant la librairie Flask de Python et ses fonctionnalitées 111
.4 Annexe D : Code des différentes librairies réseau en Python 115
.5 Annexe E : Configuration de nos appareils sur GNS3 : 119

Table des figures

1.1 Logo de OOREDOO Algérie de 2004 à 2009 19
1.2 Logo de OOREDOO Algérie de 2010 à 2013 19
1.3 Logo actuel de OOREDOO Algérie 19
1.4 Organigramme de OOREDOO Algérie 21

2.1 Différence entre les réseaux LAN, MAN et WAN 26
2.2 La topologie en BUS des réseaux informatiques 27
2.3 La topologie en étoile des réseaux informatiques 27
2.4 La topologie en anneau des réseaux informatiques 28
2.5 Les composants d'un câble à fibre optique 29
2.6 Transmission sur une fibre monomode 30
2.7 Transmission sur une fibre multimode 30
2.8 Les couches du modèle TCP/IP 32
2.9 Comparaison entre le modèle OSI et le modèle TCP/IP 34
2.10 Fonctionnement d'un Hub 34
2.11 Deux domaines de diffusions avec deux switchs différents 38
2.12 Deux domaines de diffusions avec un seul switch 38
2.13 L'architecture des réseaux GSM 40
2.14 L'architecture des réseaux UMTS 43
2.15 L'architecture des réseaux LTE 46
2.16 Les interfaces du réseau LTE 46
2.17 L'en-tête MPLS 48
2.18 L'architecture des réseaux MPLS 50

4.1 Tolopogie réalisée sur GNS3 65
4.2 Résultat de la commande show ip route sur le routeur R1 67
4.3 Résultat de la commande Show mpls ip binding sur le routeur R1 67

4.4 Résultat de la commande Show ip bgp all summary sur le routeur R1 . . 67
4.5 Résultat de la commande SShow ip route vrf PFE sur le routeur R1 . . . 68
4.6 Tests de connectivité entre le routeur R1 le routeur R3 et R8 68
4.7 Topologie réalisée sur Eve-NG en étant connecté aux serveurs de OOREDOO 69
4.8 Utilisation de Putty afin de se connecter au matériel via telnet ou SSH . 69
4.9 Configuration du routeur VMX1 sur Eve-ng 70
4.10 Configuration de nos switchs sur Eve-ng 70
4.11 Test de la connectivité de Python avec notre matériel en utilisant la librairie Netmiko 71

5.1 L'IP Plan de OOREDOO 74
5.2 L'IP Plan de OOREDOO 74
5.3 Organigramme de la démarche à suivre afin de réaliser notre projet . . . 77
5.4 L'IP Plan de OOREDOO 78
5.5 Création de la base de données sur MySQL 78
5.6 Transformation de notre fichier excel en fichier .csv 80
5.7 Visualisation de nos tables dans notre base de données 82
5.8 Tables sites dans MySQL 82
5.9 Tables ach dans MySQL 82
5.10 Premier rendu de notre interface Web 84
5.11 Ajouter un site dans notre base de données à partir de l'interface Web . . 85
5.12 Tests et résultats sur notre interface Web 89
5.13 output de notre routeur juniper après commande sur l'IP_RAN 91
5.14 description de l'interface 91
5.15 output de la commande show configuration interfaces ae41.1375 92
5.16 output de la commande show interfaces description sur GNS3 93
5.17 output de la commande show run interface fa0/0.10 sur GNS3 94
5.18 output de notre programme Python en utilisant la librairie telnetlib . . . 94
5.19 le fichier PFE.csv qui a été créé par notre code contenant l'output de nos commandes sur le routeur 96
5.20 Les changements sur notre fichier.csv grâce à la librairie regex 97
5.21 Le fichier .csv final après traitement 98
5.22 Le fichier .csv prêt à être ajouté à notre base de données 99

5.23 L'ajout automatique à notre base de données et l'affichage sur l'interface Web 99
5.24 Organigramme de la démarche suivie lors du projet de fin d'étude . . . 102

Liste des abréviations

AuC	Authentification Center
BSC	Base Station Controller
BTS	Base Tranceiver Station
CSS	Cascading style sheets
EDGE	Enhanced Data Rates for GSM Evolution
EPC	The Evolved Packet Core
E-UTRAN	The Evolved UMTS Terrestrial Radio Access Network
FEC	Forward Equivalence Class
GSM	Global System for Mobile communications
GPRS	General Packet Radio Service
GNS3	Graphical Network Simulator 3
HLR	Home Location Register
HTML	Hypertext Markup Language
IP	Internet Protocol
IEEE	Institute of Electrical and Electronics Engineers
LAN	Local Area Network
LTE	Long Term Evolution
LDP	Label distribution protocol
NMS	Network Management System
’N’G	’N’ième génération
MPLS	Multiprotocol Lable Switching
MAC	Media Access Control

MSC	Mobile Switching Center
MS	Mobile Station
MSC	Mobile service Switching Center
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OMC	Operation and Maintenance Center
PDP	Packet Data Protocol
QoS	Quality of Service
RNC	Radio Network Controller
SNMP	Simple Network Management Protocol
S-GW	Serving Gateway
SLA	Service-level agreement
TCP	Transmission Control Protocol
3GPP	Third Generation Partnership Project
UMTS	Universal Mobile Telecommunication Systems
UTP	Unshielded twisted pairs
UE	User Equipement
VLAN	Virtual Local Area Network
VLR	Visitor Location Register
Web	World Wide Web

Introduction générale

Les réseaux informatiques sont les éléments essentiels des technologies actuelles des transmissions des données entre sites éloignés. Avec la modernisation des services de télécommunications, les réseaux informatiques ainsi que leurs protocoles de communication ont trouvé un très grand usage en télécommunications, notamment pour la transmission de données. L'usage des protocoles IP en télécommunications sert principalement à assurer la convergence des diverses technologies 2G 3G et 4G via le réseau IP/MPLS. Les opérateurs téléphoniques ont su faire face à l'augmentation de la demande et à la cohexistence des différentes technologies d'accès et assurer la qualité de services, grâce à l'apparition de nouvelles techniques et protocoles comme l'IP/MPLS. Cependant, il est aussi indispensable de décentraliser les données et de faciliter l'accès afin d'optimiser la gestion et l'exploitation des données. Le but de ce travail est en premier lieu la mise en place d'un réseau IP/MPLS en utilisant de différents émulateurs ce qui nous permet de comparer les performances d'un réseau traditionnel avec un réseau MPLS. Nous proposons dans une deuxième partie l'intégration d'une plateforme Web pour la gestion et l'exploitation du réseau d'accès 2G 3G 4G de OOREDOO. Nous ferons ensuite appel à Python afin de favoriser l'automatisation de notre réseau en utilisant les librairies d'automatisation des réseaux. Ce document est organisé comme suit :

- **Chapitre 1 : Présentation de l'organisme d'accueil**

Dans ce chapitre, nous présentons notre entreprise d'accueil en passant par son historique tout en introduisant ses valeurs et sa vision.

- **Chapitre 2 : État de l'art et concepts**

Dans ce chapitre, nous avons proposé un état de l'art, et on introduit des concepts de base en réseaux et en télécommunications.

- **Chapitre 3 : Les bases de données, les interfaces WEB et l'interaction entre les deux**

Dans ce chapitre on introduit les concepts de base en base de données MySQL, HTML, CSS et la librairie Python qui nous permet de lier le tout : Flask.

- **Chapitre 4 : Implémentation**

Dans ce chapitre, on introduit les différents protocoles de communication et de configurations d'équipements. On présente des librairies réseau de Python et nous implémentons le réseau IP/MPLS de OOREDOO sur GNS3 et Eve-ng. Nous proposons une comparaison entre les deux émulateurs utilisés. Nous veillons aussi à tester les performances de notre réseau IP/MPLS comparé à un réseau IP traditionnel.

- **Chapitre 5 : Réalisation, tests et résultats**

Dans ce chapitre, nous réalisons notre base de données, interface Web, et utilisons les librairies réseau nécessaires pour la gestion, tout en montrant les tests faits et les résultats obtenus.

- **Conclusion générale**

Dans la conclusion générale, on revient sur l'essentiel de notre travail, en énumérant les principaux résultats et on présente des perspectives.

Chapitre 1

Présentation de l'organisme d'accueil

1.1 Introduction

Nous présenterons dans ce chapitre l'entreprise d'accueil OOREDOO Algérie en passant par son historique et en décrivant ses valeurs.

1.2 Historique

WATANIYA TELECOM ALGÉRIE (WTA) a été fondée par la société koweïtienne Wataniya Telecom, à qui par la suite s'est jointe United Gulf Bank (UGB). Bénéficiant d'une licence dont la durée est de 15 ans, WTA a suivi plusieurs programmes d'investissements. Suite à cela, Nedjma se place parmi leader de l'innovation et de la plus-value, rendant la technologie multimédia accessible à tous.
L'opérateur de référence Watanya Telecom, a émergé en 1999 au Koweït. Faisant partie des sociétés de KoweIt Projects COmpany (KIPCO), elle devient la plus importante entreprise privée du Koweït dotée d'actif de plus de 10 milliards USD. Wataniya Telecom connaît à ce jour une croissance considérable dans l'univers des télécommunications sans fil partout dans le monde. En 2007, Qtel détiendra (80%) de Nedjma, en devenant actionnaire majoritaire (51%) de Wataniya Telecom Kuwait.
En novembre 2013, Nedjma à connu un grand changement lors d'une conférence de presse organisée dans le presitgieux hôtel Sheraton. Ce jour-là, le changement officiel de son identité commerciale et visuelle fut annoncée. Prenant son nouveau nom, Nedjma deviendra alors Ooredoo (qui se traduit par le mot «je veux»).

Le transfert de Nedjma vers Ooredoo s'est fait en parallèle avec la mise en service de la 3G.

1.3 Présentation de OOREDOO Algérie

Ooredoo peut être considéré comme étant le premier investisseur dans ce secteur dans le pays.
En 2013, les investissements de la filiale algérienne de Ooredoo ont pu atteindre plus de 485 millions de dollars US, contre 226 millions en 2012. Selon un communiqué de l'opérateur Quatari, cela pourrait représenter (19%)des investissements totaux de la maison mère de Ooredoo.

La progression de son revenu ainsi que son bénéfice net, qui ont doublé en quelques années a ainsi fait de cette entreprise l'opérateur téléphonique le plus puissant en Algérie. Détenue en majorité par le qatari Qtel, Ooredoo a obtenu à la fin d'année 2003 une licence permettant l'exploitation de la téléphonie mobile dans le pays. Le lancement commercial de sa marque a eu lieu l'année suivante, ces numéros téléphoniques commençant par l'indicatif 05 xx xx xx xx, donnent un numéro a 10 chiffres.

Ooredoo, leader des télécommunications, est une compagnie internationale qui procure les services de l'Internet haut débit et de téléphonie mobile, ainsi que des services parfaitement adaptés aux besoins des entreprises et des particuliers, et cela, à travers les marchés dans les quatre coins du monde tel que le Moyen Orient, d'Afrique du Nord et du Sud-Est asiatique.

Orientée vers les populations, Ooredoo a pour vision d'enrichir la vie des populations et a pour conviction le pouvoir d'assurer la stimulation du développement humain à travers la communication.

Ooredoo se positionne dans plusieurs marchés, tels que le Qatar, le Koweït, la Tunisie,la Palestine,le Sultanat d'Oman, l'Algérie, l'IraK et les Maldives.

La compagnie a pu réaliser des revenus de l'ordre de 9,3 milliards de dollars avec une base clientèle globale pouvant dépasser les 92,9 millions de clients jusqu'au 31 décembre 2012.

Anciennement Qatar telecom (Qtel), la maison mère de Ooredoo devient Ooredoo Q.S.C. Ses actions sont cotées à la Bourse du Qatar « Qatar Exchange » ainsi qu'à la Bourse d'Abou Dhabi, « Abu Dhabi Securities Exchange ».

1.4 Les valeurs de OOREDOO

Prônant le changement continu, la toute nouvelle marque Ooredoo fut lancée en fin d'année 2013. Dans le respect des valeurs de Nedjma, Ooredoo a pour objectif de les enrichir et les renforcer :

Caring : Incarne le respect des autres.

Connecting : Incarne l'esprit de collaboration de Ooredoo.

Challenging : Incarne la recherche d'amélioration perpetuelle.

FIGURE 1.1 – Logo de OOREDOO Algérie de 2004 à 2009

FIGURE 1.2 – Logo de OOREDOO Algérie de 2010 à 2013

FIGURE 1.3 – Logo actuel de OOREDOO Algérie

1.5 Le réseau d'OOREDOO

Le réseau Ooredoo a été mis en place à des vitesses record pour fournir aux consommateurs algériens des communications de qualité exceptionnelles en émission et en réception. Ooredoo utilise un réseau GSM avec des fréquences de 900/1800 et un réseau GPRS/EDGE pour les applications data. Selon l'Autorité de régulation des postes et télécommunications (ARPT), le réseau Ooredoo couvre 99% de la capitale de la Wilaya et plus de 95% des routes de masse et nationales. Le 15 décembre, Nedjma,

devenu Ooredoo, procédera au déploiement commercial du réseau 3G HSPA+ après avoir reçu l'agrément ARPT sous le label 3G++, tout en collaborant parallèlement avec l'opérateur domestique Mobilis pour un événement de déploiement. le premier jour. Ouargla, Constantine, Sétif, Djelfa et Béjaïa uniquement,Chlef, bouira et Ghardaia. Elle continuera ensuite à opérer à Boumerdes, Blida, Tipaza, Tlemcen, Sidi Bel Abbès, Eindefra, Biskra et Eloued. Et exclusivement Médéa.

1.6 Organigramme

L'organigramme suivant schématise l'organisation d'OOREDOO Algérie :

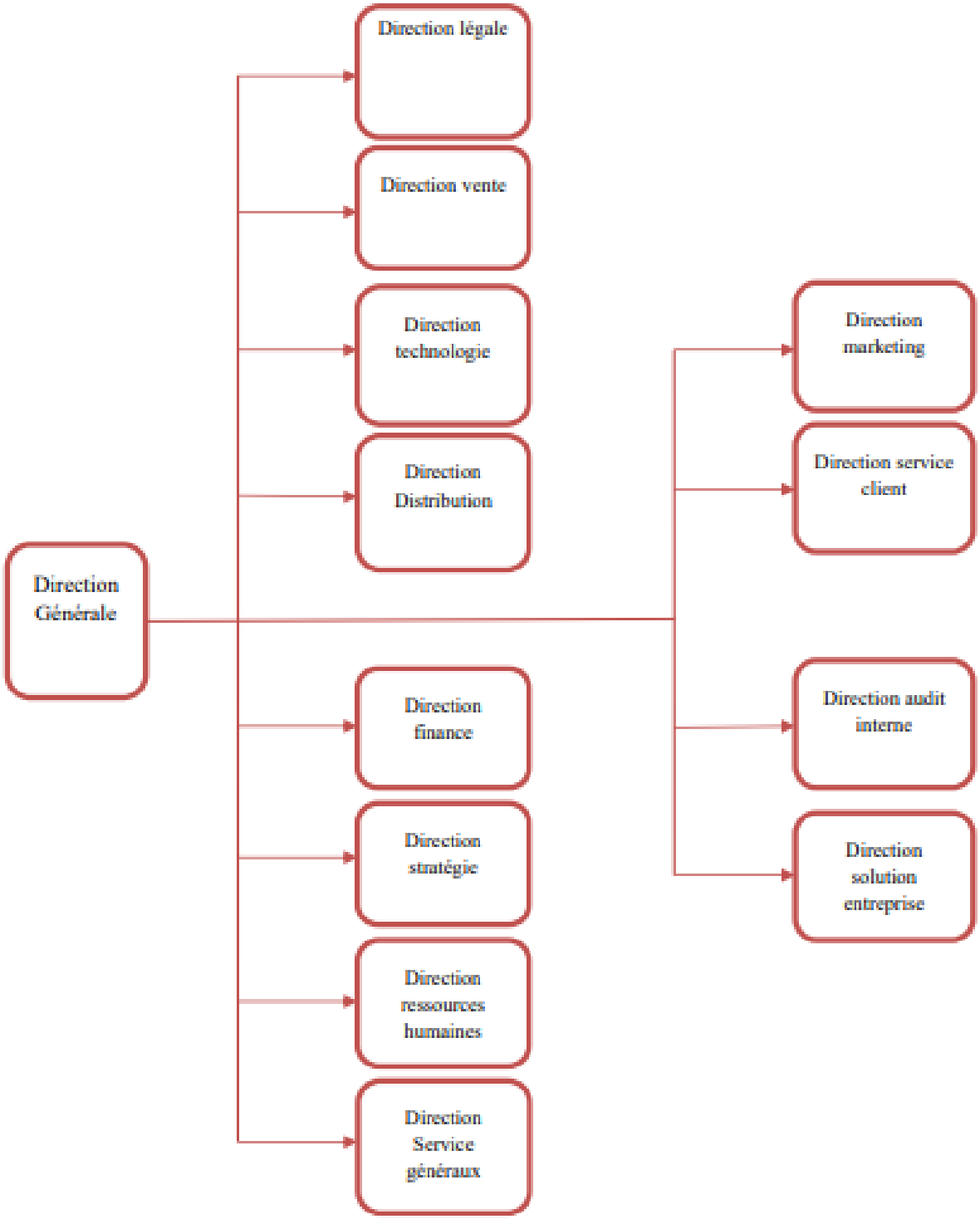


FIGURE 1.4 – Organigramme de OOREDOO Algérie

1.7 Conclusion

Dans ce chapitre, nous avons présenté l'organisme d'accueil de notre projet de fin d'étude.

Nous allons introduire dans le prochain chapitre l'état de l'art de notre thème ainsi

que des concepts sur les réseaux et les télécommunications utilisés pour concrétiser nos objectifs.

Chapitre 2

État de l'art et concepts

2.1 Introduction

Nous présenterons dans ce chapitre l'état de l'art de l'apport de la technologie IP/MPLS et l'enjeu de l'automatisation des réseaux en utilsant Python. Nous détaillerons ensuite les concepts du domaine des réseaux informatiques ainsi que les télécommunications, nécessaires pour la compréhension de la suite de notre projet.

2.2 L'état de l'art

Le réseau traditionnel basé sur un routeur relie chaque site homologue par des liaisons point à point dédiées ou des lignes louées. Cette approche est coûteuse, non évolutive et nécessite un déploiement long et difficile, bien qu'elle promette une connexion totalement sécurisée et fiable, une bande passante élevée et une bonne qualité de service. Une autre alternative qui émule les liaisons point à point est l'utilisation de services MPLS (Multi-Protocol Label Switching) avec une meilleure qualité de service.

Les opérateurs mobiles savent qu'ils doivent soutenir leurs infrastructures existantes génératrices de revenus tout en explorant les options de migration vers un backbone IP/MPLS plus évolutif, robuste et convergeant. La capacité des réseaux IP/MPLS à prendre en charge efficacement tous les services existants tout en permettant le déploiement rapide et évolutif de nouveaux services mobiles a conduit à un passage croissant à l'IP/MPLS.

Diverses simulations et expériences ont été réalisées pour analyser les performances des réseaux MPLS. . D. Adami et al. (14) ont discuté de la conception et du développement des plans de contrôle et de données qui sont nécessaires pour fournir un support de Label Switching Path (LSP) dans un nœud MPLS. En particulier, ils ont développé un simulateur et mis en œuvre de nouveaux modules logiciels pour le calcul du chemin des LSP Peer-to-Peer (P2P), le protocole de signalisation RSVP-TE et le mécanisme d'acheminement. D. Adami et al. ont proposé un nouveau module ns-2 pour accélérer la conception, le développement et le déploiement d'un réseau MPLS DiffServe-aware. MPLS DiffServaware permet aux opérateurs de réseaux de fournir des services qui exigent des garanties strictes de performance en matière de QoS(Quality of Service). Le nouveau module logiciel est utilisé pour simuler le protocole RSVP-TE à l'aide du simulateur ns-2. N. Aslam et A. Yassar(18) a comparé les performances des réseaux

MPLS et des réseaux IP. Une topologie de réseau est conçue et un outil de simulation basé sur MATLAB est utilisé pour envoyer des données en masse dans un réseau. La performance du réseau est mesurée avec MPLS activé ou désactivé. L'auteur montre que le réseau MPLS peut être plus performant que les réseaux IP traditionnels.
Il est cependant aussi important de pouvoir gérer et exploiter les réseaux d'accès des opérateurs mobiles en faisant usage de tout les outils modernes qui facilitent énormément la visualisation et la gestion et diminuent au maximum l'erreur humaine ce qui est le but principal de l'automatisation des réseaux. Kateryna Mariushkina et Joel Pettersson(16) ont réalisé un système qui réagit au changement sur le réseau. P. Mihăilă, T. Bălan, R. Curpen, F. Sandu(17) ont démontré l'importance de l'automatisation des réseaux et qu'en utilisant Python, les ingénieurs réseau n'ont pas besoin de configurer eux-mêmes chaque appareil, il leur suffit de créer l'infrastructure adéquate et en mettant en œuvre des scripts d'automatisation.

2.3 Généralités sur les réseaux informatiques

2.3.1 Définition

Un réseau informatique se présente comme étant un ensemble de terminaux et d'ordinateurs interconnectés dans le but principal est d'échanger des informations numériques. Faire circuler des élements entre chaque objet représente sa fonction principalle, cela se fait en fonction de règles définies(4).

2.3.2 Les types de réseaux informatiques

En fonction de la distance, du débit et de la localisation, les réseaux sont catégorisés en trois sortes :
MAN (Metropolitan Area Network) : Ce réseau permet d'établir une connexion entre plusieurs sites a l'échelle d'une ville.
LAN (Local Area Network) : Permet l'échange et partage de ressources a l'échelle local.
WAN (Wide Area Network) : Le plus utilisé des WAN est Internet, c'est un réseau qui oppère a l'echelle d'un pays.

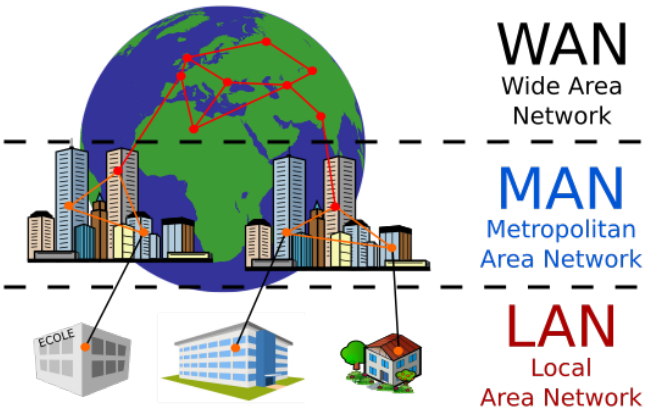


FIGURE 2.1 – Différence entre les réseaux LAN, MAN et WAN

2.3.3 La topologie des réseaux informatiques

Ce terme désigne la façon dont les élements d'un réseau sont connectés les uns aux autres.(22) Un réseau informatique se compose d'ordinateurs liés entre eux à l'aide du matériel comme des cartes réseau, des câbles ainsi que plusieurs autres équipements qui permettent d'assurer la bonne circulation des données. Nous trouvons 3 topologies physiques :

La topologie en BUS : Tous les éléments sont interconnectés au même bus et se partagent le même support de transmission. Cette topologie a certains avantages comme la simplicité, étant donné qu'un seul câble peut permettre toutes les communications. Cependant, elle présente un certain nombre d'inconvénients tel que l'obligation de mettre des terminaisons supplémentaires aux extrémités du bus afin d'éviter le phénomène de réflexion du à l'écho du signal. Un seul défaut de liaison à un seul endroit conduit à un réseau qui ne peut pas opérer. Tous les éléments partagent la meme bande passante, ce qui explique le fait que le débit de transmission diminue dès que des postes sont ajoutés.

