

UNIVERSITÉ DE YAOUNDÉ I

**ÉCOLE NATIONALE SUPÉRIEURE
POLYTECHNIQUE**

***DÉPARTEMENT DES GÉNIES
ELECTRIQUE ET DES
TELECOMMUNICATIONS***

B.P. 8390 -Yaoundé - Cameroun
Tel. /Fax: +237 222 224 547



UNIVERSITY OF YAOUNDE I

**NATIONAL ADVANCED SCHOOL
OF ENGINEERING**

***DEPARTMENT OF ELECTRICAL
AND TELECOMMUNICATIONS
ENGINEERING***

P.O. Box - 8390 Yaoundé - Cameroun
Phone/Fax: +237 222 224 547



OPEN 5GS



➤ TCHAMKO CHABRELL

Table des matières

INTRODUCTION.....	4
I. RAPPELS THEORIQUES	5
1. 5G SA :.....	5
a. Architecture générale de la technologie 5G :.....	5
b. Options de déploiement de la 5G SA :	7
c. Avantages de la 5G SA :	8
II. Réseau 5G Autonome avec Open 5GS et UERANSIM.....	9
1. Les prérequis du projet.....	9
2. Installation de la machine virtuelle	9
3. But du Projet.....	13
4. Introduction à Open5GS	14
4.1 Installation de Open5GS.....	15
5. Introduction a UERANSIM.....	18
5.1 Installation de UERANSIM.....	18
5.2 Configuration de UERANSIM.....	20
III. TEST	27
CONCLUSION	30

Figure 1 : Architecture globale de la 5g-1.....	6
Figure 2 : Architecture globale de la 5g-2.....	6
Figure 3 : Option 2 de déploiement de la 5G	7
Figure 4 : Option 5 de déploiement de la 5G	7
Figure 5 : Installation du nouvelle machine virtuelle	9
Figure 6 : Configuration du nom du système.....	10
Figure 7 : Configuration de la mémoire et du processeur de la MV	10
Figure 8 : Configuration du disque virtuel de la machine virtuelle	11
Figure 9 : Ajout de l'image iOS ubuntu	11
Figure 10 : Création du nouveau NAT	12
Figure 11 : Ajout du nouveau NAT	12
Figure 12 : Adresse IP de Open5gs.....	13
Figure 13 : Ping entre Open5gs et UERANSIM	13
Figure 14 : Architecture du projet a simulée	14
Figure 15 : Fichier yaml des fonctions réseaux 4G et 5G.....	17
Figure 16 : Changement de l'adresse IP de l'AMF.....	18
Figure 17 : Paramètres de base de UERANSIM	19
Figure 18 : Modification du fichier open5gs-gnb. yaml.....	20
Figure 19 : Modification du fichier open5gs-ue. yaml.....	21
Figure 20 : Interface d'enregistrement.....	22
Figure 21 : Information de l'UE	23
Figure 22 : Enregistrement de l'UE.....	23
Figure 23 : UE enregistré	24
Figure 24 : Démarrage du gNB	24
Figure 25 : Démarrage de l'UE	25
Figure 26 : uesimtun0.....	25
Figure 27 : Ping Google	27
Figure 28 : Wireshark-ping google	28
Figure 29 : Ouverture de google	28
Figure 30 : Ping Youtube	29
Figure 31 : Wireshark-Ping YouTube	29
Figure 32 : Ouverture de YouTube	30

INTRODUCTION

Le déploiement des réseaux 5G représente une avancée majeure dans le domaine des communications sans fil. Pour faciliter la mise en œuvre et le test de ces réseaux, des projets open source tels que OPEN5GS et UERANSIM offrent des solutions complètes et flexibles.

Dans ce projet, nous explorons l'intégration de deux projets open source, OPEN5GS et UERANSIM, pour créer un environnement de test 5G complet et flexible. OPEN5GS fournit une suite complète de logiciels pour le déploiement d'un réseau cœur 5G, tandis que UERANSIM simule le réseau d'accès radio et l'UE. En combinant ces deux plateformes, nous créons un environnement de test évolutif et réaliste, permettant aux développeurs et aux chercheurs d'évaluer les performances, la gestion des abonnés et la mobilité dans un environnement contrôlé, sans avoir besoin d'une infrastructure matérielle complexe. Cette intégration accélère le développement et le déploiement des réseaux 5G, favorisant ainsi l'innovation dans le domaine des communications sans fil.

I. RAPPELS THEORIQUES

La 5G, ou cinquième génération de réseaux mobiles, représente une avancée majeure dans les communications sans fil, offrant des débits de données rapides, une latence réduite et une capacité accrue. Grâce à l'utilisation de fréquences plus élevées, le découpage du réseau, une latence réduite et des améliorations de capacité, la 5G ouvre la voie à de nouvelles applications telles que l'Internet des objets, la réalité virtuelle et les véhicules autonomes.