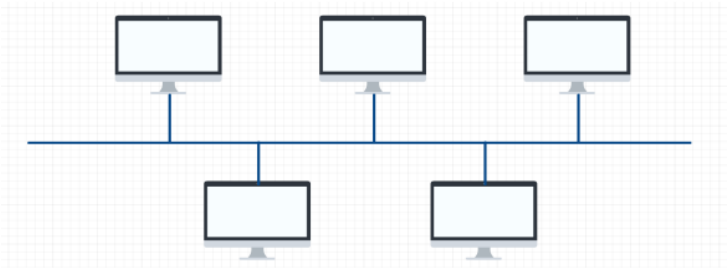


FIGURE 2.2 – La topologie en BUS des réseaux informatiques

La topologie en étoile : Le système est constitué d'un équipement central (le concentrateur ou hub) qui relie tout nos équipements. Si le concentrateur (hub) rencontre une panne, le réseau devient indisponible. Cependant, le fonctionnement du réseau ne s'arrête pas en retirant une station.(23)

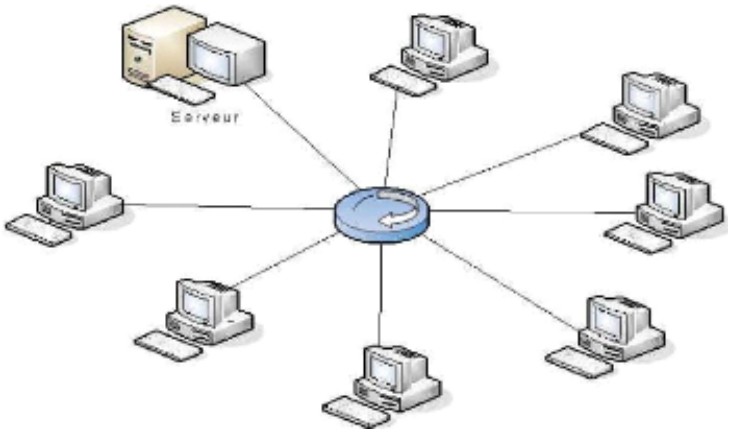


FIGURE 2.3 – La topologie en étoile des réseaux informatiques

La topologie en anneau : Cette topologie est formée d'une boucle fermée liant tous les éléments. Chaque élément se comporte comme un répéteur, les données peuvent passer via ces répéteurs. Les concentrateurs actifs sont souvent utilisés par les réseaux en anneau et prend le rôle du Multisation Access Unit (MAU).

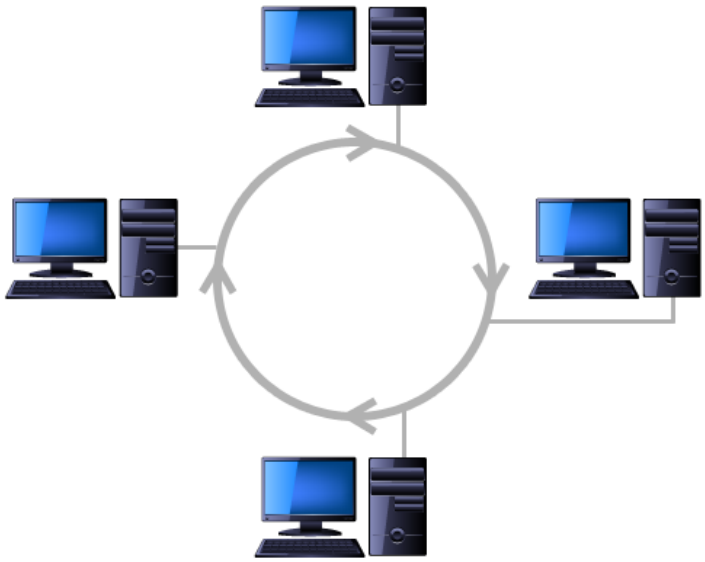


FIGURE 2.4 – La topologie en anneau des réseaux informatiques

2.3.4 Les supports de transmission

Les supports de transmission ont pour caractéristique leur hétérogénéité, aussi bien du point de vue de transfert de données (Les données circulante prennent la forme d'impulsions électriques,ou bien le forme d'ondes électromagnétiques...) qu'au point de vue du type de support (paires torsadées, câble coaxial,ondes radio, fibre optique, ...)(4).

2.3.4.1 Les câbles à paires torsadées

Ce type de câble est utilisé pour la majorité des réseaux Ethernet récents, et surtout, pour les communications téléphoniques. Une paire de fils constitue un circuit pouvant transmettre des données. Les paires sont torsadées dans le but d'empêcher la diaphonie, c'est-à-dire le bruit crée par les paires adjacentes. Deux types de pair torsadée peuvent exister :

STP ou shielded twisted pairs : qui signifie paire torsadée blindée, il lie les techniques de blindage, de torsion des fils et d'annulation. Les paires de fils est entourée par une feuille métallique dont le but est de protéger les fils contre les bruits a l'intérieur comme à l'extérieur du câble. Par la suite, les quatre paires seront elles-mêmes enveloppées dans une feuille métallique ou une tresse.

UTP ou Unshielded twisted pairs : plus connu sous le nom de paire torsadée non blindée. Ce type de câble est utilisé dans des réseaux différents. Son but étant de limiter

la dégradation du signal provoquée par les interférences électromagnétiques et radio-fréquences, ce cable compte sur l'effet d'annulation de produit par paires torsadées. Le câble UTP est employé pour les réseaux Ethernet(4).

2.3.4.2 Les cables à fibre optique

Le câblage à fibres optiques utilise le verre en guise de support pour faire passer de la lumière, en faisant varier cette lumière dans le temps pour coder les 0 et les 1. L'utilisation du verre peut sembler particulier, étant donné que la majorité d'entre nous pensent au verre des fenêtres. Le verre des fenêtres est dur, rigide, et si vous le frappez ou le pliez suffisamment, il se brisera - toutes ces propriétés ne conviennent pas à un matériau de câblage(4).
Cependant, les câbles à fibres optiques utilisent de la fibre de verre, qui aide le fabricant à filer une longue et fine corde (fibre) de verre très flexible. Un câble à fibre optique aide à maintenir la fibre au milieu du câble, permettant à la lumière de traverser le verre – ce qui est très important pour la transmission de données. Malgré le fait que l'envoi de données par une fibre de verre fonctionne facilement, la fibre de verre elle-même nécessite de l'aide et du maintien. Le verre étant fragile, la fibre de verre se doit d'être protégée et renforcée(4).

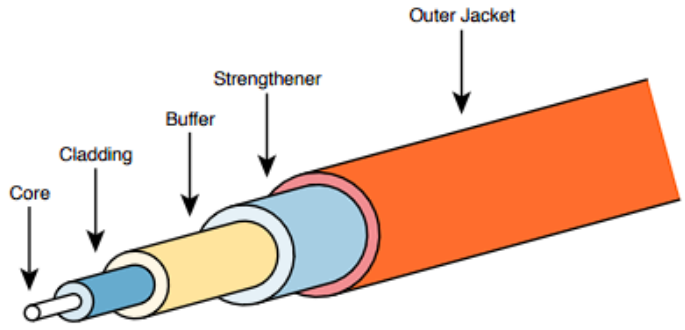


FIGURE 2.5 – Les composants d'un câble à fibre optique

Les trois couches qui se trouvent à l'extérieures du câble servent à protéger l'intérieur du câble et les rendent faciles à gérer et à installer. Ils permettent aussi la gestion et l'installation des câbles, tandis que la gaine intérieure(inner cladding) et le coeur collaborent pour créer un environnement qui permet la transmission de la lumière à travers le câble. Une source de lumière, nommée émetteur optique, renvoie une lumière dans le coeur.

La lumière peut traverser le cœur mais, elle est réfléchie par la gaine et retourne dans le coeur.

On distingue 2 types de fibre optique :

2.3.4.2.a fibre monomode(4) Les rayons lumineux peuvent suivre un seul chemin dans la transmission monomode. La finesse du cœur de fibre induit un chemin de propagation des modes quasi direct. Le signal devient peu déformé et cela du fait que la dispersion du signal sois nulle. Ses performances sont estimés a 100 gigabits/km, l'indice de réfraction peut être décroissant ou constant. Cette fibre est choisie pour les sites à très grande distance, et cela en raison de ses débit tres importants.

FIGURE 2.6 – Transmission sur une fibre monomode

2.3.4.2.b fibre multimode(4) Dans la transmission en multimode, les rayons lumineux sont capables de suivre des trajets différents en suivant l'angle de réfraction. Un certaine dispersion du signal peut être produite car les rayons arrivent au bout de la ligne à des instants différents. La fibre Multimode a pour émetteur une LED, et est en général utilisée pour les petites distances(de l'ordre de la centaine de mètres) et des performances de 1 gigabits/km. C'est la plus utilisée pour les réseaux privés.

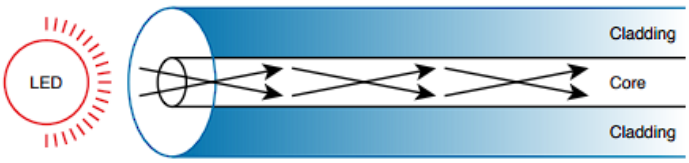


FIGURE 2.7 – Transmission sur une fibre multimode

2.3.5 Le modèle OSI

Afin de faciliter l'interconnexion des systèmes ouverts, un modèle OSI (Open Systems Interconnection) est défini par l'International Standards Organization. Le modèle OSI répartit les protocoles employés selon sept couches, définissant alors un langage commun pour le monde de l'information et de la technologie (IT). Pour la majorité des systèmes de traitement de l'information, aujourd'hui, il forme le socle de références. Chaque couche contient des dispositifs matériels (au sein des couches basses) ou logiciels (au sein des couches hautes). Nous pouvons définir des interfaces sous forme primitives de service et d'unité de données entre les couhces consécutives. Celles-ci rassemblent les information de contrôle rajoutées ainsi que les informations à transmettre.

- **Couche 1 : physique :** Les moyens mécaniques , électriques, optiques sont tous détenus dans la couche physique. Les unités de données sont représentés en bits (0 ou 1).
- **Couche 2 : liaison :** La fiabilité du transfert de bits d'un noeud du réseau à l'autre est gérée par la couche de liaison. Cette couche comprend les dispositifs de correction et de détéction d'erreus. L'unité de données à ce niveau est nommée trame.
- **Couche 3 : réseau :** Un réseau à commutation est utilisé par la couche réseau dans le but d'aiguiller les données.L'unité de données est appelée en général un paquet.
- **Couche 4 : transport :** Dans la couche de transport, nous pouvons trouver toutes les règles de fonctionnement . Cela peut assurer la transparence du réseau vis-à-vis des couches supérieurs. Elle traite principalement l'adressage, l'établissement des connexions ainsi que la fiabilité du transport.
- **Couche 5 : session :** Les procédures de dialogue entre les applications sont contenues dans la couche session : interruption et établissement de la communication, synchronisation et cohérence et des opérations.
- **Couche 6 : présentation :** Les formes de réprésentation de données sont traitées par la couche de représentation. Cette couche permet alors la traduction entre les différentes machines.
- **Couche 7 : application :** Il s'agit de la source et la destination de toutes les informations à transporter, la couche application comprend toutes les applications

éprouvant le besoin de communiquer par le réseau : transfert de fichiers messagerie électronique, gestionnaire de bases de données, etc.

2.3.6 Le modèle TCP/IP

Le modèle TCP/IP définit et fait référence à une large collection de protocoles qui permettent aux ordinateurs de communiquer. Pour définir un protocole, le protocole TCP/IP utilise des documents appelés Requests for comments (RFC) ou requêtes pour commentaires. Le modèle TCP/IP évite également de répéter le travail déjà effectué par un autre organisme de normalisation ou un consortium de fournisseurs en faisant simplement référence aux normes ou aux protocoles créés par ces groupes(4).

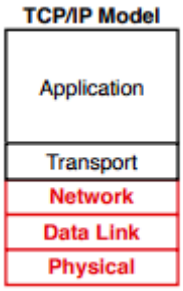


FIGURE 2.8 – Les couches du modèle TCP/IP

2.3.6.1 La couche application du modèle TCP/IP

Les protocoles de la couche application du TCP/IP fournissent des services aux logiciels d'applications exécutés sur un ordinateur. La couche application ne définit pas l'application elle-même, mais elle définit les services dont les applications ont besoin. Par exemple, le protocole d'application HTTP définit la manière dont les navigateurs Web peuvent extraire le contenu d'une page Web d'un serveur Web. En résumé, la couche application fournit une interface entre le logiciel exécuté sur un ordinateur et le réseau lui-même.

2.3.6.2 La couche transport du modèle TCP/IP

Bien qu'il existe de nombreux protocoles de la couche application du TCP/IP, la couche transport du TCP/IP comprend un plus petit nombre de protocoles. Le protocole de contrôle de transmission (TCP), ainsi que le protocole de datagramme utilisateur (UDP) sont les protocoles de couche de transport les plus couramment utilisés.

2.3.6.3 La couche réseau du modèle TCP/IP

La couche réseau TCP/IP comprend un petit nombre de protocoles, mais un seul protocole majeur : le protocole Internet (IP). En fait, le nom TCP/IP est simplement le nom des deux protocoles les plus courants (TCP et IP) séparés par un /.
l'IP offre plusieurs fonctionnalités, dont les plus importantes sont l'adressage et le routage.

2.3.6.4 Les couches liaison de données et physique du modèle TCP/IP

Les protocoles et le matériel nécessaires à la transmission des données sur un réseau physique sont définis par la couche de liaison de données physique du modèle TCP/IP. Les deux travaillent en étroite collaboration; en fait, certaines normes définissent à la fois les fonctions de la couche physique et la couche liaison de données.
La couche physique définit le câblage et l'énergie (par exemple, les signaux électriques) qui circulent sur les câbles.
Nous pouvons rencontrer certaines règles et conventions lors de l'envoi de données sur le câble, ces règles peuvent se trouver dans la couche liaison de données du modèle TCP/IP. Comme dans toutes les couches de tout modèle de réseau, la couche liaison de données TCP/IP fournit des services à la couche située au-dessus d'elle dans le modèle (la couche réseau).
Lorsque le processus IP d'un hôte ou d'un routeur choisit d'envoyer un paquet IP à un autre routeur ou hôte, cet hôte ou routeur utilise alors les détails de la couche liaison pour envoyer ce paquet à l'hôte/routeur suivant.

2.3.7 Comparaison entre le modèle TCP/IP et OSI

Le modèle OSI présente de nombreuses similitudes avec le modèle TCP/IP d'un point de vue conceptuel. Ils comportent des couches, et chaque couche définit un ensemble de fonctions réseau typiques. Comme pour le modèle TCP/ IP, les couches OSI font chacune référence à de multiples protocoles et normes qui mettent en œuvre les fonctions spécifiées par chaque couche. Dans d'autres cas, tout comme pour TCP/IP, les comités OSI n'ont pas créé de nouveaux protocoles ou de nouvelles normes, mais font référence à d'autres protocoles déjà définis. Par exemple, l'IEEE définit les normes Ethernet, les comités OSI n'ont donc pas perdu de temps à spécifier un nouveau type d'Ethernet, ils ont simplement fait référence aux normes Ethernet de l'IEEE.

Notez que le modèle TCP/IP utilisé aujourd'hui, utilise exactement les mêmes noms de couches que le modèle OSI pour les couches inférieures. Les fonctions correspondent généralement aussi, de sorte que, pour discuter de la mise en réseau et lire la documentation correspondante, considérez les quatre couches inférieures comme équivalentes, en termes de nom, de nombre et de signification. Même si le monde utilise aujourd'hui le protocole TCP/IP plutôt que le modèle OSI, nous avons tendance à utiliser la numérotation de la couche OSI. Par exemple, lorsqu'on se réfère à un protocole de couche application dans un réseau TCP/IP le monde se réfère toujours au protocole comme à un "protocole de couche 7".

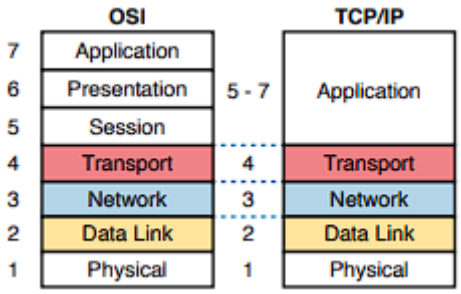


FIGURE 2.9 – Comparaison entre le modèle OSI et le modèle TCP/IP

2.3.8 Les éléments physiques d'un réseau

2.3.8.1 Le hub ou concetrateur

Dans le but d'accéder au réseau, le hub est un équipement permettant de connecter un bon nombre d'hôtes ensembles, il peut disposer d'un certain nombre de ports. Il va servir à récupérer les données qui arrivent sur un de ses ports pour alors les diffuser sur les autres ports.

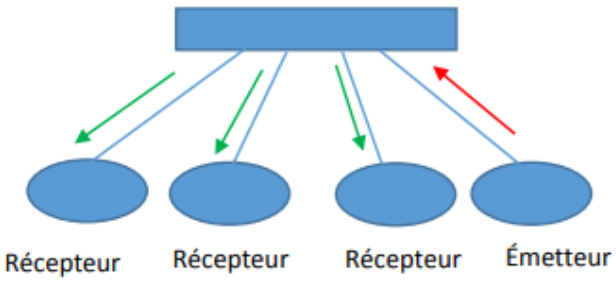


FIGURE 2.10 – Fonctionnement d'un Hub

2.3.8.2 Le switch ou commutateur

Etant un dispositif à grande vitesse, le commutateur reçoit un grand nombre de données, les transmettants par la suite vers leur destination sur un réseau local (LAN). Il peut prendre en charge tous types de protocoles de paquets car il fonctionne principalement au niveau de la couche liaison de données, ou la couche réseau du modèle OSI. La fonction du commutateur de couche 2 est d'envoyer des trames qui contiennent des paquets de données entre les segments ou les nœuds d'un réseau.
Principalement, les commutateurs sont les policiers du trafic d'un réseau local simple. La commutation sert à établir la trajectoire des trames, qui sont les unités de données, et la manière dont les données se déplacent d'une zone à l'autre du réseau.

2.3.8.3 Le routeur

Un routeur est défini comme étant un dispositif qui relie plusieurs réseaux ou sous-réseaux à commutation de paquets. Ses deux fonctions essentielles sont : gérer le trafic entre ses réseaux en acheminant les données vers les adresses IP de leurs déstitaions, et aussi, permettre l'utilisation d'une même connexion Internet par des périphériques différentes. On peut trouver plusieurs types de routeurs, mais la majorité d'entre eux transmettent les données entre les réseaux étendus (WAN) et les résaux locaux (LAN).

2.3.8.4 Le fonctionnement d'un routeur

Un routeur peut être assimilé à un contrôleur aérien et les paquets de données à des avions qui se dirigent vers différents aéroports (ou réseaux). Tout comme chaque avion possède une destination unique et suit un itinéraire spécifique, chaque paquet doit être guidé vers sa destination aussi efficacement que possible. De la même façon qu'un contrôleur aérien fait en sorte à ce que les avions atteignent leur destination sans subir des perturbation majeurs ou sans se perdre en cours de route, un routeur contribue à diriger les paquets de données vers leur adresse IP de destination.

Dans le but de diriger efficacement les paquets, un routeur met en œuvre une table de routage interne, c'est-à-dire une liste de chemins vers plusieurs destinations du réseau. Le routeur lit l'en-tête d'un paquet pour la détermination de sa destination, puis consulte la table de routage pour trouver le chemin le plus optimal vers cette destination. Il transmet, par la suite le paquet au réseau suivant sur le chemin.

2.3.9 Les techniques de routage

Le routage IP est le processus dont le principe est d'envoyer des paquets d'un hôte sur un réseau à un autre hôte sur un réseau distant et différent. Ce processus est en général effectué par des routeurs. Les routeurs examinent d'abord l'adresse IP de destination d'un paquet, déterminent par la suite l'adresse du prochain saut et transmettent le paquet. Les routeurs utilisent des tables de routage afin de déterminer l'adresse du prochain saut vers laquelle le paquet devra être acheminé.
Les routeurs se réfèrent aux tables de routage internes pour choisir l'acheminement des paquets sur les différents chemins du réseau. Une table de routage enregistre les routes que les paquets se doivent emprunter afin d'atteindre chaque destination dont le routeur est responsable.
Dans les réseaux, un protocole est une manière normalisée de formater les données de façon que tout ordinateur connecté puisse facilement les comprendre. Un protocole de routage est un protocole employé pour identifier ou annoncer les chemins du réseau.

Les protocoles qui suivent aident les paquets de données à trouver leur chemin sur Internet :

IP : Le protocole Internet (IP) identifie l'origine et la destination des paquets de données. Les routeurs inspectent l'en-tête IP des paquets pour choisir où les envoyer.

BGP : Le protocole de routage BGP (Border Gateway Protocol) est employé pour annoncer quels réseaux peuvent contrôler quelles adresses IP, et quels réseaux se connectent les uns aux autres. (Les grands réseaux qui font ces annonces BGP sont nommés systèmes autonomes). BGP est un protocole de routage dit dynamique.

OSPF : Le protocole Open Shortest Path First (OSPF) est souvent utilisé par les routeurs de réseau pour choisir dynamiquement les routes disponibles de façon à atteindre la destination finale avec un coût du chemin le plus faible possible.

RIP : Le protocole d'information de routage RIP emploi le "nombre de sauts" pour choisir le chemin le plus court d'un réseau à un autre, où le "nombre de sauts" signifie le nombre de routeurs qu'un paquet à l'obligation de traverser sur son chemin.

2.3.9.1 OSPF

Les protocoles de routage échangent des informations pour que les routeurs puissent apprendre des routes. Les routeurs apprennent des informations sur les sous-réseaux, les routes vers ces sous-réseaux ainsi que des informations métriques sur la

qualité de chaque route par rapport aux autres. Le protocole de routage peut donc choisir la meilleure route actuelle allant vers chaque sous-réseau, en construisant la table de routage IP. Les protocoles dit à état de liens comme OSPF adoptent souvent une approche différente des particularités des informations qu'ils échangent et de ce que les routeurs font de ces informations une fois qu'elles ont été apprises.

Les protocoles à états de liens construisent les routes IP en deux étapes principales. En premiers lieu, les routeurs construisent ensemble un bon nombre d'informations sur le réseau : routeurs, liens, information d'état, adresses IP, etc. Par la suite, les routeurs diffusent ces informations, de façon a ce que tous les routeurs connaissent les mêmes informations. A ce niveau, chaque routeur peut calculer les routes vers tous les sous-réseaux, mais de son propre point de vue.
L'OSPF génère des données qui permet à chaque routeur d'écouter ses voisins dans le réseau. Ces données sont alors utilisées afin de créer une carte topologique contenant tous les chemins disponibles dans le réseau. Cette base de données est sauvegardée pour être employée , et nous la nommons Base de données d'état de liaison.

Une fois que la base de données d'état de lien est établie, elle est utilisée dans le but de calculer le chemin le plus court vers les réseaux en employant un algorithme connu sous le nom de Shortest Path First. L'OSPF crée 3 tables :

La table de routage : Celle-ci contient les meilleurs chemins en cours de fonctionnement utilisés pour transmettre le trafic entre deux voisins.

La table des voisins : Elle englobe tous les voisins du type Open Short Path First découverts.

Table de topologie : Elle contient la carte routière du réseau. Celle-ci comprend la totalité des routeurs Open Short Path First disponibles en conservant les données calculées sur les meilleurs chemins et les chemins alternatifs.

2.3.10 Les VLAN

Afin de comprendre le VLAN, nous devons en premier lieu avoir une compréhension spécifique de la définition d'un réseau local ou LAN. Un réseau local contient tous les périphériques utilisateurs, les commutateurs, les serveurs, les routeurs, les câbles et les points d'accès sans fil dans le même endroit. Mais, une autre définition plus spécifique d'un réseau local peut contribuer à comprendre le concept de réseau local virtuel : Un réseau local englobe tous les dispositifs du même domaine de diffusion.

Un domaine de diffusion contient la totalité de tous les périphériques connectés au réseau local, de façon que quand l'un des périphériques envoie une trame de diffusion, tous les autres périphériques reçoivent une copie de la trame. Alors, d'un point de vue perspective, nous pouvons considérer qu'un réseau local et un domaine de diffusion sont finalement la même chose(4). En utilisant uniquement les paramètres par défaut, un commutateur considère que toutes ses interfaces se trouvent dans le même domaine de diffusion. En d'autres termes, pour un commutateur, lorsqu'une trame de diffusion entre dans un port du commutateur, le commutateur transmet cette trame de diffusion à tous les autres ports. Avec cette logique, pour créer deux LAN différents, Il faut acheter deux comumutateurs différents, comme l'illustre la figure suivante :

FIGURE 2.11 – Deux domaines de diffusions avec deux switchs différents

En utilisant deux VLAN, un seul commutateur a la possibilité d'atteindre les mêmes objectifs que la conception de la figure vue précédemment, à savoir la création de deux domaines de diffusion, avec un seul commutateur commun. Avec les VLAN, un commutateur a la possibilité de configurer certaines interfaces dans un domaine de diffusion et d'autres dans un autre, créant alors un grand nombre de domaines de diffusion. Nous pouvons appeler ces domaines de diffusion individuels créés par le commutateur les réseaux locaux virtuels (VLAN). Par exemple, dans la figure suivante, le commutateur unique crée deux VLAN, en traitant les ports de chaque VLAN comme étant totalement séparés. Le commutateur ne transmettra pas une trame envoyée par un appareil dans le VLAN 1 vers un autre dans le VLAN 2.

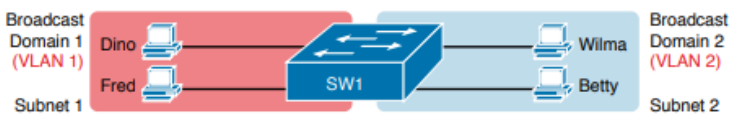


FIGURE 2.12 – Deux domaines de diffusions avec un seul switch

La création de réseaux locaux d'entreprise pour utiliser plus de VLAN, chacun avec un petit nombre de périphériques, permet d'améliorer le réseau local de nombreuses façons. A titre d'exemple, une diffusion envoyée par un hôte dans un VLAN

sera traitée par tous les autres hôtes du VLAN, mais pas par les hôtes d'un autre VLAN. En diminuant le nombre d'hôtes qui reçoivent une seule trame de diffusion, on réduit les hôtes qui gaspillent leurs efforts à traiter des diffusions superflues . Cela réduit aussi les risques de sécurité parce que moins d'hôtes voient les trames envoyées par le même hôte. Ce ne sont là que certaines raisons de séparer les hôtes en différents VLAN.

2.4 Généralités sur les télécommunications

Le terme "télécommunications" signifie l'ensemble des moyens techniques qui permettent l'acheminement fidèle d'informations entre deux points quelconques pour un coût raisonnable. Les télécommunications emploient deux techniques inséparables : la transmission a pour fonction le transport de l'information à distance ; la mise en relation de deux usagers quelconques conformément à leurs ordres relève de la commutation. Les systèmes de télécommunications et principalement les systèmes de commutation ont utilisé les techniques informatiques, tant logicielles que matérielles, dès le courant des années 60. L'utilisation de ces techniques a été engagée quasi simultanément avec celle de l'électronique. L'utilisation de telles techniques se fait à plusieurs niveaux :

— au niveau des services, par l'initiation de la numérisation et de techniques de transport et de traitement de données , aussi bien pour l'information donnée par les usagers que pour celle nécessaire à la commande du réseau (signalisation).

— au niveau des réseaux, par l'emploi d'ordinateurs programmés dans le but de commander les nœuds de ces réseaux. Cette utilisation met en évidence l'importance du logiciel, ce qui reflète l'« intelligence » du réseau et la qualité des services qu'il peut procurer.

— au niveau des techniques de développement de ces systèmes programmés, l'informatique peut servir d'outil pour son propre développement.

2.4.1 Global System for Mobile Communication

La GSM ou plus connue sous le nom de la 2G ou deuxième génération de réseau cellulaire est l'intitulé d'un groupe de normalisation établi en 1982 afin créer une norme européenne commune en matière de téléphonie mobile qui formulerait les spécifica-

tions d'un système européen de radiocommunication cellulaire mobile qui fonctionne à 900 MHz.

2.4.1.1 L'architecture des réseaux GSM

L'architecture d'un réseau GSM est divisée en trois sous-systèmes : 1. Le sous-système radio qui contient la station mobile, la station de base ainsi que son contrôleur. 2. Le sous-système réseau ou dit d'acheminement. 3. Le sous-système opérationnel ou d'exploitation et de maintenance.

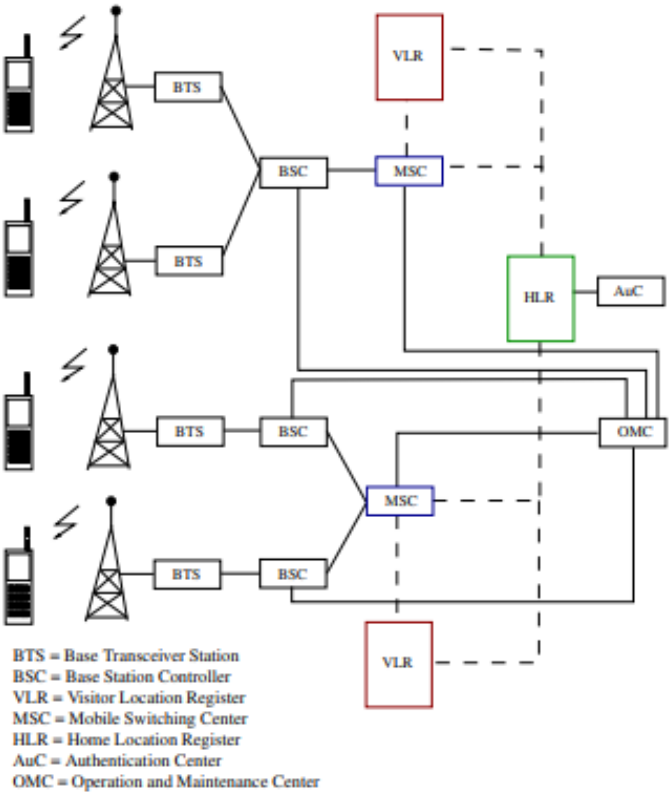


FIGURE 2.13 – L'architecture des réseaux GSM

2.4.1.2 La station de base BTS(26)

La station de base est l'élément central, on pourrait la définir comme étant un ensemble émetteur/récepteur qui pilotent une ou un bon nombre de cellules. Dans le réseau GSM, chaque cellule principale au centre de laquelle se situe une station base pourrait être divisée (à l'aide des antennes directionnelles) en des cellules plus petites qui sont des portions de celle de départ et qui utilisent des fréquences porteuses différentes. La puissance d'émission des BTS détermine la taille d'une cellule. Chaque BTS a entre 1 et 16 émetteurs-récepteurs, et ce, en fonction de la densité d'utilisateurs au sein de la cellule. Chaque BTS est considéré comme étant une cellule unique. Elle assure également les fonctions suivantes :

— Cryptage, multiplexage, codage, modulation et transmission des signaux RF à l'antenne.

— Adaptation et transcodage du débit

— Synchronisation de la fréquence et du temps.

— Voix via de services à plein débit ou à demi-débit

— Décryptage, égalisation et décodage des signaux reçus.

— Détection et détermination d'accès aléatoire

— Avancées temporelles

— Mesures des canaux de liaison montante

2.4.1.3 Le contrôleur de station de base BSC(26)

Le BSC a pour fonction de gérer les ressources radio pour une ou plusieurs BTS. Il contribue à gérer la configuration des canaux radio, les sauts de fréquence ainsi que les transferts. La BSC traduit le canal vocal de 13 Kbps utilisé sur la liaison radio en canal standard de 64 Kbps employée par le réseau téléphonique public commuté . La BSC gère aussi le transfert intercellulaire. La fonction principale de la BSC est d'allouer les créneaux horaires nécessaires entre la BTS et le MSC. C'est un dispositif de commutation qui gère principalement les ressources radio. Les fonctions supplémentaires englobent :

— Le contrôle des sauts de fréquence

— La concentration du trafic afin de réduire le nombre de lignes en provenance du MSC.

— La procuration d'une interface au centre d'exploitation et de maintenance pour le BSS.

— La réaffectation des fréquences entre les BTS

— La synchronisation de la fréquence et du temps

— La gestion de l'alimentation

— La mesures du temps de propagation des signaux reçus de la MS

2.4.1.4 L'interface Abis

L'interface Abis a pour fonction de relier le BSC à la station de base(BTS).
Le BTS et le BSC interagissent à travers l'interface Abis spécifiée, ce qui facilite les opérations entre des composants fabriqués par des fournisseurs différents.

2.4.2 Universal Mobile Telecommunications System

L'UMTS ou plus communément connue sous le terme 3G ou troisième génération de technologie de téléphonie mobile.
L'UMTS est un système cellulaire numérique de communication avec des mobiles ou entre mobiles, destiné essentiellement à offrir une vaste gamme de services de voix, de données et d'images, ainsi que l'accès à l'internet.
L'UMTS est doté d'une architecture de système complète et comme pour le GSM, l'accent est mis sur les interfaces normalisées.

Le réseau UMTS vont relayer les paquets IP entre les stations mobiles et les stations mobiles et les réseaux externes. Les stations mobiles se voient attribuer une nouvelle adresse IP lorsqu'elles invoquent une session de données, nommée PDP (Packet Data Protocol). A partir de la, la station mobile pourra transmettre et recevoir des paquets IP à des réseaux externes.

2.4.2.1 L'architecture des réseaux UMTS

L'UMTS emploie une architecture fondamentale pour les services de voix et de données semblable a la GSM. Les opérateurs ont la possibilité d'utiliser le même réseau central (composé a partir des centres de commutation mobile et des nœuds de données par paquets) à la fois pour le réseau d'accès radio GSM/GPRS/EDGE ainsi que pour l'UMTS.

Les mêmes centres de commutation mobile sont employés pour la GSM et l'UMTS.

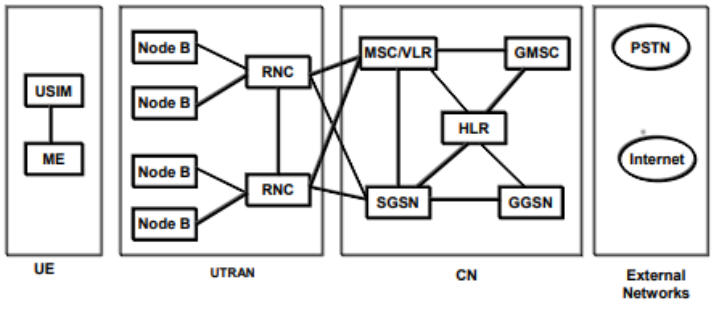


FIGURE 2.14 – L'architecture des réseaux UMTS

2.4.2.2 La node B

Le node B est un terme qui désigne un récepteur de station de base radio. Sa fonction est d'assurer la couverture radio et de convertir les données entre le réseau radio et les RNC.
Nous pouvons décrire la fonctionnalité du node B comme étant une procédure de mise en correspondance, entre les canaux logiques des couches supérieures (à partir de la couche 2) et les canaux physiques (couche 1). Dans la liaison descendante, les trames de données des canaux de transport des couches supérieures sont au départ codées, puis groupées et modulées avant d'être transmises par l'antenne. Dans la liaison montante, les canaux physiques sont au départ démodulés, puis décodés et mappés sur les trames de données de la couche supérieure. Les canaux de transport comprennent des canaux de type dédié et de type commun. Les canaux dédiés sont consacrés à un unique utilisateur, tandis que les canaux communs sont partagés par plusieurs. Plusieurs canaux de transport peuvent être dédiés à un seul utilisateur. Ils sont multiplexés en un canal de transport composite codé. Plusieurs canaux de transport peuvent être attribués à un service et/ou à plusieurs services fonctionnant au même temps. A titre d'exemple, un utilisateur a la possibilité télécharger des e-mails en arrière-plan et cela en parlant au téléphone en même temps.

2.4.2.3 Le RNC

Le RNC ou Radio Network Controller est défini comme étant un élément qui dirige un réseau d'accès radio UMTS et est chargé de contrôler les nodes B qui y sont connectés. Le RNC assure la gestion des ressources radio , quelques-unes des fonctions de gestion de la mobilité et est le point où le cryptage est effectué avant que les données de l'utilisateur ne soient envoyées vers et a partir du mobile. Le RNC se connecte au réseau central à commutation de circuits à travers la passerelle média (Media Gateway) et au SGSN (nœud de support GPRS) dans le réseau central à commutation de paquets.
Le RNC assure l'établissement des liaisons radio physiques via des ressources d'accès radio appelées communément supports d'accès radio sur les demandes de service des utilisateurs. Le RNC gère aussi le contrôle de l'encombrement et l'équilibrage de la charge des canaux alloués.

2.4.2.4 L'interface IuB

L'interface Iub se situe sur un réseau UMTS et se trouve entre le RNC (Radio Network Controller) et le Node B.
Les principes de la spécification de l'interface IuB sont les suivants :

- Le partage de la transmission faite entre l'interface GSM/GPRS Abis et l'interface Iub ne doit absolument pas être exclu.
- La division fonctionnelle entre le RNC et le nœud B ne doit comporter que les options nécessaires.
- L'interface Iub se doit d'être basée sur un modèle logique du nœud B.
- Le nœud B contrôle un bon nombre de cellules et nous pouvons lui ordonner d'ajouter ou de supprimer des liaisons radio dans ces cellules.
- La structure physique et les protocoles internes du nœud B ne doivent pas être visibles sur l'Iub et ne sont donc pas des facteurs limitatifs.
- Seule la maintenance logique et l'exploitation du nœud B sont prises en charge par l'Iub.
- Les fonctionnalités complexes doivent être évitées sur l'Iub (Dans la mesure du possible). Plusieurs solutions d'optimisation avancées pourront être ajoutées dans les versions ultérieures de la norme.
- La répartition fonctionnelle de l'Iub tient en compte de la possibilité d'une commutation fréquente entre différents types de canaux.

2.4.3 Long Term Evolution

Long Term Evolution (LTE), plus connue sous le nom de 4G, est la nouvelle norme pour le haut débit de la sécurité publique à l'échelle du pays. Cette norme permet d'accéder aux technologies numériques et d'offrir des capacités étendues aux praticiens de la sécurité publique sur le terrain. Le LTE a pour fonction de permettre de faire entrer la sécurité publique dans l'ère numérique. Les nouveaux appareils et les applications technologiques sortantes chaque jour sont plus performantes et rivalisent avec ceux qui ne pouvaient fonctionner que sur des serveurs et des ordinateurs de bureau il y a quelques années à peine. Cette technologie aura pour vocation de favoriser le développement d'applications adéquate à la sécurité générale et contribuera à rendre les opérations des premiers intervenants plus efficientes et efficaces.

La norme LTE prend des vitesses rapides et est prometteuse, cependant, il existe des limites à l'utilisation de la technologie associée dans le domaine.

2.4.3.1 L'architecture des réseaux 4G (LTE)

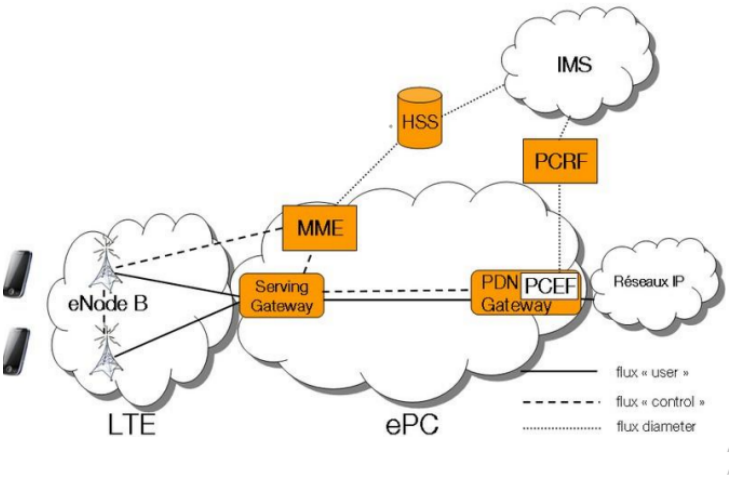


FIGURE 2.15 – L'architecture des réseaux LTE

L'architecture de réseau de haut niveau du LTE est constituée des trois éléments essentiels suivants :

— L'équipement de l'utilisateur (UE).

— Le réseau dédié à l'accès radio terrestre UMTS évolué (E-UTRAN).

— Le noyau de paquets évolué(Evolved Packet Core).

Le noyau de paquets évolué est en communication avec les réseaux de données par paquets du monde extérieur, comme l'Internet, les réseaux privés d'entreprise ou les sous-systèmes multimédia IP.

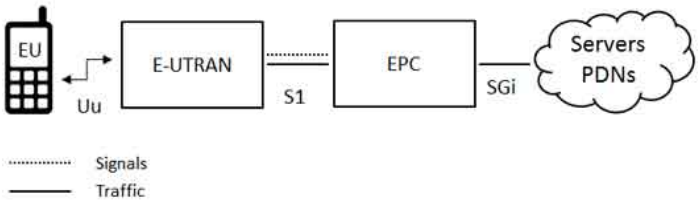


FIGURE 2.16 – Les interfaces du réseau LTE

2.4.3.2 L'eNode-B

L'E-UTRAN s'occupe de gérer les communications radio qui se passent entre le mobile et le noyau de paquets évolué et ne contient qu'un seul composant principal, les stations de base évoluées, connues sous le nom de eNodeB ou eNB. Chaque eNB est définie comme étant une station de base dont la fonction est de contrôler les mobiles dans une ou plusieurs cellules. La station de base qui communique avec un mobile est connue sous le nom de eNB de service.
Les mobiles LTE communique seulement avec une unique station de base et une seule cellule à la fois.
L'eNB reçoit et envoie des transmissions radio à tous les mobiles grâce à l'utilisation des fonctions de traitement des signaux analogiques et numériques de l'interface radio LTE.
L'eNB contrôle également le fonctionnement de bas niveau de tous ses mobiles, et cela, en leur envoyant des messages de signalisation comme des commandes de transfert.
Tous les eNB se connectent à l'EPC à l'aide de l'interface S1 et peuvent aussi être connectés aux stations de base voisines par l'interface X2, qui est essentiellement utilisée pour la signalisation et le transert de paquets.

2.4.3.3 Les interfaces S1/X2

L'interface dite S1X2 est définie entre l'eNodeB LTE et la gateway LTE. Elle procure une livraison de données non garantie des unités de données de protocole (PDU) du plan utilisateur LTE entre l'eNodeB et la gateway.
La couche réseau de transport est construite sur les bases du transport IP.
L'interface S1X2 a pour fonction la livraison des données utilisateur qui est située entre l'eNodeB et la gatewat. Le marquage IP DSCP (Differentiated Service Code Point) est pris en charge pour la QoS par support radio.

2.5 L'IP/MPLS

L'MPLS a été introduit par l'Internet Engineering Task Force (IETF) pour rendre l'Internet évolutif, rapide, adaptable aux nouveaux mécanismes de routage et gérable. L'MPLS utilise le TE(traffic engineering) pour partager la charge du réseau entre des liens à chemin d'accès inégal.

L'MPLS est une méthode de transmission des paquets IP qui utilise des étiquettes au lieu des adresses IP ou des en-têtes de paquets de couche 3. Il combine les meilleures caractéristiques des modèles Overlay et Peer-to-peer. Comme il utilise des étiquettes, il accélère la transmission des paquets et le routage IP. Il supprime la charge de traitement de l'inspection IP lors de l'utilisation d'une opération de routage normale.
L'MPLS est souvent appelée la technologie de la couche 2,5. Elle permet de construire facilement les routes explicites pour une source ou un service spécifique.
Dans un réseau IP traditonnel chaque routeur effectue une recherche d'adresse IP ("routage"), détermine le saut suivant à partir de sa table de routage et transmet le paquet à ce saut suivant. Le processus se répète pour chaque routeur, chacun prenant ses propres décisions de routage indépendantes, jusqu'à ce que la destination finale soit atteinte.
Dans un réseau MPLS le premier appareil fait une recherche de routage, comme avant, mais au lieu de trouver un next-hop, il trouve le routeur de destination finale puis il trouve un chemin prédéterminé pour y arriver. Le routeur applique un "label" (étiquette) basé sur cette information, ainsi les routeurs suivants utilisent cette étiquette pour acheminer le trafic sans avoir besoin d'effectuer des recherches d'IP supplémentaires.

2.5.1 l'en-tête MPLS

L'en-tête MPLS est insérée entre les en-têtes de la couche 2 et la couche 3 du modèle TCP/IP.
Un en-tête MPLS de 32 bits se compose d'un champ d'étiquette, d'un champ expérimental, d'une pile et d'un champ de temps de vie.

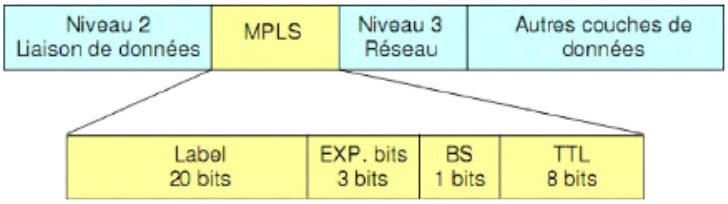


FIGURE 2.17 – L'en-tête MPLS

2.5.2 Le protocole LDP

Le protocole LDP est similaire à l'OSPF. Il s'exécute au-dessus d'une configuration de routage et nécessite d'être configuré sur toutes les interfaces des routeurs. Une fois que le LDP est configuré sur une interface, il commence à transmettre et à recevoir des messages.
Le protocole LDP envoie des messages de découverte à toutes les interfaces compatibles. Lorsqu'un routeur adjacent reçoit le message de découverte, il établit une session TCP avec le routeur source. Il peut également établir de nouveaux chemins à l'aide de messages LDP après une défaillance d'une liaison.

2.6 Les opérations de l'MPLS

Tout le trafic qui arrive dans le domaine MPLS entre au routeur d'entrée et sort au routeur de sortie. Au niveau du routeur d'entrée, une étiquette est attribuée à chaque paquet et le paquet est transféré à travers le domaine MPLS en fonction de cette étiquette. A chaque routeur, l'étiquette est échangée avec une autre étiquette appelée "label swapping", qui représente le prochain routeur. Enfin, lorsque le paquet atteint le routeur de sortie, il élimine l'étiquette du paquet et transmet le paquet en fonction de l'en-tête de la couche réseau.
Les différentes étapes du MPLS sont :
Les protocoles de routage échangent des informations de routage vers les réseaux de destination.
Le protocole de distribution d'étiquettes (Label Distribution Protocol) établit des mappages d'étiquettes vers le réseau de destination.
Le Routeur d'entrée reçoit les paquets et leur attribue une étiquette aux paquets en fonction du Forwarding Equivalence Class.
Le routeur à commutation d'étiquettes transmet les paquets en utilisant le label swapping.
Le routeur de sortie enlève l'étiquette et délivre le paquet en fonction de l'en-tête de la couche réseau.

2.7 L'architecture MPLS

L'architecture MPLS comprend les parties suivantes :

Plan de contrôle : génère et maintient les informations de routage et d'étiquetage.

Base d'informations de routage (RIB) : est générée par les protocoles de routage IP et utilisée pour sélectionner les routes.

Protocole de distribution d'étiquettes (LDP) : alloue les étiquettes, crée une base d'informations sur les étiquettes (LIB), et établit et détruit les LSP.

Base d'informations sur les étiquettes (LIB) : est générée par le protocole LDP et utilisée pour gérer les étiquettes.

Plan d'acheminement (plan de données) : achemine les paquets IP et les paquets MPLS.

Forwarding information base (FIB) : est générée sur la base des informations de routage obtenues à partir de la RIB et utilisée pour transmettre les paquets IP communs.

Base d'informations de transfert d'étiquettes (LFIB) : créée par LDP sur un abel switching router et utilisée pour transférer les paquets MPLS.

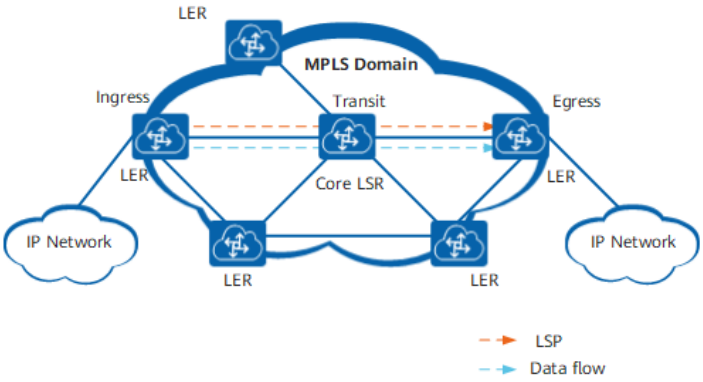


FIGURE 2.18 – L'architecture des réseaux MPLS

2.8 L'ingénierie du trafic

En raison de la popularité croissante des applications en temps réel, de nouveaux défis sont lancés à la communauté Internet. D'autre part, les fournisseurs de services ont besoin d'un outil particulier pour tirer le meilleur parti de leurs réseaux afin d'augmenter leurs revenus en répondant aux besoins des applications en temps réel ou critiques, car les différentes applications ont des besoins variables en matière de retard, de bande passante, de perte de paquets, etc. Les applications générant des revenus élevés

sont souvent critiques et dépendent fortement de la latence, la rapidité des données étant de la plus haute importance. Ainsi, sur Internet, où le flux de trafic est imprévisible, ces applications ont besoin d'un certain degré d'ingénierie du trafic pour fonctionner efficacement. L'ingénierie du trafic était nécessaire en raison de la nature aléatoire de l'Internet où la capacité est requise pour définir les routes dynamiquement, un résultat important de ce processus est l'évitement de la congestion sur n'importe quel chemin. La planification des engagements de ressources sur la base de la qualité de service requise qui comprend les demandes connues et l'utilisation optimisée du réseau est connue sous le nom d'ingénierie du trafic. Pourquoi elle est nécessaire ? Parce que le réseau IP conventionnel offre un mauvais support pour l'ingénierie du trafic car son protocole de base IP (Internet Protocol) n'a jamais été conçu avec la QOS à l'esprit, il a plutôt été conçu à des fins d'éducation et de recherche. L'important est de savoir comment allouer les ressources réseau disponibles afin d'optimiser les performances du réseau lorsque celui-ci doit supporter de fortes charges de trafic avec des ressources limitées. Le paradigme de l'acheminement basé sur la destination provoque des congestions dans les réseaux IP conventionnels car certains liens sont fortement congestionnés tandis que d'autres restent sous-utilisés, ce qui est un phénomène inévitable. Les protocoles de passerelle intérieure tels que les protocoles de routage OSPF et IS-IS utilisent le paradigme de la transmission basée sur la destination sans prendre en considération d'autres paramètres de réseau comme la bande passante disponible et le flux de trafic entre la source et la destination par le chemin le plus court. Il est donc évident que lorsque tout le trafic suit la même route, cela crée une situation de point chaud et de congestion, tandis que les autres liens du réseau restent inutilisés, ce qui entraîne une dégradation du débit, du retard et de la perte de paquets. Le MPLS est un élément clé de l'équilibrage efficace de la charge. possible de mapper des routes sur la base d'un flux individuel ou de différents flux entre les mêmes points d'extrémité. D'un point de vue pratique, les routes supportées par MPLS changent sur une base flux par flux lorsque la demande de trafic pour ce flux change. En ayant un contrôle sur les routes, il est possible d'optimiser les ressources du réseau et de mieux utiliser les liens par répartition de la charge et de supporter différents niveaux d'exigences de trafic(27).

2.9 Conclusion

Nous avons présenté dans ce chapitre l'état de l'art de notre projet de fin d'étude, ainsi que les concepts qui nous intéressent en réseaux informatiques et en télécommunications.
Nous verrons dans le prochain chapitre des concepts générales sur les bases de données, les interfaces Web ainsi que l'interaction entre les deux.

Chapitre 3

Les bases de données, les interfaces WEB et l'interaction entre les deux

3.1 Introduction

Dans le milieu de l'entreprise, il est indispensable d'établir des bases de données, tout simplement car elles facilitent le stockage et l'accès aux informations. Nous présenterons dans ce chapitre des concepts et généralités sur les bases de données, les interfaces Web ainsi que l'interaction entre les deux.

3.2 Les bases de données

Les bases de données (BDD) ont pris une place d'une grande importance dans le monde, principalemen celui de la gestion, ainsi que celui de l'informatique. Ces systèmes occupent une place centrale dans le développement des technologies. L'étude des bases de données a contribué de façon évidente au développement des algorithmes spécifique dont le but principal est de gérer les données(19).

Comme nous pouvons observer dans de multiples innovations technologiques, la création des Systèmes de Gestion de Bases de Donnée (SGBD) est causé par l'apparition d'une grande pression des besoins. Comme nous observons dans l'environnement informatique traditionnel des gros systèmes d'exploitation, l'unique mode de gestion de données est le gestionnaire de fichiers. Les données qui sont traitées par une application (gestion de la paie, des stocks, de la comptabilité...) sont spécifiques et destinés de façon unique à cette application.
L'organisation conventionnelle des données en fichiers traite des besoins spécifiques : les informations utilisées par les services sont le plus souvent distinctes pour des traitements qui différent completement les uns des autres .

C'est pour cette raison que toutes les équipe de l'entreprise ont des fichiers de donnéesqui leurs sont propres où ne figurent que les informations qui la concernent.

La forme de stockage de ces données dépend de facteurs variés et nombreux : disponibilité du matériel, harmonisation et surtout volonté des équipes dédiés au développement (choix de l'architecture de stockage, choix du langage de programmation...) ; principalement, les accès à l'information déterminent très souvent l'organisation qui a été fixée pour les données : les informations choisies qui seront regroupées dans le même fichier, l'organisation du fichier choisie (aléatoire, séquentiel,...).

3.2.1 La modélisation Entité-Association

Ce modèle, permet d'avoir une description naturelle du monde via des concepts dits d'entité et d'association. Fondé principalement sur la théorie des relations, ce modèle est universel et répond au but d'indépendance données-programmes.
Le principe du modèle entité-association est d'effectuer des représentations, par un schéma standard, les différents éléments qui construisent le système d'information, connus sous le nom de attributs (par exemple : l'adresse, le nom, l'âge, la profession ...), et les relations les reliant, nommées associations.

3.2.1.1 Le type entité

Un type-entité défini un ensemble de d'entités possédant une sémantique et propriétés communes.
Les livres, les commandes, les produits, les personnes, sont des type-entités. Par exemple, dans le cas d'une personne, les informations qui y sont associées (leurs propriétés) comme le nom, le prénom, leurs adresses, leurs numéros qui ne changent pas de nature en peu importe la personnes.
Une entité est définie comme étant une occurrence de son type-entité (ou son instance). Le terme entité peut désigner le type-entité et ses entités. Ces deux notions ne devraient pas être confondues.

3.2.1.2 Le type association

Un type relation (ou un type association) défini un ensemble de relations possédant des caractéristiques qui sont semblables. Le lien qui peut exister entre des types entité sont décrit par le type association.

3.2.1.3 Attribut et Valeur

Un attribut caractérise un type-association ou un type-entité.
Chaque attribut détient un domaine définissant l'ensemble des valeurs possibles pouvant être choisies pour lui (entier, booléen chaîne de caractères…). Dans l'entité, chaque attribut a une valeur qui se veut compatible avec son domaine.

3.2.1.4 MySQL(20)

MySQL est un outil qui gère les base de données.Il est open source. Sa fonction principale est de gérer de très grandes bases de données. MySQL est connu pour sa fiabilité et sa performance(20).
Il emploie un langage de requêtes bien structuré. Les bases de données relationnelles sont des bases de données qui emploient une structure qui nous permettent d'identifier et d'accéder aux données qui sont en relation avec un autre élément de données au sein de la base de données. Ce format est communément organisé sous forme de tables.

3.3 Les interfaces Web

Les interfaces web sont des interfaces hommes-machines dont les bases de construction sont les pages web. Ils permettent l'utilisation d'applications web dans quelques cas. Un navigateur web tel que Safari, Google Chrome, Firefox, Internet Explorer ou alors Opera sont installable et souvent présent sur tout les ordinateur, une interface web peut être visualisée via la totalité des dispositifs qui détiennent un navigateur web (smarphone, ordinateur, tablette etc...). Grace a cela, il suffit seulement d'avoir le bon URL, ainsi que les mots de passes essentiels afin d'accéder à n'importe quel type de donnée partout dans le monde à partir d'Internet

3.3.1 HTML

Le langage HTML (Hyper Text Markup Language) est connu comme étant un langage dont la fonction est de spécifier la façon dont le texte et les graphiques sont visibles sur une page web. Il décrit la structure des pages Web via des balises.
Les balises sont la représentation des éléments HTML, ceux-ci sont des blocs de construction des pages HTML.
Les balises HTML servent à étiqueter les éléments de contenu comme le "titre", "paragraphe", "tableau",...
Le but des navigateurs n'est pas d'afficher les balises HTML, mais plutôt de rendre le contenu de la page.
La page HTML est stockée sur l'ordinateur qui héberge le site web et la page est par la suite, envoyée à votre navigateur.

3.3.1.1 Quelques exemples de HTML tag

Il existe différentes sortes de tag HTML :

— Pour les paragraphes nous utilisons le "p" tag comme suit : Ceci est un paragraphe

— Pour les listes nous utilisons le tag "li" : <li>

— Pour les titres nous utilisons des "heading" donc "h" tag : selon la taille de <h1> à <h6> h1 représente la plus grande taille et h6 représente la plus petite.

— Il est également possible de diviser notre page en parties en fonction de leur contenu : <header> </header> pour désigner l'entête, <body> </body> pour désigner le contenu principal et <trailer> </trailer> pour le pied de la page.

— Il est également possible d'insérer des tableaux comme : <table><tr>Pour désigner les lignes</tr><th>pour désigner les titres des colonnes</th><td>Pour désigner le contenu du tableau(les données)</td> </table>

3.3.2 CSS

CSS est le langage qu'on utilise pour styliser un document HTML. Il décrit comment les éléments HTML sont affichés(28).

Le langage CSS rassemble les propriétés qui ifluencent sur la visualisation des éléments d'une page. Il subsiste un ensemble de valeurs possibles pour chaque propriété. Ces propriétés peuvent être fixé pour chaque éléments d'un document HTML. Les propriétés qui agissent sur l'apparence d'un block d'un élément concernent :

— L'apparence du contenu.

— La taille de la boîte.

— Le positionnement de la boîte.

Les règles CSS définissent la valeur que prend une propriétée dans un selecteur précis :

selecteur propriete : valeur

le sélecteur spécifie les règles appliquées sur les éléments.

3.3.3 Jinja

Jinja est connu comme étant un moteur de création de modèles qui se veut être expressif, rapide, et extensible. Nous pouvons trouver des espaces réservés dans un code HTML dans le modèle qui permettent d'écrire du code semblable à la syntaxe Python. Par la suite, le modèle reçoit des données afin de rendre le document final(13). Jinja 2 donne la possibilité aux développeurs de produir des pages Web du fait qu'il soit un moteur déstiné à Python. Ainsi, il contient du code HTML de base, et des espaces dédiés pour que Jinja 2 les remplisse. Du fait qu'il donne la possibilité aux développeurs d'utiliser des concepts puissants comme l'hérirage, il est considéré comme étant le système le plus utilisé. Il est basé sur le système de modéle Django.

Le framework web **Flask**, utilise des modèles Jinja par défaut. Il configure un environnement Jinja et un chargeur de modèles (render template) , et fournit des fonctions pour rendre (return) à nos fonctions des pages webs programmées au préalable grâce à « render template »

3.4 La relation entre les bases de données et les interfaces Web

Dans ce qui va suivre, nous étudierons la probabilité de lier une base de données (MySQL dans le cas présent) à notre interface Web HTML dans le but de visualiser le contenu de cette dernière et de mettre en place des requêtes (query) sur notre base de données.

3.4.1 Python

Le langage de programmation Python est un langage interprété qui est multiplateformes multi-paradigme.Il est doté d'un typage dynamique fort, Python est également équipé d'un système de gestion de la mémoire automatiquement par ramasse-miettes. Il met en place une programmation structurée, fonctionnelle et orientée objet. Python peut permettre une initiation trés simple aux concepts de la programmation grâce a sa syntaxe. Python s'adapte à tout type d'utilisation et peut s'utiliser dans de nombreux contextes et cela grâce a des **bibliothèques spécialisées**. Aussi, il est utilisé en guise de langage de script afin de rendre automatiques des tâches faciles mais

fastidieuses, et cela, dans le but d'éviter de répéter certains enchaînements d'actions répétitives.
Focalisons-nous dans ce cas à quelques bibliothèques Python spécifiques, en premier lieu la librairie **Flask** puis ultérierement dans le prochain chapitre, les librairies réseau : **Netmiko** , **Paramiko** et **Telnetlib**.

3.4.2 Flask

Flask est un farmework open-source de développement web en Python. Il figure parmi les microframework parce qu'il est particulièrement léger(8).
Le but de Flask est de garder un noyau simple et extensible. Plusieurs extension peuvent permettre d'ajouter quelques fontionnalités, telles que le système d'authentification, les couches d'abstraction de base de données et des outils de validation de formulaire.
Flask est dépendant du moteur de template Jinja. Après la définition dans des fichiers indépendants les pages (en HTML), nous pouvons désormais les lier à l'aide de Flask. A titre d'exemple si vous voulez entrer dans une page Web différente que l'actuelle, il est suffisant de cliquer sur "connecter" ou sur n'importe quel autre objet ou onglet qui pourrait nous y rediriger.
Pour effectuer cela, nous utilisons la fonction **"Render template"** de Flask.

3.4.2.1 Render template

Nous définissons les template comme etant de fichiers contenant des données statiques et des espaces réservés pour des données dites dynamiques. Flask emploie la bibliothèque de **Render template** Jinja pour rendre les templates.
Dans notre application, nous allons employer des templates pour rendre le HTML qui va s'afficher dans le navigateur de l'utilisateur. Dans Flask, la configuration de Jinja est faite de façon à ce que toutes les données qui sont rendues dans les modèles HTML soient autoescape. Cela implique que le rendu de l'entrée de l'utilisateur est sûr. Tout les caractères qu'il a saisit et qui pourraient probablement perturber le HTML, tel que < et >, seront échappés avec des valeurs sûres qui auront une apparence semblable dans le navigateur mais ne conduirons pas a des effets indésirables.
Jinja se présente et se comporte principalement tel que Python. Des délimiteurs spéciaux sont employés dans le but de distinguer la syntaxe Jinja des données statiques du

modèle. Tout ce que nous pouvons trouver entre et est une expression qui sera par la suite affichée dans le document final. Les blocs sont désignés par des balises de début et de fin contrairement à Python ou ils sont représentés par une indentation, parce que le texte statique à l'intérieur d'un bloc peut potentiellement modifier l'indentation.

3.4.2.2 La fonction request

Les données de request détiennent des informations que le client peut transmettre au serveur.
request.form va nous permettre l'obtention des réponses (entrées) de l'utilisateur de la page web à un formulaire (form) dans le but de les enregistrer ou les réutiliser.
request.args.get(key, default) - récupère les clés à partir des URLs qui détiennent : key=valeur

3.4.3 Flask-MySQLAlchemy(9)

Flask-SQLAlchemy est une sorte d'extension pour Flask qui peut servir à ajouter le support de SQLAlchemy à une application(9). Elle a pour but de faciliter l'utilisation de SQLAlchemy avec Flask en procurant des valeurs par défaut utiles plus facile la concrétisation des tâches courantes. L'extension FlaskMySQLAlchemy nous permet de connecter notre base de données à notre code Python.
De ce fait, nous pouvons connecter notre page web ainsi que notre base de données et nous pouvons réaliser le traitement que nous souhaitons à nos données pour enfin les afficher sur notre interface Web.

3.5 Conclusion

Nous développerons lors du prochain chapitre l'implémentation d'un réseau IP/MPLS sur nos émulateurs, tout en passant par les librairies réseau de Python. Nous veillerons aussi à tester les performances de notre réseau IP/MPLS par rapport à un réseau traditionnel.

Chapitre 4

Mise en oeuvre du réseau

4.1 Introduction

Nous réaliserons dans ce chapitre l'implémentation de notre réseau IP/MPLS sur GNS3 et Eve-ng deux émulateurs que nous présenterons, tout en donnant plus de détails par rapport à notre topologie ainsi que les librairies réseau. Nous réaliserons aussi une comparaison entre un réseau IP/MPLS et un réseau IP.

4.2 Outils d'implémentation réseau

4.2.1 Présentation des émulateurs

Un émulateur de réseau est utilisé pour tester les performances d'un réseau réel. Il peut également être utilisé à des fins telles que la validation de concept. Un émulateur de réseaux permet aux architectes, aux ingénieurs et aux développeurs de réseaux d'évaluer avec précision la réactivité, le débit et la qualité de l'expérience de l'utilisateur final d'une application avant d'apporter des modifications ou des ajouts à un système.

4.2.1.1 GNS3

GNS3 fournit un environnement virtuel permettant de concevoir et d'optimiser des réseaux de toute taille sans avoir à construire une infrastructure matérielle physique. GNS3 utilise de véritables images Cisco IOS qui émulent les routeurs à l'aide d'un programme appelé Dynamips.
GNS3 est similaire à la partie interface utilisateur graphique (GUI) de tout autre logiciel installé. En utilisant l'interface graphique, il est facile de construire des laboratoires complexes composés d'une variété de routeurs Cisco pris en charge. Dynampis est souvent appelé le back-end tandis que Dynagen est le système front-end, principalement parce que Dynagen communique avec Dynamips en utilisant un hyperviseur. L'ensemble du système simplifie le processus de configuration. Le simulateur de réseau graphique est installé sur un ordinateur et l'architecture du réseau est conçue et configurée pour les réseaux IP et MPLS.

4.2.1.2 Eve-NG

EVE-NG est un outil similaire à GNS3 qui permet aux administrateurs de réseaux de simuler des routeurs, des commutateurs, des pare-feu et de nombreux autres appareils virtuels. Vous pouvez créer un laboratoire réseau avec des dispositifs de Cisco, Juniper, Citrix, Arista, A10, Alcatel, Checkpoint, F5, Palo Alto, PFSense, SonicWALL, Trend Micro TippingPoint vTPS, et bien d'autres encore. Si le fournisseur de réseau dispose d'un appareil virtuel, il est plus que probable qu'il puisse fonctionner dans un environnement EVE-NG. Vous pouvez même ajouter des images de serveurs Linux et Windows.

4.3 Telnet et SSH

Afin de configurer un matériel réseau physique, nous devons seulement brancher notre pc à la console de l'appareil.
Ceci n'est pas pratique car si nous voulons configurer ou avoir accès à un appareil géographiquement loin, nous devons nous déplacer.
Les protocoles telnet et SSH permettent de remédier à celà.
Le protocole telnet nous permet d'établir une connexion directement à notre matériel sous condition d'en connaître l'adresse IP et avoir les accès nécessaires. Cependant, lorsque nous l'utilisons, toutes les données véhiculées sont visibles sous forme de texte, donc n'importe quelle personne qui suit le trafique des données peut voir tout ce qui est transmis.
C'est alors là que vient le protocole SSH pour y remédier en nous donnant l'option de crypter les données. Il y aura alors toujours du texte transmis, mais il sera cependant totalement incompréhensible par un humain.

4.4 Programmation réseau Python

4.4.1 telnetlib

Le module telnetlib fournit une classe Telnet qui implémente le protocole Telnet. Nous pouvons alors nous connecter directement via Python à un routeur où l'accès via telnet est autorisé.

4.4.2 Netmiko et Parmiko

Paramiko est une implémentation purement Python du protocole SSHv2, fournissant à la fois des fonctionnalités client et serveur(21).

L'automatisation du réseau pour les dispositifs de capture d'écran consiste principalement à collecter la sortie des commandes d'affichage et à effectuer des changements de configuration.
Netmiko vise à accomplir ces deux opérations et à le faire sur un très large ensemble de plateformes. Il cherche à le faire tout en abstrayant le contrôle d'état de bas niveau.
Paramiko est davantage un module SSH générique que vous pouvez utiliser pour automatiser des tâches SSH spécifiques. En revanche, Netmiko est plus large et bien optimisé pour la gestion des périphériques réseau tels que les commutateurs et les routeurs.
L'abstraction est l'autre avantage de l'utilisation de Netmiko. Netmiko fournit une fonction simple que vous pouvez utiliser pour désactiver la pagination. Par exemple, une sortie de la session SSH pourrait être plus d'une page. En utilisant les sessions SSH ordinaires, vous devrez ajouter un espace de type entrée pour montrer la page suivante. Netmiko vous fournit un moyen de contourner cela.

4.5 But de l'implémentation

Le but principal de l'implémentation sur GNS3 et Eve-NG et d'émuler le réseau IP/MPLS de OOREDOO et de tester si possible ses performances.
Nous nous intéresserons qu'à une petite partie du réseau, (le backbone et l'accès) car le reste du réseau y est similaire.
Nous verrons aussi plus tard la possibilité de connecter notre application Python à notre matériel via les librairies Netmiko ou Telnet.

4.6 Réalisation sur GNS3

Nous réalisons sur GNS3 la topologie suivante avec des routeurs cisco **c7200** :

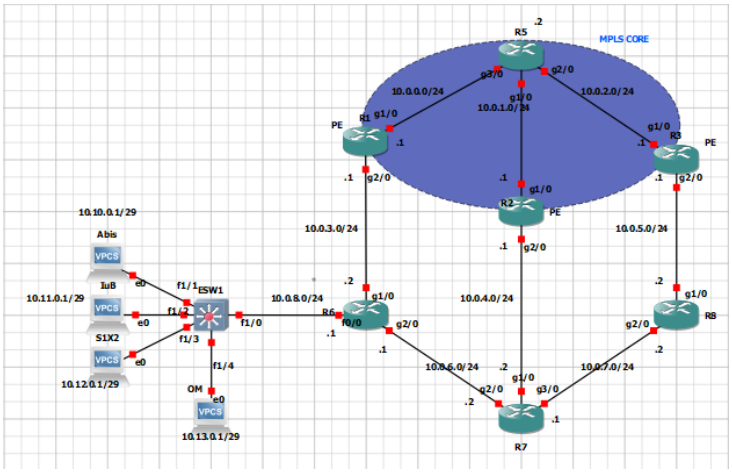


FIGURE 4.1 – Tolopogie réalisée sur GNS3

Notre topologie est divisée en 2 parties, la première qui est le core où les différents data center sont interconnectés ainsi que la partie access où l'on a les "provider edge" (routeur de bord) et le matériel des différentes technologies (2G 3G 4G) et leurs interfaces. Les différentes technologies d'un sité sont localisés dans le même site et chaque technologie appartient à un VLAN différent. La passerelle par défaut de tout les vlans est le routeur de bord connecté au site.

Nous veillons bien-sûr à bien configurer nos VLANs.

Nous configurerons la technique de routage OSPF en premier lieu sur toutes les interfaces connectées de nos routeurs ainsi que les interfaces loopback.

Nous configurerons les protocoles MPLS IP et LDP sur nos routeurs.

Nous configurerons les tunnels BGP sur les interfaces des routeurs de notres core network.

Nous ajouterons finalement des VRF sur nos routeurs PE(Provider Edge).

La configuration et les commandes de tous les appareils sera transcrite en l'annexe.

4.6.1 Découpage réseau et tests

Nous réalisons le découpage réseau de notre topologie suivant le tableau :

Équipement	Interface	Adresse IPv4	Équipement connecté
R1	GigabitEthernet1/0	10.0.0.1/24	R5
	GigabitEthernet2/0	10.0.3.1/24	R6
R2	GigabitEthernet1/0	10.0.1.1/24	R5
	GigabitEthernet2/0	10.0.4.1/24	R6
R3	GigabitEthernet1/0	10.0.2.1/24	R5
	GigabitEthernet2/0	10.0.5.1/24	R8
R5	GigabitEthernet1/0	10.0.1.2/24	R2
	GigabitEthernet2/0	10.0.2.2/24	R3
	GigabitEthernet3/0	10.0.0.2/24	R1
R6	GigabitEthernet1/0	10.0.3.2/24	R1
	GigabitEthernet2/0	10.0.6.1/24	R7
	FastEthernet0/0	10.0.8.1/24	ESW1
R7	GigabitEthernet1/0	10.0.4.2/24	R2
	GigabitEthernet2/0	10.0.6.2/24	R6
	GigabitEthernet3/0	10.0.7.1/24	R8
R8	GigabitEthernet1/0	10.0.5.2/24	R3
	GigabitEthernet2/0	10.0.7.2/24	R7

Nous réalisons à présent quelques commandes sur notre matériel pour vérifier la connectivité et la configuration de quelques paramètres de nos routeurs.
Afin de vérifier les routes atteignables par notre routeur on utilise la commande **"Show ip route" :**

Afin de vérifier les labels mpls selon les interfaces des routeurs voisins on utilise la commande**"Show mpls ip binding" :**

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
C       1.1.1.1 is directly connected, Loopback0
     2.0.0.0/32 is subnetted, 1 subnets
O       2.2.2.2 [110/3] via 10.0.0.2, 01:44:51, GigabitEthernet1/0
     3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/3] via 10.0.0.2, 01:44:51, GigabitEthernet1/0
     5.0.0.0/32 is subnetted, 1 subnets
O       5.5.5.5 [110/2] via 10.0.0.2, 01:45:01, GigabitEthernet1/0
     10.0.0.0/24 is subnetted, 3 subnets
O       10.0.2.0 [110/2] via 10.0.0.2, 01:45:01, GigabitEthernet1/0
C       10.0.0.0 is directly connected, GigabitEthernet1/0
O       10.0.1.0 [110/2] via 10.0.0.2, 01:45:01, GigabitEthernet1/0
```

FIGURE 4.2 – Résultat de la commande show ip route sur le routeur R1

```
R1#sh mpls ip binding
  1.1.1.1/32
        in label:     imp-null
        out label:    17        lsr: 5.5.5.5:0
  2.2.2.2/32
        in label:     20
        out label:    18        lsr: 5.5.5.5:0        inuse
  3.3.3.3/32
        in label:     19
        out label:    16        lsr: 5.5.5.5:0        inuse
  5.5.5.5/32
        in label:     16
        out label:    imp-null  lsr: 5.5.5.5:0        inuse
  10.0.0.0/24
        in label:     imp-null
        out label:    imp-null  lsr: 5.5.5.5:0
  10.0.1.0/24
        in label:     17
        out label:    imp-null  lsr: 5.5.5.5:0        inuse
  10.0.2.0/24
        in label:     18
        out label:    imp-null  lsr: 5.5.5.5:0        inuse
```

FIGURE 4.3 – Résultat de la commande Show mpls ip binding sur le routeur R1

Afin de vérifier les tunnels bgp selon les interfaces loopback des routeurs voisins on utilise la commande**"Show ip bgp all summary" :**

```
R1#sh ip bgp all summary
For address family: IPv4 Unicast
BGP router identifier 1.1.1.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2.2.2.2         4          1     107     107        1    0    0 01:46:22        0
3.3.3.3         4          1     107     107        1    0    0 01:46:35        0

For address family: VPNv4 Unicast
BGP router identifier 1.1.1.1, local AS number 1
BGP table version is 1, main routing table version 1

Neighbor        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2.2.2.2         4          1     107     107        1    0    0 01:46:22        0
3.3.3.3         4          1     107     107        1    0    0 01:46:35        0
```

FIGURE 4.4 – Résultat de la commande Show ip bgp all summary sur le routeur R1

Afin de vérifier les routes possibles de notre table de routage virtuelle (VRF) on utilise

la commande**"Show ip route vrf PFE" :**

```
R1#sh ip route vrf PFE

Routing Table: PFE
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     6.0.0.0/32 is subnetted, 1 subnets
O       6.6.6.6 [110/2] via 10.0.3.2, 01:44:03, GigabitEthernet2/0
     7.0.0.0/32 is subnetted, 1 subnets
O       7.7.7.7 [110/3] via 10.0.3.2, 01:44:03, GigabitEthernet2/0
     8.0.0.0/32 is subnetted, 1 subnets
O       8.8.8.8 [110/4] via 10.0.3.2, 01:44:03, GigabitEthernet2/0
     10.0.0.0/24 is subnetted, 5 subnets
C       10.0.3.0 is directly connected, GigabitEthernet2/0
O       10.0.6.0 [110/2] via 10.0.3.2, 01:44:03, GigabitEthernet2/0
O       10.0.7.0 [110/3] via 10.0.3.2, 01:44:03, GigabitEthernet2/0
O       10.0.4.0 [110/3] via 10.0.3.2, 01:44:03, GigabitEthernet2/0
```

FIGURE 4.5 – Résultat de la commande SShow ip route vrf PFE sur le routeur R1

Nous testons maintenant la connectivité de notre routeur dans le core network et dans l'access network.

```
R1#ping 10.0.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/36/48 ms
R1#ping VRF PFE 10.0.7.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.7.2, timeout is 2 seconds:
!!!!!
```

FIGURE 4.6 – Tests de connectivité entre le routeur R1 le routeur R3 et R8

Nous pouvons remarquer que dans le core (où l'MPLS est activé) le temps de réponse est plus court que pour un réseau IP classique (dans la partie access).

4.7 Réalisation sur Eve-NG

Nous réalisons sur Eve-NG la topologie suivante, où nous utiliserons pricnipalement des routeurs Juniper :

Afin de configurer notre matériel nous utilisons le logiciel Putty qui est un émulateur de clients, afin de nous connecter à notre materiel via telnet ou SSH.

Nous réalisons la configuration suivante sur le routeur VMX1. La configuration sera les mêmes pour les autres routeurs, on veillera juste à changer l'adresse IP.

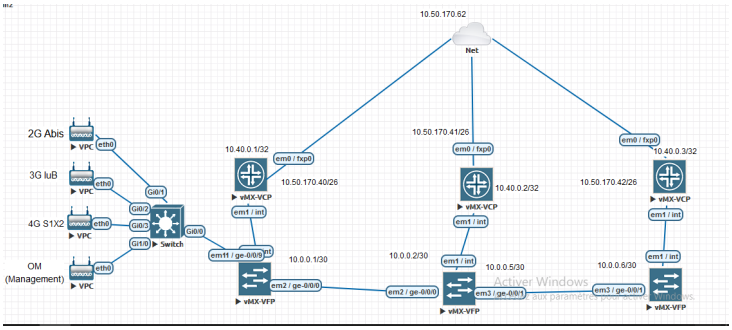


FIGURE 4.7 – Topologie réalisée sur Eve-NG en étant connecté aux serveurs de OOREDOO

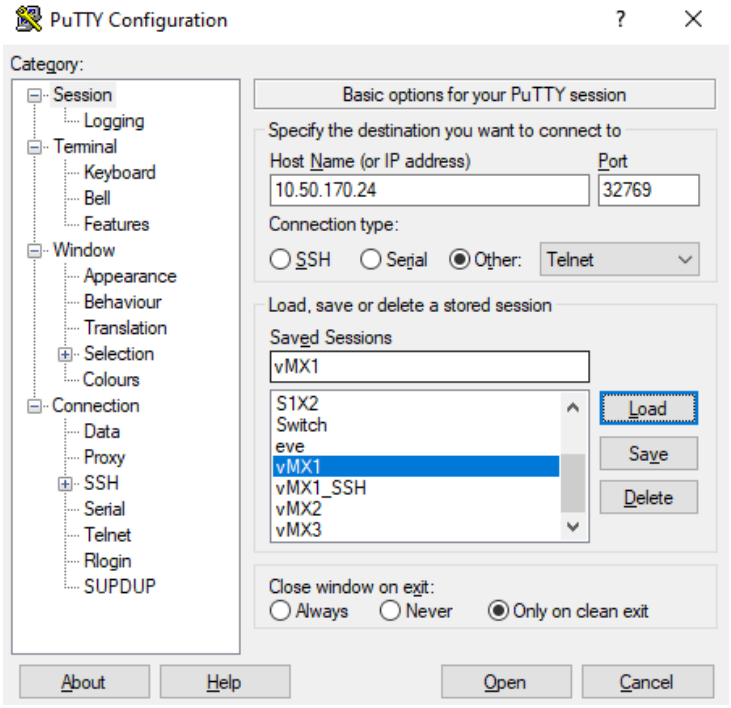


FIGURE 4.8 – Utilisation de Putty afin de se connecter au matériel via telnet ou SSH

Nous réalisons aussi la configuration des différents VLAN sur nos switchs :

Nous testons maintenant la connectivité à notre librairie Python netmiko :

```
R1 = {
    'device_type': 'juniper',
    'host':    '10.50.170.24',
    'username': 'BI0700_MX480_SR',
    'password': 'test',
    'port' : '32774',
    'secret': 'test',}
```

```
from devices import all_devices
```

FIGURE 4.9 – Configuration du routeur VMX1 sur Eve-ng

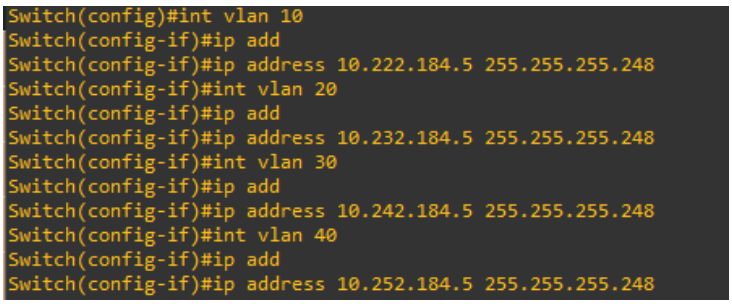


FIGURE 4.10 – Configuration de nos switchs sur Eve-ng

```
from commands import all_commands
import getpass
import netmiko
import paramiko
from netmiko import ConnectHandler
from paramiko import SSHException
for devices in all_devices :
    net_connect = ConnectHandler(**devices, banner_timeout=200,
    blocking_timeout=20, timeout=100, session_timeout=60,encoding='utf-8')
    net_connect.enable()
    cmd = ["show interfaces terse routing-instance IP_RAN"]
```

On obtient alors :

FIGURE 4.11 – Test de la connectivité de Python avec notre matériel en utilisant la librairie Netmiko

4.8 Tests des performances

Nous faisons un test de performance sur notre réseau IP/MPLS en le comparant avec un réseau traditionnel IP.
Nous utiliserons pour celà un fichier de traçage.
Nous utiliserons pour la simulation d'un réseau IP traditionnel, la même topologie que pour notre réseau MPLS, en veillant bien-sûr à ce que l'MPLS soit désactivée et en utilisant comme protocole de routage l'OSPF.
Voici les résultats que nous obtenons :

	Réseau traditionnel IP	Réseau MPLS
Nombre de paquets reçus	335	332
Moyenne de paquets par seconde	0.769	4.728
Moyenne de la taille d'un paquet	83	98
Octets transférés	27900	32700
Moyenne d'octets par seconde	64	466
Moyenne de megabits par seconde	0.001	0.004

Nous remarquons que, bien que la taille des paquets MPLS est plus grande qu'en IP en raison de l'étiquette (label) supplémentaire, la moyenne de paquets par seconde dans l'MPLS est trois-quatre fois plus élevée que dans le routage IP traditionnel.
De même, le nombre moyen de mégabits par seconde est plus élevé dans le domaine

MPLS.
On peut donc dire que l'MPLS est plus rapide que que le routage IP traditionnel.

4.9 Comparaison entre GNS3 et Eve-NG

Eve-ng et GNS3 sont deux excellents logiciels d'émulation gratuits. Ils ont rendu très facile la construction de réseaux de test virtuels très rapidement. Grâce aux fonctions produites par les dispositifs de réseau virtuel dans GNS3 et EVE NG, il nous est plus facile de comprendre comment les dispositifs de réseau physique réels vont réagir dans le monde réel.
Une image d'un matériel hardware nous permet d'émuler le matériel sur notre machine virtuelle, et de ce fait grâce à l'image, nous pouvons reproduire sur nos émulateurs les même défauts techniques de la vie réelle.
Lorsque on utilise gns3 ou Eve-NG, on peut avoir des besoins différents selon le type d'images que nous utilisons et le fournisseur. Eve-NG ne nécessite pas trop de connaissances en matière de configuration une fois que nous avons importé l'image supportée. Lorsqu'on utilise le client htlm5, nous disposons d'une interface très conviviale qui permet de faire plus de mise en réseau que de configuration.
Le plus grand avantage de Eve-ng est qu'il peut être accédé via une page web (html5) comme dans notre cas une fois connecté au serveur OOREDDO, de plus, il est très gourmand au niveau matériel.
Tandis que l'avantage de GNS3 est dans sa communauté toujours à l'écoute en cas de besoin et toujours là pour proposer des solutions en cas de problèmes.

4.10 Conclusion

Nous avons vu dans ce chapitre deux des émulateurs les plus populaires en réseaux informatiques.
Nous avons veillé à mettre en oeuvre notre réseau IP/MPLS sur GNS3 et Eve-NG et démontré que pour les opérateurs mobiles, les réseaux MPLS sont beaucoup plus performants que les réseaux traditionnels IP.
Nous verrons dans le prochain chapitre la problématique à laquelle on fait face. Puis nous proposerons une solution que nous veillerons à développer.

Chapitre 5

Intégration de la base de données et de la plateforme Web, tests et résultats

5.1 Introduction

Nous présenterons dans ce chapitre la problématique à laquelle on fait face qui est le manque d'automatisation et de facilité d'accès à la donnée. Puis nous proposerons une solution que nous développerons au préalable.

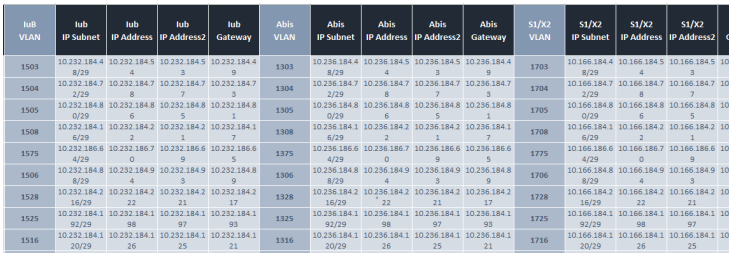
5.2 Développement de la problématique

L'IP Plan est un document contenant toutes les données relatives au réseau de OOREDOO.
On y trouve notamment tous les sites(plus de 3000) sur le territoire national, les technologies qu'on peut y trouver (2G 3G ou 4G), les données IP relatives à ces technologies, la connectivité de tout les sites ainsi que l'acheminement des sites, c'est à dire les chemins par lesquels passent sites afin d'atteindre les différents Data Center de OOREDOO.

2G site code	MPLS code	MPLS TX site	MPLS transmission connection	TCU connection(free port)	O&M VLAN	O&M IP Subnet	O&M NodeB	O&M TCU	O&M eNodeB	O&M Gateway	IuB VLAN
AI4402	AIDSR	AIT01	AIT01 TN1	AI4402 TN1 LAN 1/10/6	2303	10.222.184.48/29	10.222.184.53	10.222.184.52	10.222.184.54	10.222.184.49	1503
AI4403	AIDSR	AIT01	AIT01 TN1	AI4403 0 LAN 0	2304	10.222.184.72/29	10.222.184.77	10.222.184.76	10.222.184.78	10.222.184.73	1504
AI4404	AIDSR	AIT01	AIT01 TN1	AI4404 0 LAN 0	2305	10.222.184.80/29	10.222.184.85	10.222.184.84	10.222.184.86	10.222.184.81	1505
AI4442	AIDSR	AIT01	AIT01 TN1	AI4442 0 LAN 0	2308	10.222.184.16/29	10.222.184.21	10.222.184.20	10.222.184.22	10.222.184.17	1508
AI4483	AIDSR	AIT01	AIT01 TN1	AI4483 TN LAN 1/1/2	2375	10.222.186.64/29	10.222.186.70	10.222.186.69	10.222.186.68	10.222.186.65	1575
AI4405	AIDSR	AIT01	AIT01 TN1	AI4405 0 LAN 0	2306	10.222.184.88/29	10.222.184.93	10.222.184.92	10.222.184.94	10.222.184.89	1506
AI4419	AIDSR	AIT01	AIT01 TN1	AI4419 0 LAN 0	2328	10.222.184.216/29	10.222.184.222	10.222.184.221	10.222.184.220	10.222.184.217	1528
AI4416	AIDSR	AIT01	AIT01 TN1	AI4416 TN LAN 1/7/2	2325	10.222.184.192/29	10.222.184.198	10.222.184.197	10.222.184.196	10.222.184.193	1525
AI4407	AIDSR	AIT01	AIT01 TN1	AI4407 TN LAN 1/1/2	2316	10.222.184.120/29	10.222.184.126	10.222.184.125	10.222.184.124	10.222.184.121	1516
AI4461	AIDSR	AIT01	AIT01 TN1	AI4461 TN LAN 1/1/2	2361	10.222.185.224/29	10.222.185.230	10.222.185.229	10.222.185.228	10.222.185.225	1561

Rech Conectivité | Rech IP+Vlan | Rech DCN | POS | Colocated site | Conectivité | DCN | IP+Vlan

FIGURE 5.1 – L'IP Plan de OOREDOO



IuB VLAN	Iub IP Subnet	Iub IP Address	Iub IP Address2	Iub Gateway	Abis VLAN	Abis IP Subnet	Abis IP Address	Abis IP Address2	Abis Gateway	S1/X2 VLAN	S1/X2 IP Subnet	S1/X2 IP Address	S1/X2 IP Address2
1503	10.232.184.48/29	10.232.184.54	10.232.184.53	10.232.184.49	1303	10.236.184.48/29	10.236.184.54	10.236.184.53	10.236.184.49	1703	10.166.184.48/29	10.166.184.54	10.166.184.53
1504	10.232.184.72/29	10.232.184.78	10.232.184.77	10.232.184.73	1304	10.236.184.72/29	10.236.184.78	10.236.184.77	10.236.184.73	1704	10.166.184.72/29	10.166.184.78	10.166.184.77
1505	10.232.184.80/29	10.232.184.86	10.232.184.85	10.232.184.81	1305	10.236.184.80/29	10.236.184.86	10.236.184.85	10.236.184.81	1705	10.166.184.80/29	10.166.184.86	10.166.184.85
1508	10.232.184.16/29	10.232.184.22	10.232.184.21	10.232.184.17	1308	10.236.184.16/29	10.236.184.22	10.236.184.21	10.236.184.17	1708	10.166.184.16/29	10.166.184.22	10.166.184.21
1575	10.232.186.64/29	10.232.186.70	10.232.186.69	10.232.186.65	1375	10.236.186.64/29	10.236.186.70	10.236.186.69	10.236.186.65	1775	10.166.186.64/29	10.166.186.70	10.166.186.69
1506	10.232.184.88/29	10.232.184.93	10.232.184.92	10.232.184.89	1306	10.236.184.88/29	10.236.184.94	10.236.184.93	10.236.184.89	1706	10.166.184.88/29	10.166.184.94	10.166.184.93
1528	10.232.184.216/29	10.232.184.222	10.232.184.221	10.232.184.217	1328	10.236.184.216/29	10.236.184.222	10.236.184.221	10.236.184.217	1728	10.166.184.216/29	10.166.184.222	10.166.184.221
1525	10.232.184.192/29	10.232.184.198	10.232.184.197	10.232.184.193	1325	10.236.184.192/29	10.236.184.198	10.236.184.197	10.236.184.193	1725	10.166.184.192/29	10.166.184.198	10.166.184.197
1516	10.232.184.120/29	10.232.184.126	10.232.184.125	10.232.184.121	1316	10.236.184.120/29	10.236.184.126	10.236.184.125	10.236.184.121	1716	10.166.184.120/29	10.166.184.126	10.166.184.125

FIGURE 5.2 – L'IP Plan de OOREDOO

Nous sommes intéressés dans notre étude par les différents sites et leur IP Plan. Chaque site sur le territoire national possède au moins une ou toutes les technologies

de télécommunications (2G,3G 4G). Nous pouvons comme nous l'avons vu précédemment dans le chapitre 1, communiquer avec ces technologies d'un point de vue informatique en utilisant le modèle TCP/IP.
Chaque technologie est représentée par une interface :

- Abis pour la 2G.
- IuB pour la 3G.
- S1/X2 pour la 4G.
- OM pour le management des réseaux.

Les interfaces d'un site ne sont pas séparées physiquement. Nous utilisons pour celà des VLANs (réseaux locaux virtuels) afin de diminuer le coût et pour divers raisons de sécurité. Chaque interface de chaque site appartient à un VLAN précis et chacune possède une adresse IPv4 unique et une passerelle unique.
L'IP Plan de OOREDOO est sauvegardé et partagé dans un fichier excel. De plus il doit à chaque fois être modifié manuellement après qu'il y ait eu un changement sur le réseau.
De ces faits, l'IP Plan de se retrouve totalement décentralisé, et n'importe quelle personne ayant les droits d'y accéder doit le demander à une autre le possédant. Et si jamais il est envoyé par erreur à une personne non concernée celà pourrait atteindre la confidentialité de OOREDOO.
Nous pouvons citer parmis les différents incovénients de l'utilisation d'Excel :

- L'absence d'automatisation.
- L'absence de lien entre les données.
- La décentralisation des données.
- L'absence d'intégrité référentielle.
- Difficulté dans le suivi de l'évolution du projet.
- Pas de mise en place de processus possibles.
- Pas de possibilité de différencier les niveaux d'accès aux infos

Le fait de devoir à chaque fois mettre à jour manuellement notre fichier excel peut être fatiguant et long à réaliser, sans oublier l'erreur humaine dans l'équation.

5.3 Solution proposée

Nous faisons face comme dit précédemment à deux grandes problématiques principales, notamment l'utilisation d'Excel et le fait de devoir changer manuellement notre IP Plan.
Pour pallier au premier problème, nous proposons d'utiliser une base de données que nous gérerons grâce à MySQL qui est un système de gestion de bases de données.
De plus nous aurons la possibilité d'afficher cette base de données sur une interface Web, interface sur laquelle on pourra insérer des requêtes afin de chercher des sites spécifiques selon les critères que nous souhaitons. Ce qui permettra d'accéder à notre base de données à tout moment tant que nous avons accès à internet, en ayant bien entendu les autorisations et les accès nécessaires.
Quant au second problème, en étant connectés au réseau OOREDOO nous proposons d'exploiter certaines commandes des routeurs utilisés (nous utiliserons dans notre cas des routeurs Cisco et/ou Juniper), via Python grâce aux protocoles SSH ou telnet en utilisant les librairies netmiko, paramiko et telnet.
Il suffira alors d'obtenir les données souhaitées sur nos sites via certaines commandes, et utiliser l'output (le résultat) comme on le souhaite afin de l'ajouter automatiquement à notre base de données.

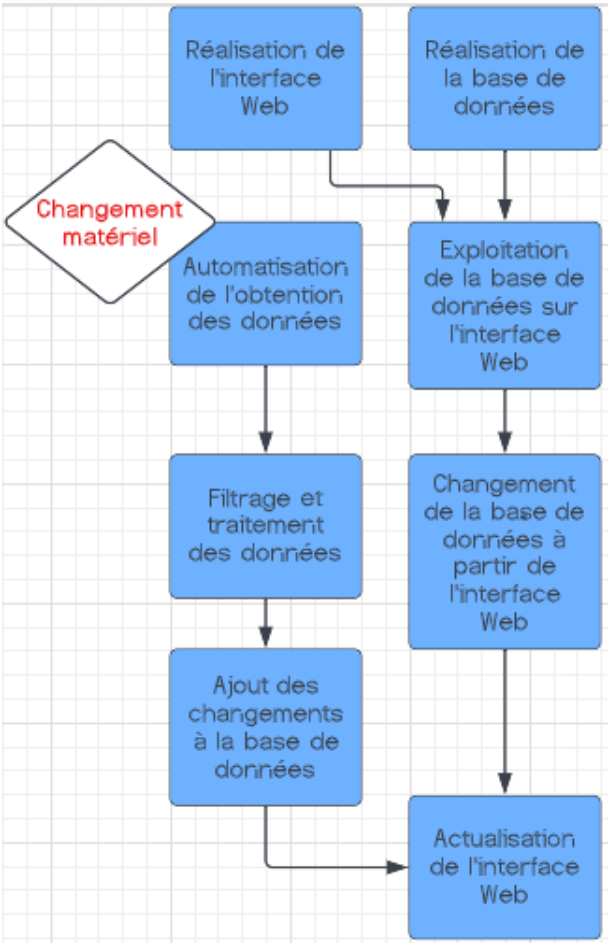


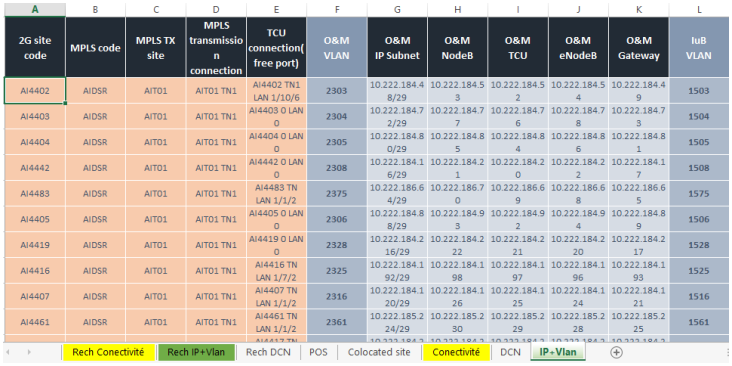
FIGURE 5.3 – Organigramme de la démarche à suivre afin de réaliser notre projet

5.4 Développement et réalisation de la solution

La réalisation de la solution proposée se divise en cinq grandes étapes :

5.4.1 Réalisation de la base de données

Notre point de départ afin de réaliser la base de données est **l'IP PLAN** de OOREDOO dans un fichier Excel.



2G site code	MPLS code	MPLS TX site	MPLS transmission connection	TCU connection(free port)	O&M VLAN	O&M IP Subnet	O&M NodeB	O&M TCU	O&M eNodeB	O&M Gateway	IuB VLAN
AI4402	AIDSR	AIT01	AIT01 TN1	AI4402 TN1 LAN 1/10/6	2303	10.222.184.48/29	10.222.184.53	10.222.184.52	10.222.184.54	10.222.184.49	1503
AI4403	AIDSR	AIT01	AIT01 TN1	AI4403 0 LAN 0	2304	10.222.184.72/29	10.222.184.77	10.222.184.76	10.222.184.78	10.222.184.73	1504
AI4404	AIDSR	AIT01	AIT01 TN1	AI4404 0 LAN 0	2305	10.222.184.80/29	10.222.184.85	10.222.184.84	10.222.184.86	10.222.184.81	1505
AI4442	AIDSR	AIT01	AIT01 TN1	AI4442 0 LAN 0	2308	10.222.184.16/29	10.222.184.21	10.222.184.20	10.222.184.22	10.222.184.17	1508
AI4483	AIDSR	AIT01	AIT01 TN1	AI4483 TN LAN 1/1/2	2375	10.222.186.64/29	10.222.186.70	10.222.186.69	10.222.186.68	10.222.186.65	1575
AI4405	AIDSR	AIT01	AIT01 TN1	AI4405 0 LAN 0	2306	10.222.184.88/29	10.222.184.93	10.222.184.92	10.222.184.94	10.222.184.89	1506
AI4419	AIDSR	AIT01	AIT01 TN1	AI4419 0 LAN 0	2328	10.222.184.216/29	10.222.184.222	10.222.184.221	10.222.184.220	10.222.184.217	1528
AI4416	AIDSR	AIT01	AIT01 TN1	AI4416 TN LAN 1/7/2	2325	10.222.184.192/29	10.222.184.198	10.222.184.197	10.222.184.196	10.222.184.193	1525
AI4407	AIDSR	AIT01	AIT01 TN1	AI4407 TN LAN 1/1/2	2316	10.222.184.120/29	10.222.184.126	10.222.184.125	10.222.184.124	10.222.184.121	1516
AI4461	AIDSR	AIT01	AIT01 TN1	AI4461 TN LAN 1/1/2	2361	10.222.185.224/29	10.222.185.230	10.222.185.229	10.222.185.228	10.222.185.225	1561

Rech Conectivité | Rech IP+Vlan | Rech DCN | POS | Colocated site | Conectivité | DCN | IP+Vlan

FIGURE 5.4 – L'IP Plan de OOREDOO

Nous devons pouvoir mettre(transférer) en premier lieu ce tableau Excel dans une base de donnée.

Nous commençons tout d'abord par créer notre base de données. On utilisera MySQL pour tout traitement de notre base de données : Nous utilisons alors maintenant la

```
create database if not exists test;
use test;
```

FIGURE 5.5 – Création de la base de données sur MySQL

base de donnée nommée **"test"**.

Nous devons maintenant créer les tables de notre base de données. étant donné que les VLAN ID des différentes interfaces ne sont pas uniques, nous devons choisir nos clés primaires pour chaque table.

Nous choisissons de créer en premier lieu une table pour les sites contenant toutes les données relatives à chaque site ainsi qu'aux adresses IP de chaque interface. Nous choisissons comme clé primaire le **2G site code** de chaque site puisqu'il est unique et propre à chaqu'un sur le territoire national.

```
use test;
DROP TABLE IF exists sites;
CREATE TABLE sites(
 SITE_CODE varchar(15) PRIMARY KEY,
 MPLS_CODE varchar(15),
 MPLS_TX_SITE varchar(15),
 TCU_CONNECTION varchar(40),
```

```
 OM_VLAN_id int,
 OM_Subnet varchar(20),
 OM_NodeB varchar(20),
 OM_TCU varchar(20),
 OM_enodeB varchar(20),
 OM_gateway varchar(20),
 IuB_VLAN_id int,
 IuB_Subnet varchar(20),
 IuB_Adress varchar(20),
 IuB_Adress2 varchar(20),
 IuB_gateway varchar(20),
 Abis_VLAN_id int,
 Abis_Subnet varchar(20),
 Abis_Adress varchar(20),
 Abis_Adress2 varchar(20),
 Abis_gateway varchar(20),
 S1X2_VLAN_id int,
 S1X2_Subnet varchar(20),
 S1X2_Adress varchar(20),
 S1X2_Adress2 varchar(20),
 S1X2_gateway varchar(20)
 );
```

Nous avons maintenant créé la table **"sites"** avec tout les attributs souhaités. Nous créons aussi une seconde table **"ach"** pour l'acheminement des données, nous choisissons encore, le code du site comme clé primaire :

```
    DROP TABLE IF exists ach;
CREATE TABLE ach(
SITE_CODE varchar(50) PRIMARY KEY,
MPLS_SITE varchar(10),
MPLS0 varchar(50),
A1 varchar(50),
A2 varchar(20),
```

```
A3 varchar(20),
A4 varchar(20),
A5 varchar(20),
A6 varchar(20),
A7 varchar(20),
A8 varchar(20),
A9 varchar(20),
A10 varchar(20),
A11 varchar(20),
A12 varchar(20)
);
```

Maintenant que nous avons créé nos tables dans notre base de données, nous devons les remplir avec les données que nous possédons déjà sur les sites, notamment le fichier Excel **"IP PLAN"**. Nous devons d'abord transformer notre fichier excel en fichier *.csv* afin de permettre à MySQL d'y récolter nos données :

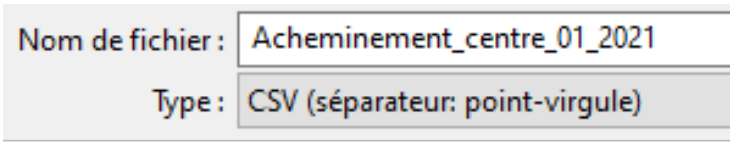


FIGURE 5.6 – Transformation de notre fichier excel en fichier .csv

Avant de remplir notre base de données, nous devons d'abord veiller à ce que les autorisations de lecture soient bien vérifiés tout d'abord en exécutant dans une cmd. window la commande :

```
SET GLOBAL local_infile = TRUE;
```

et mettre nos fichiers .csv dans un répértoire bien précis où il est possible pour MySQL de lire les données. Les dossiers :

```
C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/
```

Nous pouvons dès lors, remplir notre base de données avec nos données initiales.

```
    LOAD DATA LOCAL INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/
    Uploads/final.csv" INTO TABLE test.sites
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
(SITE_CODE,MPLS_CODE,MPLS_TX_SITE,TCU_CONNECTION,OM_VLAN_id, OM_Subnet,
OM_NodeB, OM_TCU, OM_enodeB, OM_gateway,IuB_VLAN_id, IuB_Subnet,
IuB_Adress, IuB_Adress2, IuB_gateway, Abis_VLAN_id,
Abis_Subnet, Abis_Adress,
Abis_Adress2, Abis_gateway,S1X2_VLAN_id, S1X2_Subnet,
S1X2_Adress, S1X2_Adress2, S1X2_gateway);

LOAD DATA LOCAL INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/
Uploads/Acheminement.csv" INTO TABLE test.ach
CHARACTER SET latin1
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
(SITE_CODE,MPLS_SITE,MPLS0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12);
```

Nous obtenons alors notre base de donnée que nous pouvons afficher sur MySQL :

Nous pouvons aussi afficher nos tableaux grâce aux commandes :

```
SELECT * FROM test.sites;
SELECT * FROM test.ach;
```

et nous obtenons :

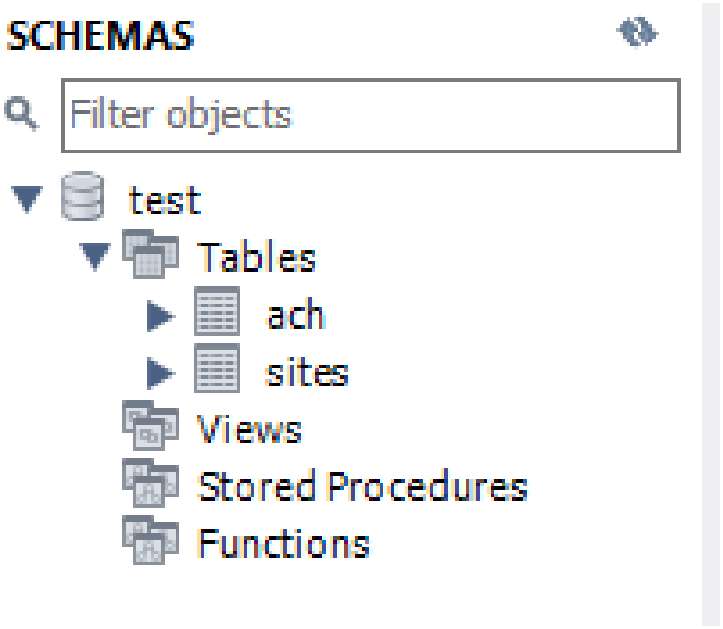


FIGURE 5.7 – Visualisation de nos tables dans notre base de données

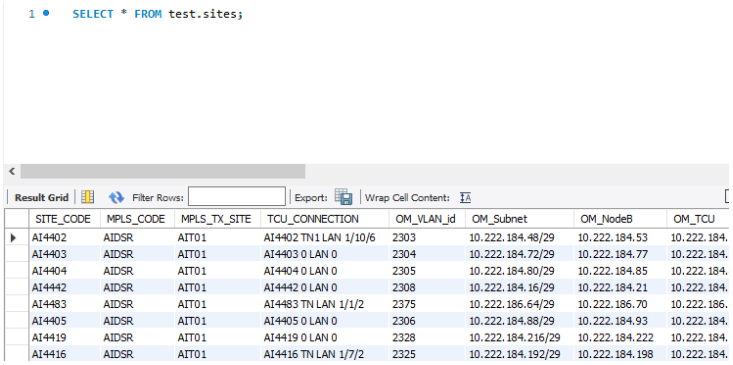


FIGURE 5.8 – Tables sites dans MySQL

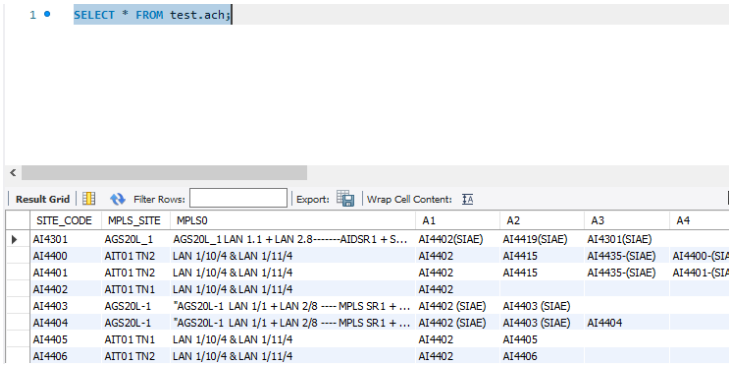


FIGURE 5.9 – Tables ach dans MySQL

Nous pouvons maintenant passer à l'étape suivante qui est la réalisation de notre interface Web.

5.4.2 Réalisation de l'interface Web

Le but de notre interface Web est principalement d'afficher les données de notre base de données et de chercher et trouver n'importe quel site, selon n'importe quel critère, c'est à dire n'importe quel attribut de nos tables.
Nous commençons par réaliser le front end, c'est à dire l'interface web visible à l'utilisateur sans qu'il y ait un traitement de données (input ou output).
Nous utilisons HTML5 et CSS pour la réalisation de l'interface Web.
Nous commençons par créer les fichiers .html que nous veillons à mettre dans un dossier **"template"**. Nous créeons les pages :

- IP_PLAN.html
- acheminement.html
- add.html
- change.html
- remove.html
- search.html
- parameters.html

Nous devons avoir une page pour afficher l'**L'IP PLAN** c'est à dire la table sites de notre base de données, une autre pour afficher l'acheminement des sites, donc la table ach de notre base de données.
Nous devons aussi pouvoir chercher n'importe quel site à partir de notre interface, et avoir la possiblité de changer manuellement n'importe quel site(à partir de l'interface aussi).

5.4.2.1 L'affichage de l'IP PLAN et la recherche de sites

Voici le code HTML pour afficher et chercher dans notre IP PLAN :

```
<h1>Les adresses IP des VLAN des diff rents sites :</h1>

          <div class="Les_IP">
              <table border="0" style="box-shadow:1px 1px 10px white"
  width="70%" height="70%" align="center">
                  <tr>
                  <td style="background-position:center center" valign="
  top" align="center">
```

```
7                    <table border="0" cellpadding="8px" width="100%" style
  ="background-color:#CCCCCC">
8                        <tr style="background-color:navy;color:white">
9                        <td>Site Code</td>
10                       <td>OM VLAN id</td>
11                       <td>IuB VLAN id</td>
12                       <td>Abis VLAN id</td>
13                       <td>S1X2 VLAN id</td>
14                    </tr>
15                     {% for sites in data %}
16                   <tr>
17                    <td>{{sites.SITE_CODE}}</td>
18                    <td>{{sites.OM_VLAN_id}}</td>
19                    <td>{{sites.IuB_VLAN_id}}</td>
20                    <td>{{sites.Abis_VLAN_id}}</td>
21                    <td>{{sites.S1X2_VLAN_id}}</td>
22                   </tr>
23                   {% endfor %}
24               </table>
25           </table>
26           </div>
```

La boucle for entre les pourcentages, est du jinja et elle sera utilisé plus tard lorsque nous utiliserons Flask afin d'utiliser les données de notre base de données.

Voici pour l'instant le rendu graphique de notre site :

FIGURE 5.10 – Premier rendu de notre interface Web

Voici le rendu graphique de l'interface web pour ajouter un site :

FIGURE 5.11 – Ajouter un site dans notre base de données à partir de l'interface Web

Tout nos codes HTML seront transcrits en l'annexe.

5.4.3 Liaison entre la base de données et l'interface Web

Maintenant que nous avons notre base de données prête à être utilisée, et notre interface Web prête à afficher les données, nous devons lier les deux. Nous pourrons aussi lier nos différentes pages Web.

Nous utilisons pour celà la librairie Flask de Python, plus précisemment la fonction **"Flask_SQLAlchemy"**.

Nous importons les librairies nécessaires à notre code Python :

```
from flask import Flask, render_template, redirect, url_for, request,
    jsonify,request
from flask_sqlalchemy import SQLAlchemy
from flask_mysqldb import MySQL
import MySQLdb.cursors
import mysql.connector
```

Le reste des librairies importées sera transcrit dans l'annexe avec le code complet.

5.4.3.1 Connexion avec la base de données

En utilisant**Flask_SQLAlchemy** nous connectons Python à notre base de données comme suit :

```
app = Flask(__name__, template_folder='template')
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'password'
app.config['MYSQL_DB'] = 'test'
app.config['SECRET_KEY'] = "PFE key"
mysql = MySQL(app)
```

Nous pouvons désormais exécuter n'importe quelle requête à partir de Python(Flask) vers notre base de données et en afficher le résultat.

Voici le code Python qui permet de réaliser la liaison entre notre base de données et nos différentes pages Web :

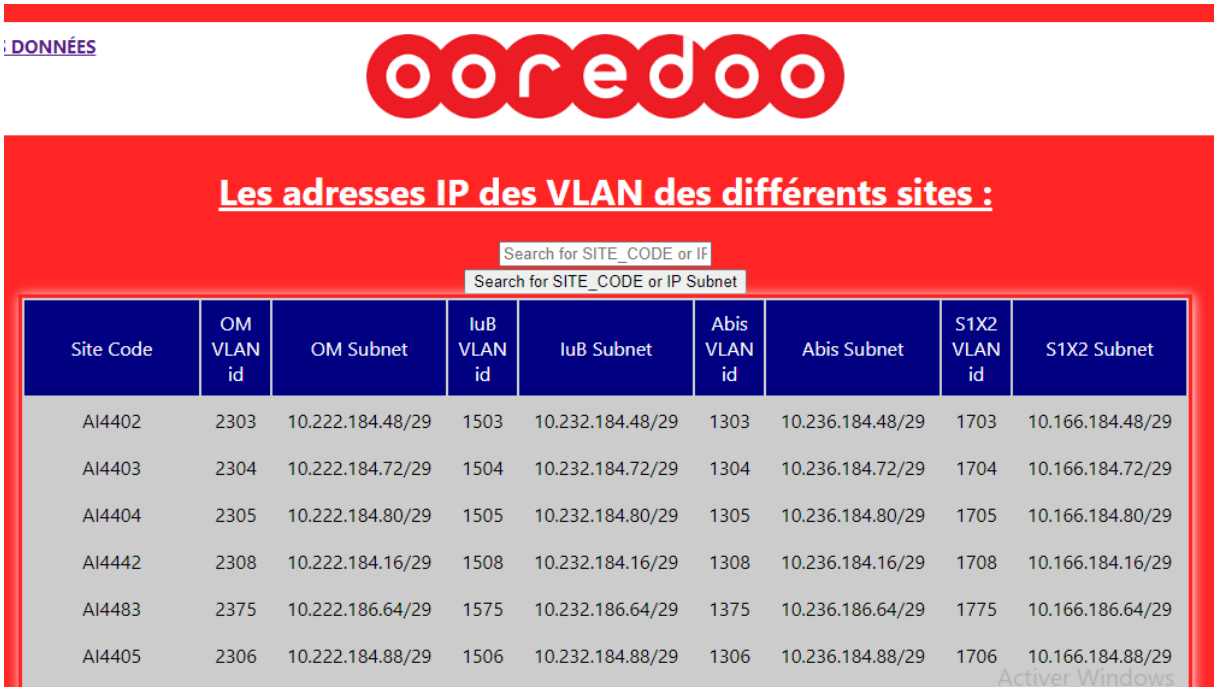
```
@app.route("/")
@app.route('/search', methods=["POST" , "GET"])
def search() :
    if request.method == "POST" :
        search = request.form["search"]
        cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute( 'SELECT * FROM sites WHERE SITE_CODE LIKE %s OR
    OM_Subnet LIKE %s OR IuB_Subnet LIKE %s OR Abis_Subnet LIKE %s OR
    S1X2_Subnet LIKE %s', ['%' + search + '%','%' + search + '%','%' +
    search + '%','%' + search + '%','%' + search + '%'])
        result = cursor.fetchall()
        return render_template("search.html", result=result)
    else :
        return render_template("search.html")

@app.route('/add', methods=["POST","GET"])
def add():
    if request.method == "POST" :
        a = request.form["siteadd"]
        b = request.form["omid"]
        c = request.form["omsubnet"]
        d = request.form["iubid"]
        e = request.form["iubsubnet"]
        f = request.form["abisid"]
```

```
22          g = request.form["abissubnet"]
23          h = request.form["s1x2id"]
24          i = request.form["s1x2subnet"]
25          cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
26          cursor.execute("""INSERT INTO sites (SITE_CODE,
27          OM_VLAN_id,
28          OM_Subnet,
29          IuB_VLAN_id,
30          IuB_Subnet,
31          Abis_VLAN_id,
32          Abis_Subnet,
33          S1X2_VLAN_id,
34          S1X2_Subnet) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)""",(a,b,c,
    d,e,f,g,h,i,))
35          mysql.connection.commit()
36          return("Site bien ajout !")
37      else :
38          return render_template("add.html")
```

Le reste du code sera transcrit en l'annexe.

Voici pour l'instant le rendu de notre interface Web totalement connectée à notre base de données :



NÉES

Les adresses IP des VLAN des différents sites :

AI4400

Search for SITE_CODE or IP Subnet

Site Code	OM VLAN id	OM Subnet	IuB VLAN id	IuB Subnet	Abis VLAN id	Abis Subnet	S1X2 VLAN id	S1X2 Subnet
AI4400	2301	10.222.184.0/29	1501	10.232.184.0/29	1301	10.236.184.0/29	1701	10.166.184.0/29

Entrez les données du site à ajouter

test

0

0

0

0

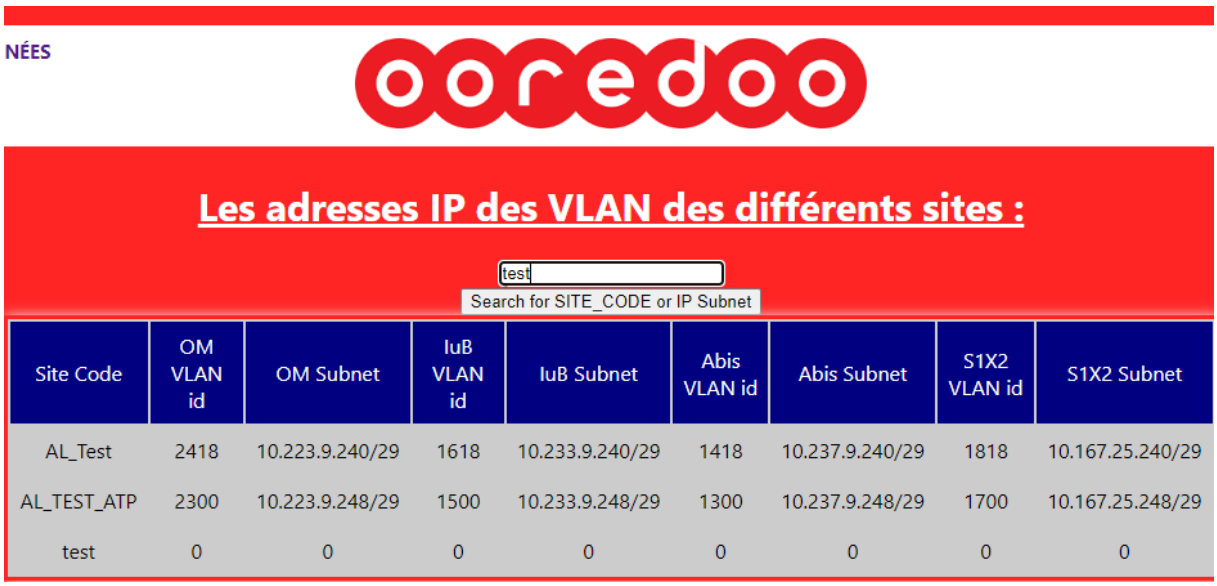
0

0

0

0

AJOUTER LE SITE



Site Code	OM VLAN id	OM Subnet	IuB VLAN id	IuB Subnet	Abis VLAN id	Abis Subnet	S1X2 VLAN id	S1X2 Subnet
AL_Test	2418	10.223.9.240/29	1618	10.233.9.240/29	1418	10.237.9.240/29	1818	10.167.25.240/29
AL_TEST_ATP	2300	10.223.9.248/29	1500	10.233.9.248/29	1300	10.237.9.248/29	1700	10.167.25.248/29
test	0	0	0	0	0	0	0	0

FIGURE 5.12 – Tests et résultats sur notre interface Web

Notre interface Web fonctionne comme souhaité et la liaison avec notre base de donnée a bien été réalisée.
Il est désormais possible de réaliser tout traitement à partir de l'interface Web.

5.4.4 Récupération des données en étant connecté au matériel

Nous passons maintenant à la partie de l'automatisation de notre travail. Le but est de nous connecter au matériel (aux routeurs) et d'exécuter des commandes bien précises dans des boucles temporelles, afin d'obtenir des informations voulues. C'est à dire qu'à chaque X période de temps, notre programme s'exécutera, et nous récolterons les données souhaitées.
Nous utiliserons ces informations à notre avantage afin de finalement ajouter des données à notre base de données s'il y a un changement noté sur le matériel.
Nous utiliserons pour celà 3 librairies réseau : **Paramiko, Netmiko et telnet**.
Ces librairies nous permettront de nous connecter au matériel via SSH ou telnet. Nous devrons cependant connaître le matériel sur lequel on travaille et tout types de mots de passe d'accès.
Dans le cas où nous utilisons les librairies **Netmiko/Paramiko**, nous divisons notre dossier en 3 fichiers pythons : un contenant les appareils auxquels on souhaiterait se connecter, un autre contenant les commandes que l'on souhaiterait exécuteret un dernier contenant notre main.

Voici un exemple de routeurs de type cisco auxquels on pourrait se connecter :

```
R1 = {
    'device_type': 'cisco_ios',
    'host':   '192.168.23.128',
    'username': 'test',
    'password': 'password',
    'port' : '5000',
    'secret': 'secret',}
R2 = {
    'device_type': 'cisco_ios',
    'host':   '192.168.23.1',
    'username': 'test',
    'password': 'password',
    'port' : '5001',
    'secret': 'secret',}
```

Il faut noter que les commandes à utiliser sont différentes d'un confectionneur de routeurs à un autre.

Les informations qui nous intéressent sont le hostname de l'interface (elle contient le site code) le VLAN ID ainsi que l'adresse IP de l'interface.

Pour les routeurs juniper il nous suffit d'exécuter la commande dans notre VRF :

```
show interfaces terse routing-instance IP_RAN
```

On obtient alors nos différentes interfaces ainsi que leur adresses IP :

Nous pouvons dès lors chercher plus de détails en utilisant la commande :

```
show interfaces description
```

```
{master}
EricSupp@BI0700_MX480_SR> show interfaces terse routing-instance IP_RAN
Interface               Admin Link Proto    Local                 Remote
ae41.1345               up    up   inet     10.234.97.105/29
                                    multiservice
ae41.1371               up    up   inet     10.236.97.249/29
                                    multiservice
ae41.1373               up    up   inet     10.234.98.65/29
                                    multiservice
ae41.1375               up    up   inet     10.234.98.81/29
                                    multiservice
ae41.1379               up    up   inet     10.234.98.113/29
                                    multiservice
ae41.1382               up    up   inet     10.234.98.137/29
                                    multiservice
ae41.1545               up    up   inet     10.230.97.105/29
                                    multiservice
ae41.1571               up    up   inet     10.232.97.249/29
                                    multiservice
ae41.1573               up    up   inet     10.230.98.65/29
                                    multiservice
ae41.1575               up    up   inet     10.230.98.81/29
                                    multiservice
ae41.1579               up    up   inet     10.230.98.113/29
                                    multiservice
ae41.1582               up    up   inet     10.230.98.137/29
                                    multiservice
ae41.1745               up    up   inet     10.164.97.105/29
                                    multiservice
ae41.1771               up    up   inet     10.166.97.249/29
                                    multiservice
ae41.1773               up    up   inet     10.164.98.65/29
                                    multiservice
```

FIGURE 5.13 – output de notre routeur juniper après commande sur l'IP_RAN

Sous condition que la description de l'interface contienne les données relatives au site.

```
                                ^
syntax error, expecting <command>.
EricSupp@BI0700_MX480_SR> show interfaces descriptions ae41.1375
Interface       Admin Link Description
ae41.1375       up    up   Abis_BI0779

{master}
EricSupp@BI0700_MX480_SR>
```

FIGURE 5.14 – description de l'interface

Comme nous pouvons le voir, la description de l'interface contient le site code (BI0779) et la technologie de l'interface (Abis dans notre cas donc 2G).

Il est alors très important qu'avec la configuration du matériel, l'ingénieur sur le terrain ajoute toutes ses informations à la description de l'interface afin de faciliter l'ajout automatique dans l'IP_PLAN.
Cependant à la recherche d'une commande qui pourrait nous donner tout ce que l'on souhaite comme informations nous avons trouvé :

```
show configuration interfaces
```

Dont l'output est :

```
{master}
EricSupp@BI0700_MX480_SR> show configuration interfaces ae41.1375
description Abis_BI0779;
vlan-id 1375;
family inet {
    address 10.234.98.81/29;
}
```

FIGURE 5.15 – output de la commande show configuration interfaces ae41.1375

Ce qui est exactement ce qu'on recherche comme informations.
Le code Python afin de pouvoir communiquer la commande à notre routeur Juniper :

```
import netmiko
import paramiko
from netmiko import ConnectHandler
from paramiko import SSHException
from devices import all_devices
from commands import all_commands
import getpass
import telnetlib
for devices in all_devices :
    net_connect = ConnectHandler(**devices, banner_timeout=200,
   blocking_timeout=20, timeout=100, session_timeout=60,encoding='utf-8')
    net_connect.enable()
    cmd = ["show interfaces terse routing-instance IP_RAN"]
    for show in cmd :
        output=net_connect.send_command(show)
        y.append(output)

for x in y :
    print(x)
```

Ce code permet seulement d'afficher l'output de notre routeur Juniper dans notre terminal Windows.

Concernant les routeurs Cisco. Il faut comme pour Juniper veiller à ajouter une description de la technologie et du site code aux interfaces.
Après notre implémentation de notre réseau sur GNS3 nous pouvons grâce à la commande :

```
show interfaces description
```

```
R6#show interfaces description
Interface                      Status         Protocol Description
Fa0/0                          up             up
Fa0/0.10                       up             up       Abis_TESTPFE
Fa0/0.20                       up             up       IuB_TESTPFE
Fa0/0.30                       up             up       S1X2_TESTPFE
Fa0/0.40                       up             up       OM_TESTPFE
Gi1/0                          up             up
Gi2/0                          up             up
Gi3/0                          admin down     down
Gi4/0                          admin down     down
Fa5/0                          admin down     down
Fa6/0                          admin down     down
Lo0                            up             up
```

FIGURE 5.16 – output de la commande show interfaces description sur GNS3

Cette commande nous permet en premier lieu d'identifier les interfaces utilisées par notre routeur tout en reconnaissant les technologies et le site code des interfaces.
Ces informations ne sont pas suffisantes, il nous faut aussi le VLAN ID des interfaces et surtout leur adresse IP.
Pour celà nous devrons utiliser une boucle qui doit exécuter pour chaque interface la commande suivante :

```
show run interface fa0/0.10
```

Nous visualiserons dans cet exemple seulement l'interface fa0/0.10, il suffira juste de trouver un moyen de changer les interfaces via une boucle afin de retrouver leur configuration, à moins qu'elle soit prédéfinie ce qui faciliterait énormément notre tâche.
Nous essayons d'exécuter notre programme Python avec la librairie telnetlib :

```
tn = telnetlib.Telnet('192.168.23.128', 5004)
tn.write(b"show interfaces description\n\r")
tn.write(b"")
while True:
    line = tn.read_until(b"\n")
```

```
R6#show run interface fa0/0.10
Building configuration...

Current configuration : 124 bytes
!
interface FastEthernet0/0.10
 description Abis_TESTPFE
 encapsulation dot1Q 10
 ip address 10.10.0.2 255.255.255.248
end

R6#show run interface fa0/0.20
Building configuration...

Current configuration : 123 bytes
!
interface FastEthernet0/0.20
 description IuB_TESTPFE
 encapsulation dot1Q 20
 ip address 10.11.0.2 255.255.255.248
end
```

FIGURE 5.17 – output de la commande show run interface fa0/0.10 sur GNS3

```
6       print(line)
7       if b'abcd' in line:
8           break
```

Voici le résultat que nous obtenons dans terminal Windows après avoir exécuté le programme :

```
22  tn = telnetlib.Telnet('192.168.23.128', 5004)
23  tn.write(b"show interfaces description\n\r")
24  tn.write(b"")
25  while True:
26      line = tn.read_until(b"\n")
27      print(line)
28      if b'abcd' in line:
29          breaks

PROBLEMS 13  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

b'R6#show interfaces description\r\n'
b'Interface                    Status         Protocol Description\r\n'
b'Fa0/0                        up             up       \r\n'
b'Fa0/0.10                     up             up       Abis_TESTPFE\r\n'
b'Fa0/0.20                     up             up       IuB_TESTPFE\r\n'
b'Fa0/0.30                     up             up       S1X2_TESTPFE\r\n'
b'Fa0/0.40                     up             up       OM_TESTPFE\r\n'
```

FIGURE 5.18 – output de notre programme Python en utilisant la librairie telnetlib

Comme nous pouvons le voir, il y a certes les données qui nous intéressent, cependant, il y a beaucoup de données non utilisables. De plus, il nous est impossible d'ajouter ce format de données directement à notre base de données.

Nous devons maintenant filter toutes les données en plus et mettre les données utiles dans un format qui nous permettrait d'ajouter les données à notre base de données en un seul clique.

5.4.5 Ajout des données récupérées à notre base de données

Au lieu d'utiliser la fonction print de Python qui affiche l'output de notre code dans le terminal Windows, nous devons trouver un moyen de mettre directement cet output dans un fichier csv(fichier utilisable par une base de données MySQL).
Pour celà, nous utiliserons le code suivant :

```
1
2 f = open('D:/pfe OOREDOO/PFE.csv','w')
3 writer = csv.writer(f,delimiter=' ', quoting=csv.QUOTE_MINIMAL, quotechar='
    ,')
4 tn = telnetlib.Telnet('192.168.23.128', 5004, timeout = 5)
5 tn.write(b"show interfaces description | include TESTPFE\n\r")
6 tn.write(b"show run interface fa0/0.10\n\r")
7 tn.write(b"show run interface fa0/0.20\n\r")
8 tn.write(b"show run interface fa0/0.30\n\r")
9 tn.write(b"show run interface fa0/0.40\n\r")
10 while True:
11     line = tn.read_until(b"\n")
12     print(line)
13     writer.writerow(str(line))
14     if b'abcd' in line:
15         breaks
16         tn.close()
```

Ce code nous permettra de créer un fichier .csv **PFE.csv** et d'y mettre directement l'output de nos commandes sur les routeurs.
Voici ce que l'on obtient :
Nous devons maintenant enlever les virgules en plus et diviser notre colonne en plusieurs de façon à pouvoir utiliser les colonnes qui nous intéressent.
Pour celà nous utiliserons les librairies regex, et pandas.
Voici le code avec regex afin de remplacer chaque suite de virgules par une seule virgule :

```
1 import re
2 pattern = re.compile('(\, ){2,}')
```

```
b'R6#show,,interfaces,,description,,|,,include,,TESTPFE\r\n'
b'Fa0/0.10,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,Abis_TESTPFE\r\n'
b'Fa0/0.20,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,IuB_TESTPFE\r\n'
b'Fa0/0.30,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,S1X2_TESTPFE\r\n'
b'Fa0/0.40,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,,,,,,,,,,,,,up,,,,,,,,,,,,,OM_TESTPFE\r\n'
b'R6#show,,run,,interface,,fa0/0.10\r\n'
b'Building,,configuration...\r\n'
b'Current,,configuration,,:,,124,,bytes\r\n'
b'interface,,FastEthernet0/0.10\r\n'
b',,description,,Abis_TESTPFE\r\n'
b',,encapsulation,,dot1Q,,10\r\n'
b',,ip,,address,,10.10.0.2,,255.255.255.248\r\n'
b'R6#show,,run,,interface,,fa0/0.20\r\n'
b'Building,,configuration...\r\n'
b'Current,,configuration,,:,,123,,bytes\r\n'
b'interface,,FastEthernet0/0.20\r\n'
b',,description,,IuB_TESTPFE\r\n'
b',,encapsulation,,dot1Q,,20\r\n'
b',,ip,,address,,10.11.0.2,,255.255.255.248\r\n'
b'R6#show,,run,,interface,,fa0/0.30\r\n'
b'Building,,configuration...\r\n'
b'Current,,configuration,,:,,124,,bytes\r\n'
b'interface,,FastEthernet0/0.30\r\n'
b',,description,,S1X2_TESTPFE\r\n'
b',,encapsulation,,dot1Q,,30\r\n'
b',,ip,,address,,10.12.0.2,,255.255.255.248\r\n'
b'interface,,FastEthernet0/0.40\r\n'
b',,description,,OM_TESTPFE\r\n'
b',,encapsulation,,dot1Q,,40\r\n'
b',,ip,,address,,10.13.0.2,,255.255.255.248\r\n'
```

FIGURE 5.19 – le fichier PFE.csv qui a été créé par notre code contenant l'output de nos commandes sur le routeur

```python
input_file = open('PFE.csv', 'r')
output_file = open('PFE_fixed.csv', 'w')
data = csv.reader(input_file)
writer = csv.writer(output_file,quoting=csv.QUOTE_ALL)# dialect='excel')
for line in data:
    line = str(line)
    new_line = re.sub(pattern, '.', line)
    writer.writerow(new_line)
input_file.close()
output_file.close()
```

On obtient alors :

```
1  b'R6#show,interfaces,description,|,include,TESTPFE\
2  b'Fa0/0.10,up,up,Abis_TESTPFE\r\n'
3  b'Fa0/0.20,up,up,IuB_TESTPFE\r\n'
4  b'Fa0/0.30,up,up,S1X2_TESTPFE\r\n'
5  b'Fa0/0.40,u,up,OM_TESTPFE\r\n'
6  b'R6#show,run,interface,fa0/0.10\r\n'
7  b'interface,FastEthernet0/0.10\r\n'
8  b'description,Abis_TESTPFE\r\n'
9  b'encapsulation,dot1Q,10\r\n'
10 b'ip,address,10.10.0.2,255.255.255.248\r\n'
11 b'R6#show,run,interface,fa0/0.20\r\n'
12 b'interface,FastEthernet0/0.20\r\n'
13 b'description,IuB_TESTPFE\r\n'
14 b'encapsulation,dot1Q,20\r\n'
15 b'ip,address,10.11.0.2,255.255.255.248\r\n'
16 b'R6#show,run,interface,fa0/0.30\r\n'
17 b'interface,FastEthernet0/0.30\r\n'
18 b'description,S1X2_TESTPFE\r\n'
19 b'encapsulation,dot1Q,30\r\n'
20 b'ip,address,10.12.0.2,255.255.255.248\r\n'
21 b'R6#show,run,interface,fa0/0.40\r\n'
22 b'interface,FastEthernet0/0.40\r\n'
23 b'description,OM_TESTPFE\r\n'
24 b'encapsulation,dot1Q,40\r\n'
25 b'ip,address,10.13.0.2,255.255.255.248\r\n'
```

FIGURE 5.20 – Les changements sur notre fichier.csv grâce à la librairie regex

Nous devons maintenant utiliser la librairie Panda afin de séparer notre unique colonne en plusieurs, tel que chaque virgule démarque une nouvelle colonne :

```
1 import pandas as pd
2 f= open ('D:/pfe OOREDOO/PFE_fixed.csv','r')
3 df = pd.read_csv(f,sep=",", encoding='cp1252')
4 print (df)
5 df.to_csv('FINALCSV.csv', index = False, header=True)
```

Voici ce que l'on obtient dans notre fichier .csv :

b'description	Abis_TESTPFE\r\n'		
b'encapsulation	dot1Q	10\r\n'	
b'ip	address	10.10.0.2	255.255.255.248\r\n'
b'R6#show	run	interface	fa0/0.20\r\n'
b'interface	FastEthernet0/0.20\r\n'		
b'description	IuB_TESTPFE\r\n'		
b'encapsulation	dot1Q	20\r\n'	
b'ip	address	10.11.0.2	255.255.255.248\r\n'
b'R6#show	run	interface	fa0/0.30\r\n'
b'interface	FastEthernet0/0.30\r\n'		
b'description	S1X2_TESTPFE\r\n'		
b'encapsulation	dot1Q	30\r\n'	
b'ip	address	10.12.0.2	255.255.255.248\r\n'
b'R6#show	run	interface	fa0/0.40\r\n'
b'interface	FastEthernet0/0.40\r\n'		
b'description	OM_TESTPFE\r\n'		
b'encapsulation	dot1Q	40\r\n'	
b'ip	address	10.13.0.2	255.255.255.248\r\n'

FIGURE 5.21 – Le fichier .csv final après traitement

On remarque qu'il est désormais beaucoup plus facile de tirer n'importe quel information souhaitée de notre fichier .csv et de l'ajouter directement à notre base de données. La prochaine étape consiste à chercher les données souhaitées et de les mettre dans un dernier fichier que l'on pourra utiliser afin d'actualiser notre base de données.
Nous réalisons un code Python afin de chercher n'importe quel information (l'adresse IP dans notre exemple) et de l'insérer dans un fichier final.csv afin d'être utilisé et actualiser notre base de données.

```
a='i p'
with open("FINALCSV.csv") as f_obj:
    f = open('final.csv','w')
    reader = csv.reader(f_obj, delimiter=';')
    for line in reader:      #Iterates through the rows of your csv
            if a in str(line):      #If the string you want to search is in
    the row
                print(line[2],line[3])
                writer = csv.writer(f,delimiter=',', quoting=csv.
   QUOTE_MINIMAL, quotechar=',')
                writer.writerow(str(line))
```

Ce code nous permet de rechercher un mot spécifique (i p) dans notre cas et d'écrire dans le dernier fichier.csv les colonnes qui nous intéressent (2 et 3 dans notre cas) et le résultat est :

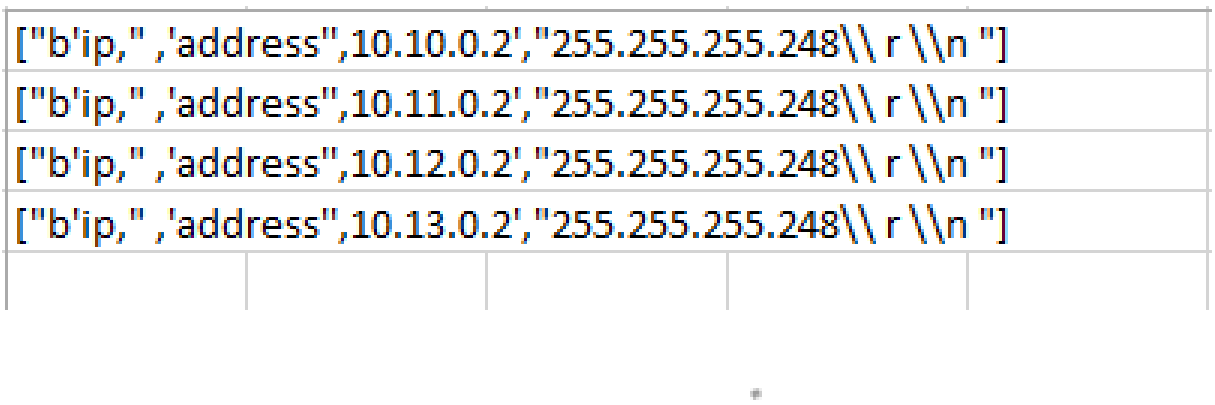


FIGURE 5.22 – Le fichier .csv prêt à être ajouté à notre base de données

Le fichier est alors prêt à être utilisé et on peut ajouter l'information utile directement à notre base de données comme nous l'avons fait dans la première partie de notre résolution.

Afin de pouvoir ajouter ces données à notre base de données, il faudra procéder de la même façon pour trouver le "site code" en remplaçant seulement le champ de recherche ("a" dans notre code) par les interfaces Abis, IuB, S1X2 et OM.

Voici finalement le site de notre implémentation sur GNS3, ajouté à notre base de données et interface Web.

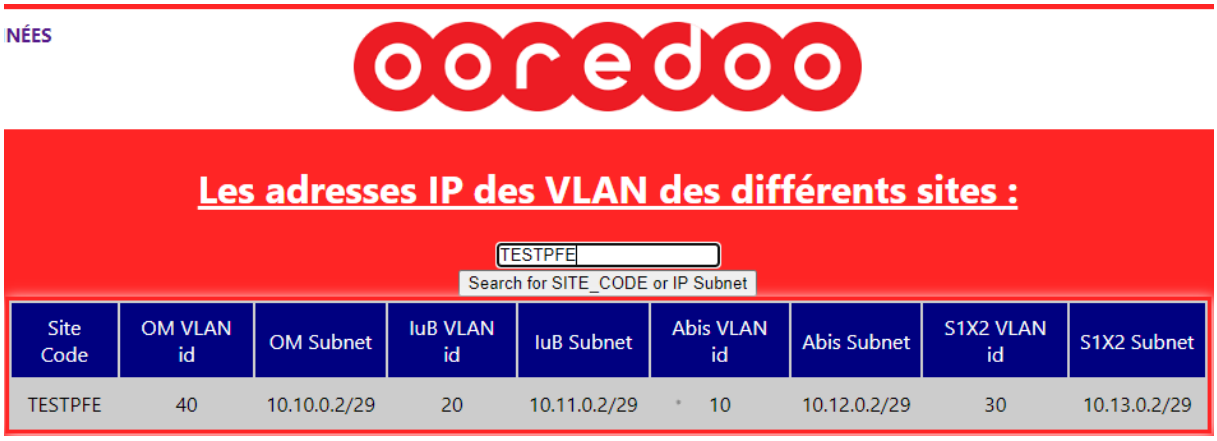


FIGURE 5.23 – L'ajout automatique à notre base de données et l'affichage sur l'interface Web

5.5 Conclusion

Lors de ce chapitre, nous avons développé la problématique à laquelle nous faisons face qui est le manque d'automatisation et de facilité d'accès à la donnée, tout en proposant et développant une solution.
Comme nous pouvons l'observer notre programme fonctionne bien comme on le souhaite, et nous avons pu atteindre notre objectif et régler les deux problèmes auxquels nous avont fait face.

Conclusion générale

Ce travail contribue à la convergence des opérateurs mobiles vers les réseaux IP/MPLS, qui sont beaucoup plus performants et rapides que les réseaux traditionnels, et à la décentralisation des données en encourageant à l'utilisation de base de données et l'exploitation des interfaces Web pour la gestion.
L'utilisation des émulateurs GNS3 et Eve-ng nous a permis de reproduire un réseau réel connectant divers abonnés utilisant des technologies d'accès différentes, de travailler et configurer sur différents équipements réseaux (routeurs, switch, Etherswitch) issue de différents équipementiers (Cisco et Juniper...). Celà nous a aussi permis d'en apprendre sur les infrastructures de télécommunications pour diverses technologies d'accès (2G 3G et 4G) avec les équipements associés (BTS, Node B, eNode B...).
Nous avons conclu que l'MPLS est une technologie qui est extrêmement bénéfique pour les entreprises. Elle simplifie l'infrastructure du réseau en permettant la consolidation de plusieurs technologies et applications telles que la voix et les données. L'MPLS offre une sécurité, une évolutivité et une haute disponibilité.
Nous avons ensuite exploité les différentes librairies réseaux de Python qui nous ont permis de faire abstraction de la complexité et des détails du matériel.
Puis, nous nous sommes intéressés à l'usage d'interfaces Web en faisant recours à d'autre librairies Python. Nous avons fait intéragir la base de données et l'interface Web, et modifier le second à partir du premier cité.
Nous avons finalement pu nous connecter à notre matériel une dernière fois grâce aux librairies réseau de Python, retirer et traiter les données qui nous intéressent et les ajouter à notre base de données sans avoir à le faire manuellement. Voici finalement un organigramme résumant notre travail :

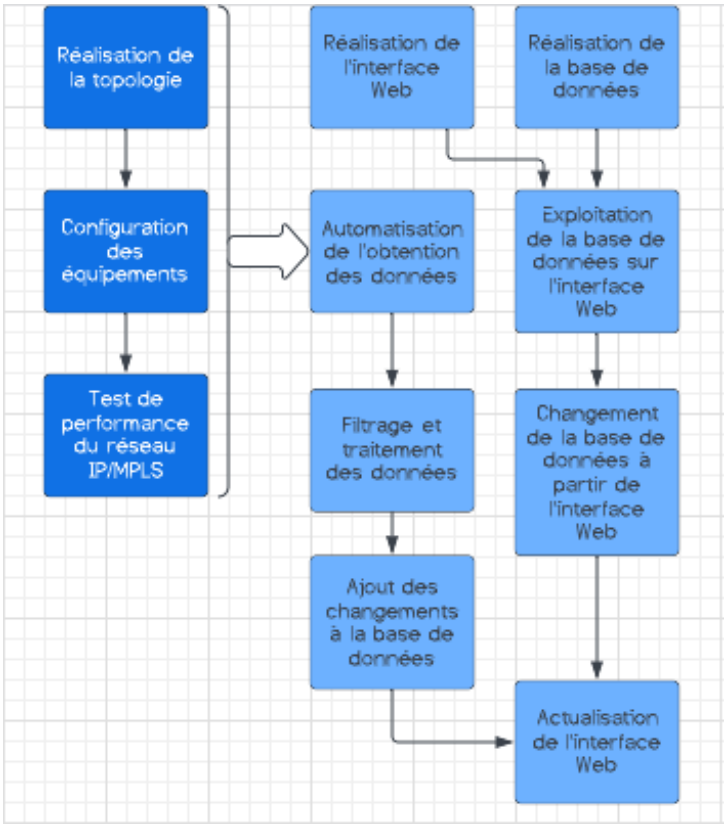


FIGURE 5.24 – Organigramme de la démarche suivie lors du projet de fin d'étude

Il convient de souligner les limites causées par les différentes librairies et le hardware. Pour pallier à ce problème, nous avons dû avoir recours aux serveurs d'OOREDOO pour utiliser le logiciel Eve-NG. Néanmoins, notre travail est largement suffisant pour l'application demandée.

Par rapport aux perspectives de recherches futures, Le projet pourrait porter sur l'automatisation des réseaux afin de réduire au maximum possible l'intervention humaine et en allant encore plus loin, il est possible d'inclure l'analyse des données pour l'aide à la prise de décision afin d'optimiser les opérations de migration des sites.

Bibliographie

[1] ALVAREZ, Santiago. QoS for IP/MPLS Networks : QoS for IP/MPLS Networks. Cisco press, 2006.

[2] NEDYALKOV, Ivan et GEORGIEV, Georgi. Performance Comparison of IP Network Using MPLS and MPLS TE. In : 2021 12th National Conference with International Participation (ELECTRONICA). IEEE, 2021. p. 1-4.

[3] ODOM, Wendell. CCNA 200-301 Official Cert Guide, Volume 2. Cisco Press, 2019.

[4] ODOM, Wendell. CCNA 200-301 Official Cert Guide, Volume 1. Cisco Press, 2019.

[5] Document technique Orange developer : Généralités et architecture de la 4G.

[6] AUDIBERT, Laurent. Base de Données et langage SQL. Developpez. com.[En ligne]. Disponible sur : http ://laurent-audibert. developpez. com/Cours-BD/.[Consulté le : 06-août-2018], 2007.

[7] https ://www.fil.univ-lille1.fr/ routier/enseignement/licence/tw1/spoc/diapos/CSS-introduction.pdf

[8] Documentation de la librairie Flask de Python `https://flask.palletsprojects.com/en/2.1.x/`

[9] Documentation de la fonction FlaskMySQLALchemy de Flask. `https://flask-sqlalchemy.palletsprojects.com/en/2.x/`

[10] I. Hussain "Overview of MPLS technology and traffic engineering applications," Internet Technologies Divisions, Cisco Systems, USA.

[11] J. L. Marzo, E. Calle, C. Scoglio, and T. Anjali, "QoS online outing and MPLS multilevel protection : a survey," IEEE Communication Magazine, vol. 41, no. 10, pp. 126–132, October 2003.

[12] Introduction aux telecommunications : http://www.volle.com/ENSPTT/introtcom.htm

[13] Documentation de jinja2 : https://jinja.palletsprojects.com/en/3.1.x/intro/

[14] D. Adami, "A new ns2 module for the simulation of MPLS networks with point-to-multipoint LSPs support," IEEE International Conference on Communications (ICC 2009), Dresden, Germany, June 2009, pp. 1–5.

[15] B. Boudani, B. Cousin, C. Jawhar, and M. Doughan, "Multicast routing simulator over MPLS networks", Proceedings of the 36th Annual Simulation Symposium (ANSS'03), Orlando, Florida, March 2003, pp. 327–334.

[16] Kateryna Mariushkina Joel Pettersson Towards Automated Network Configuration Simulations In A Switched Ethernet Network Master's thesis in Communication engineering

[17] Paul MIHĂILĂ, Titus BĂLAN, Radu CURPEN, Florin SANDU Network Automation and Abstraction using Python Programming Methods, October 19, 2017.

[18] MUHAMMAD NAEEM ASLAM YASSAR AZIZ, Traffic Engineering with Multi-Protocol Label Switching Performance Comparison with IP networks https://www.diva-portal.org/smash/get/diva2:833436/FULLTEXT01.pdf

[19] Dr. Esma BENDIAB (ENP) Introduction aux Bases de Données.

[20] Documentation sur MySQL https://dev.mysql.com/doc/

[21] Documentation sur la librairie réseau de Python Netmiko. https://github.com/ktbyers/netmiko

[22] Univesité d'ElOued, cours réseaux informatiques.

[23] Mémoire de fin d'étude en réseaux et télécommunicatuons, Berkani Randa UMMTO, "Etude et simulation d'un réseau IP-MPLS sous GNS3".

[24] Université Aix-Marseille les réseaux informatiques.

[25] Mémoire en ligne : "Etude d'implémentation d'une solution VOIP sécurisée dans un réseau informatique d'entreprise. Cas de l'ISTA de Kinshasa", Denish TASHMINGA.

[26] GSM - The Base Station Subsystem(BSS) https://www.tutorialspoint.com/gsm/gsm_base_station_subsystem.htm

[27] Imran Ikram, Master Thesis in electrical engineering"Traffic Engineering with MPLS and QOS".

[28] Tutoriel CSS https://www.w3schools.com/css/.

Annexes

.1 Annexe A : Code des pages Web en HTML et jinja

- Code de la page de recherche :

```
<!DOCTYPE html>
<html>
    <head>
        <link rel= "stylesheet" type= "text/css" href= "{{ url_for('static',filename='styles/style.css') }}">
        <title>IP PLAN</title>
    </head>
    <body>
        <header>
            <img src="{{url_for('static', filename='/img/Logo Oooredoo.png')}}" width="400" height="70" alt="Logo OOREDOO" class="logo_ooredoo">
        </header>


        <main>
            <nav class="sidenav">
                <ul class="link">
                    <li><a href={{url_for('acheminement')}}><b> L'Acheminement des donn es</b> </a> </li>
                    <li><a href="{{url_for('search')}}"><b> L'IP PLAN </b></a></li>
                    <li><a href="{{url_for('parameters')}}"><b>Changer les sites</b></a></li>
                </ul>
            </nav>
        <h1>Les adresses IP des VLAN des diff rents sites :</h1>
            <div class="search">
                <form action="{{ url_for('search') }}" method="POST">
                    <input type=text name="search" placeholder="Search for SITE_CODE or IP Subnet..."></br>
                    <div class="actions"><input type=submit value="Search for SITE_CODE or IP Subnet"></div>
                </form>
                <!-- <form action="{{ url_for('search') }}" method="POST">
                    <input type=text name="searchsub"></br>
                    <div class="actions"><input type=submit value="Search Subnet"></div>
```

```
30              </form> -->
31          </div>
32
33          <div class="Les_IP">
34              <table align = "center" border="0" style="box-shadow:1px 1
    px 10px white" width="70%" height="70%" text-align="center">
35                  <tr>
36                  <td style="background-position:center center" valign="
    top" text-align="center"  font-family: -apple-system, BlinkMacSystemFont
    , 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica
     Neue', sans-serif;>
37                   <table align="center" border="0" cellpadding="8px"
    width="100%" style="background-color:#CCCCCC">
38                      <tr style="background-color:navy;color:white" font-
    family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
    Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;>
39                      <td>Site Code</td>
40                      <td>OM VLAN id</td>
41                      <td>OM Subnet</td>
42                      <td>IuB VLAN id</td>
43                      <td>IuB Subnet</td>
44                      <td>Abis VLAN id</td>
45                      <td>Abis Subnet</td>
46                      <td>S1X2 VLAN id</td>
47                      <td>S1X2 Subnet</td>
48                   </tr>
49                    {% for sites in result %}
50                  <tr>
51                   <td>{{sites.SITE_CODE}}</td>
52                   <td>{{sites.OM_VLAN_id}}</td>
53                   <td>{{sites.OM_Subnet}}</td>
54                   <td>{{sites.IuB_VLAN_id}}</td>
55                   <td>{{sites.IuB_Subnet}}</td>
56                   <td>{{sites.Abis_VLAN_id}}</td>
57                   <td>{{sites.Abis_Subnet}}</td>
58                   <td>{{sites.S1X2_VLAN_id}}</td>
59                   <td>{{sites.S1X2_Subnet}}</td>
60                  </tr>
61                  {% endfor %}
62              </table>
```

```
            </table>
            </div>
        </main>
    </body>
</html>
```

- Code de la page d'ajouts de site :

```
<!DOCTYPE html>
<html>
    <head>
        <link rel= "stylesheet" type= "text/css" href= "{{ url_for('static',filename='styles/style.css') }}">
        <title>Acheminement</title>
    </head>
    <body>
        <header>
            <img src="{{url_for('static', filename='/img/Logo Oooredoo.png')}}" width="400" height="70" alt="Logo OOREDOO" class="logo_ooredoo">
        </header>


        <main>
            <nav class="sidenav">
                <ul class="link">
                    <li><a href={{url_for('acheminement')}}><b> L'Acheminement des donn es</b> </a> </li>
                    <li><a href="{{url_for('search')}}"><b> L'IP PLAN </b></a></li>
                    <li><a href="{{url_for('parameters')}}"><b>Changer les sites</b></a></li>
                </ul>
            </nav>
        <h1>Entrez les donn es du site    ajouter</h1>
            <div class="search">
                <form action="{{ url_for('add') }}" method="POST">
                    <input type=text name="siteadd" placeholder="SITE CODE"></br>
                    <input type=text name="omid" placeholder="OM VLAN ID"></br>
                    <input type=text name="omsubnet" placeholder="OM Subnet "></br>
```

```
                    <input type=text name="iubid" placeholder="IuB VLAN ID"
    ></br>
                    <input type=text name="iubsubnet" placeholder="Iub
    Subnet"></br>
                    <input type=text name="abisid" placeholder="Abis VLAN
    ID"></br>
                    <input type=text name="abissubnet" placeholder="Abis
    Subnet"></br>
                    <input type=text name="s1x2id" placeholder="S1X2 VLAN
    ID"></br>
                    <input type=text name="s1x2subnet" placeholder="S1X2
    Subnet"></br>
                    <div class="actions"><input type=submit value="AJOUTER
    LE SITE">
                </form>
            </div>
        </main>
    </body>
</html>
```

.2 Annexe B : Code de la stylisation des pages Web en CSS

- Code de la page de recherche :

```
body      {
            background-color: rgba(255, 0, 0, 0.856);}

main{         position: relative;
    top: 40px;}
.logo_ooredoo {
             max-width: 100%;}
header {
            position:absolute;
            top: 0px;
            left: 0px;
            right: 0px;
            text-align: center;
            background-color: white;
```

```
            padding: 10px ;}

h1 {font-size: 30px;
        text-decoration: underline;
        font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
    Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
        text-align: center;
        color: white;}

.link {font-size: 15px ;
        text-decoration: none;
        font-family :-apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
    Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
        text-transform: uppercase;
        position: relative;
        top: -30px;
        left: -40px;}
a {text-decoration: none;}

.IP {text-align: center;
        max-width: 100%;}
/* pseudo classes */

.link a:hover {
    text-decoration: underline;
    padding: 4px;
}

form input:focus {
    border: 1px dashed #4b4b4b;
}
```

.3 Annexe C : Code utilisée pour lier la base de données et nos pages Web en utilisant la librairie Flask de Python et ses fonctionnalitées

```
import imp
from multiprocessing import connection
```

```
import site
from flask import Flask, render_template, redirect, url_for, request,
    jsonify,request
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from pickle import GET
from sqlite3 import connect
from tokenize import String
from unittest import SkipTest, result
from colorama import Cursor
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired
from wtforms.widgets import TextArea
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re
import mysql.connector
from sqlalchemy import true


class SearchForm(FlaskForm):
    searched =  StringField("searched",validators=[DataRequired()] )
    submit = SubmitField("Submit")

app = Flask(__name__, template_folder='template')
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'password'
app.config['MYSQL_DB'] = 'test'
app.config['SECRET_KEY'] = "PFE key"
mysql = MySQL(app)
# @app.route("/")
# @app.route('/IP_PLAN',methods=['GET','POST'])
# def IP_PLAN():
#     cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
#     cursor.execute("SELECT * FROM sites")
#     data=cursor.fetchall()
#     return render_template("IP_PLAN.html",data=data)

```

```
@app.route("/")
@app.route('/search', methods=["POST" , "GET"])
def search() :
    if request.method == "POST" :
        search = request.form["search"]
        cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute( 'SELECT * FROM sites WHERE SITE_CODE LIKE %s OR OM_Subnet LIKE %s OR IuB_Subnet LIKE %s OR Abis_Subnet LIKE %s OR S1X2_Subnet LIKE %s', ['%' + search + '%','%' + search + '%','%' + search + '%','%' + search + '%','%' + search + '%'])
        result = cursor.fetchall()
        return render_template("search.html", result=result)
    else :
        return render_template("search.html")



@app.route('/acheminement', methods=["POST","GET"])
def acheminement():
        if request.method == "POST" :
            searchach = request.form["searchach"]
            cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute( 'SELECT * FROM ach WHERE SITE_CODE LIKE %s', ['%' + searchach + '%'])
            resultach = cursor.fetchall()
            return render_template("acheminement.html", resultach=resultach)
        else :
            return render_template("acheminement.html")

@app.route('/parameters', methods=["POST","GET"])
def parameters():
    return render_template("parameters.html")


@app.route('/add', methods=["POST","GET"])
def add():
    if request.method == "POST" :
        a = request.form["siteadd"]
        b = request.form["omid"]
```

```
77          c = request.form["omsubnet"]
78          d = request.form["iubid"]
79          e = request.form["iubsubnet"]
80          f = request.form["abisid"]
81          g = request.form["abissubnet"]
82          h = request.form["s1x2id"]
83          i = request.form["s1x2subnet"]
84          cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
85          cursor.execute("""INSERT INTO sites (SITE_CODE,
86          OM_VLAN_id,
87          OM_Subnet,
88          IuB_VLAN_id,
89          IuB_Subnet,
90          Abis_VLAN_id,
91          Abis_Subnet,
92          S1X2_VLAN_id,
93          S1X2_Subnet) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)""",(a,b,c,
    d,e,f,g,h,i,))
94          mysql.connection.commit()
95          return("Site bien ajout !")
96      else :
97          return render_template("add.html")
98
99  @app.route('/remove', methods=["POST","GET"])
100 def remove():
101     if request.method == "POST" :
102         retirer = request.form["siteremove"]
103         cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
104         cursor.execute("DELETE FROM sites WHERE SITE_CODE = %s",(retirer,))
105         mysql.connection.commit()
106         return("Site bien retir !")
107     else :
108         return render_template("remove.html")
109
110 @app.route('/change', methods=["POST","GET"])
111 def change():
112     if request.method == "POST" :
113         aa = request.form["sitechange"]
114         ba = request.form["omidc"]
115         ca = request.form["omsubnetc"]
```

```
116         da = request.form["iubidc"]
117         ea = request.form["iubsubnetc"]
118         fa = request.form["abisidc"]
119         ga = request.form["abissubnetc"]
120         ha = request.form["s1x2idc"]
121         ia = request.form["s1x2subnetc"]
122         cursor=mysql.connection.cursor(MySQLdb.cursors.DictCursor)
123         cursor.execute("""UPDATE sites set
124         OM_VLAN_id=%s,
125         OM_Subnet=%s,
126         IuB_VLAN_id=%s,
127         IuB_Subnet=%s,
128         Abis_VLAN_id=%s,
129         Abis_Subnet=%s,
130         S1X2_VLAN_id=%s,
131         S1X2_Subnet=%s
132         WHERE SITE_CODE=%s""",(ba,ca,da,ea,fa,ga,ha,ia,aa))
133         mysql.connection.commit()
134         return("Site bien chang !")
135     else :
136         return render_template("change.html")
137
138 if __name__== "__main__":
139     app.run(debug=true)
```

.4 Annexe D : Code des différentes librairies réseau en Python

- Code de la connexion au matériel en utilisant la librairie telnetlib :

```
1 from tkinter import Y
2 import netmiko
3 import paramiko
4 from netmiko import ConnectHandler
5 from getpass import getpass
6 from netmiko import NetMikoTimeoutException
7 from netmiko import NetMikoAuthenticationException
8 from netmiko import NetmikoTimeoutError
9 from netmiko import NetmikoAuthError
```

```
from netmiko.ssh_exception import NetMikoTimeoutException
from netmiko import NetMikoTimeoutException
from paramiko import SSHException
from paramiko.ssh_exception import SSHException
from netmiko.ssh_exception import AuthenticationException
from getpass import getpass
from datetime import datetime
from devices import all_devices
from commands import all_commands
import getpass
import telnetlib
import csv
import pandas as pd
import string
import re

f = open('D:/pfe OOREDOO/PFE.csv','w')
writer = csv.writer(f,delimiter=' ', quoting=csv.QUOTE_MINIMAL, quotechar='
    ,')
tn = telnetlib.Telnet('192.168.23.128', 5004, timeout = 5)
tn.write(b"show interfaces description | include TESTPFE\n\r")
tn.write(b"show run interface fa0/0.10\n\r")
tn.write(b"show run interface fa0/0.20\n\r")
tn.write(b"show run interface fa0/0.30\n\r")
tn.write(b"show run interface fa0/0.40\n\r")
while True:
    line = tn.read_until(b"\n")
    print(line)
    writer.writerow(str(line))
    if b'abcd' in line:
        breaks
        tn.close()
```

- Code de la connexion au matériel en utilisant la librairie netmiko :

```
from tkinter import Y
import netmiko
import paramiko
from netmiko import ConnectHandler
from getpass import getpass
from netmiko import NetMikoTimeoutException
from netmiko import NetMikoAuthenticationException
```

```
from netmiko import NetmikoTimeoutError
from netmiko import NetmikoAuthError
from netmiko.ssh_exception import NetMikoTimeoutException
from netmiko import NetMikoTimeoutException
from paramiko import SSHException
from paramiko.ssh_exception import SSHException
from netmiko.ssh_exception import AuthenticationException
from getpass import getpass
from datetime import datetime
from devices import all_devices
from commands import all_commands
import getpass
import telnetlib


for devices in all_devices :
    net_connect = ConnectHandler(**devices, banner_timeout=200,
   blocking_timeout=20, timeout=100, session_timeout=60,encoding='utf-8')
    net_connect.enable()
    cmd = ["show interfaces terse routing-instance IP_RAN"]
```

- Code Python contenant les informations des appareils auxquels on souhaiterait se connecter :

```
R1 = {
    'device_type': 'juniper',
    'host':   '10.50.170.24',
    'username': 'BI0700_MX480_SR',
    'password': 'test',
    'port' : '32774',
    'secret': 'test',
}
# R13 = {
#     'device_type': 'cisco_ios',
#     'host':   '192.168.23.1',
#     'username': 'test',
#     'password': 'password',
#     'port' : '5012',
#     'secret': 'secret',
# }
R14 = {
    'device_type': 'cisco_ios',
```

```
    'host':   '10.50.170.24',
    'username': 'test',
    'password': 'test',
    'port' : '32782',
    'secret': 'test',
}

all_devices = [
    R1,
    # R13,
    R14
]
```

• Code Python contenant les commandes que l'on souhaiterait exécuter sur nos appareils :

```

all_commands = ["show interfaces terse routing-instance IP_RAN",
                "show interfaces description | include TESTPFE",
                "show run interface fa0/0.10",
                "show run interface fa0/0.20",
                "show run interface fa0/0.30",
                "show run interface fa0/0.40",]
```

• Code Python afin de traiter nos fichiers .csv pour pouvoir les ajouter à notre base de données :

```
import csv
import pandas as pd
import string
import re

f = open('D:/pfe OOREDOO/PFE.csv','w')
writer = csv.writer(f,delimiter=' ', quoting=csv.QUOTE_MINIMAL, quotechar='
    ,')
tn = telnetlib.Telnet('192.168.23.128', 5004, timeout = 5)
tn.write(b"show interfaces description | include TESTPFE\n\r")
tn.write(b"show run interface fa0/0.10\n\r")
tn.write(b"show run interface fa0/0.20\n\r")
tn.write(b"show run interface fa0/0.30\n\r")
tn.write(b"show run interface fa0/0.40\n\r")
while True:
    line = tn.read_until(b"\n")
```

```
    print(line)
    writer.writerow(str(line))
    if b'abcd' in line:
        breaks
        tn.close()
    pattern = re.compile('(\, ){2,}')
    input_file = open('PFE.csv', 'r')
    output_file = open('PFE_fixed.csv', 'w')
    data = csv.reader(input_file)
    writer = csv.writer(output_file,quoting=csv.QUOTE_ALL)# dialect='excel
')
    for line in data:
        line = str(line)
        new_line = re.sub(pattern, '.', line)
        writer.writerow(new_line)

    input_file.close()
    output_file.close()

    f= open ('D:/pfe OOREDOO/PFE_fixed.csv','r')
    df = pd.read_csv(f,sep=",", encoding='cp1252')
    print (df)
    df.to_csv('FIINALCSV.csv', index = False, header=True)
    a='i p'
    with open("FINALCSV.csv") as f_obj:
        f = open('final.csv','w')
        reader = csv.reader(f_obj, delimiter=';')
        for line in reader:      #Iterates through the rows of your csv
                if a in str(line):      #If the string you want to search
 is in the row
                    print(line[2],line[3])
                    writer = csv.writer(f,delimiter=',', quoting=csv.
QUOTE_MINIMAL, quotechar=',')
                    writer.writerow(str(line))
```

.5 Annexe E : Configuration de nos appareils sur GNS3 :

- Switch ESW1 :

```
interface Vlan1
```

```
 no ip address
 no ip route-cache
 shutdown
!
interface Vlan10
 ip address 10.10.0.1 255.255.255.248
!
interface Vlan20
 ip address 10.11.0.1 255.255.255.248
!
interface Vlan30
 ip address 10.12.0.1 255.255.255.248
!
interface Vlan40
 ip address 10.13.0.1 255.255.255.248
!
ip forward-protocol nd
```

- Les Routeurs :

```


ip vrf PFE
 rd 10:10
 route-target export 10:10
 route-target import 10:10
!
!
!
no ip domain lookup
no ipv6 cef
!
multilink bundle-name authenticated

archive
 log config
  hidekeys
!
ip tcp synwait-time 5
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
```

```
 ip ospf 1 area 0
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex half
!
interface GigabitEthernet1/0
 ip address 10.0.0.1 255.255.255.0
 ip ospf 1 area 0
 negotiation auto
!
interface GigabitEthernet2/0
 ip vrf forwarding PFE
 ip address 10.0.3.1 255.255.255.0
 ip ospf 2 area 0
 negotiation auto
!
interface GigabitEthernet3/0
 no ip address
 shutdown
 negotiation auto
!
interface GigabitEthernet4/0
 no ip address
 shutdown
 negotiation auto
!
interface FastEthernet5/0
 no ip address
 shutdown
 duplex half
!
interface FastEthernet6/0
 no ip address
 shutdown
 duplex half
!
router ospf 2 vrf PFE
 log-adjacency-changes
```

```
 redistribute bgp 1 subnets
!
router ospf 1
 mpls ldp autoconfig
 log-adjacency-changes
!
router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
 neighbor 3.3.3.3 remote-as 1
 neighbor 3.3.3.3 update-source Loopback0
 no auto-summary
 address-family vpnv4
  neighbor 2.2.2.2 activate
  neighbor 2.2.2.2 send-community extended
  neighbor 3.3.3.3 activate
  neighbor 3.3.3.3 send-community extended
 exit-address-family
 address-family ipv4 vrf PFE
  no synchronization
 exit-address-family
```