1. 5G SA :

La 5G SA (**Standalone**) fait référence à la 5G autonome, qui est la version complète et indépendante de la technologie 5G. Contrairement à la 5G non autonome (5G NSA), qui s'appuie sur l'infrastructure de réseau existante de la 4G LTE, la 5G SA est conçue pour fonctionner sur un réseau entièrement dédié à la 5G.

a. Architecture générale de la technologie 5G :

La 5G, contrairement à ses prédécesseuses, propose une virtualisation de son cœur de réseau permettant une amélioration conséquente des performances du réseau (vitesse ou débit, latence, densité d'équipements) ainsi qu'une 4ème partie appelée : réseau de données. Ces 04 grandes parties de l'architecture sont :

- **Le terminal ou UE (User Equipment) :** c'est l'appareil utilisé pour la communication mobile. Exemple : Smartphone, Ipad (tablette fonctionnant en Wi-Fi et réseau cellulaire)
- **Le réseau d'accès ou 5G NR-RAN (New Radio Radio Access Network) :** il fournit une connectivité sans fil aux UEs. Plus tard nous verrons qu'il est possible d'accéder au réseau de plusieurs manières.
- **Le cœur du réseau (Core Network) :** il coordonne les communications entre les UEs, alloue les ressources du réseau de manière efficace et garantit la sécurité des échanges de données. Il est également responsable du Network slicing qui sera vu plus tard.
- **Le réseau de données (Data Network) :** se charge du traitement de toutes les données qui sont émises au sein du réseau.

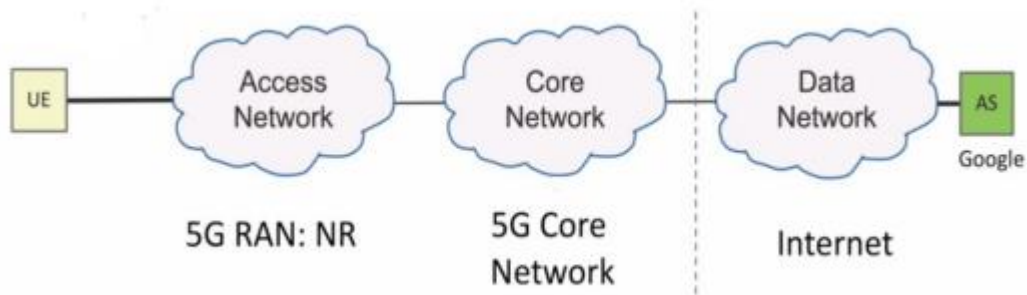


Figure 1 : Architecture globale de la 5g-1

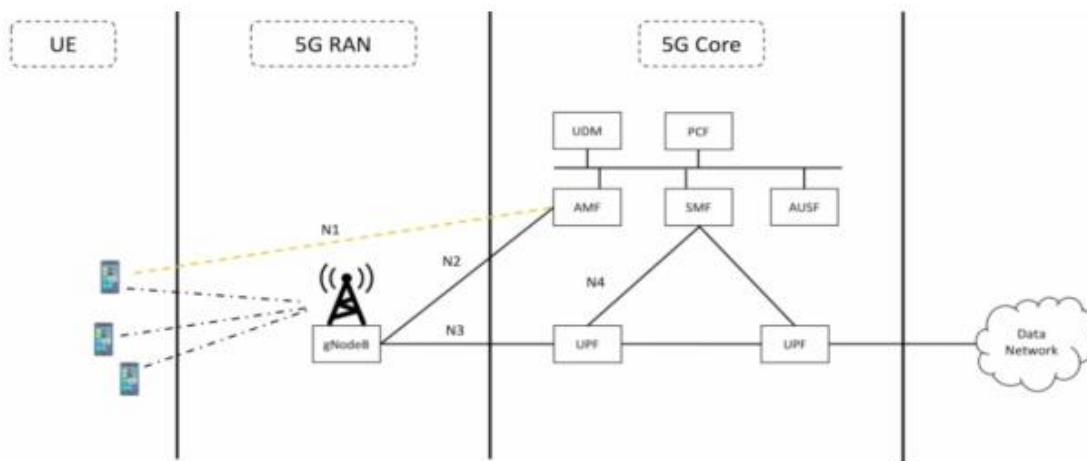


Figure 2 : Architecture globale de la 5g-2

- **AMF** - Access and Mobility Management Function (Fonction de gestion de l'accès et de la mobilité)
- **SMF** - Session Management Function (Fonction de gestion de session)
- **UPF** - User Plane Function (Fonction de plan d'utilisateur)
- **AUSF** - Authentication Server Function (Fonction de serveur d'authentification)
- **UDM** - Unified Data Management (Gestion des données unifiées)
- **PCF** - Policy and Charging Function (fonction de stratégie et de tarification)

b. Options de déploiement de la 5G SA :

La 3GPP (3rd Partnership Project) a décidé d'implémenter la technologie 5G SA de plusieurs façons. Ce sont les suivantes :

- **Option 2** : C'est l'architecture finale attendue pour le déploiement de la 5G. Elle est encore appelée « Full-5G » car elle fait intervenir un réseau d'accès 5G (gNodeB) et un cœur de réseau 5G (5G Core)

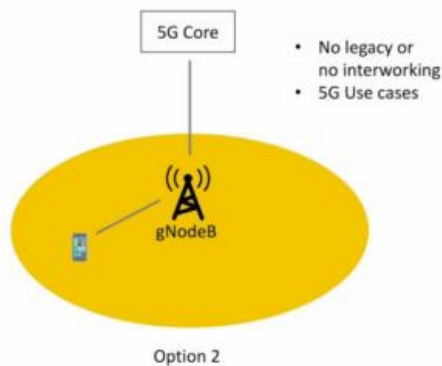


Figure 3 : Option 2 de déploiement de la 5G

- **Option 5** : Ici, on parle plus d'une option 5G « théorique » car on souhaite conserver les équipements 4G existants, en leur rajoutant des fonctionnalités pour qu'ils puissent supporter le cœur de réseau 5G. Le réseau d'accès est dit e-eNodeB (enhanced eNodeB).

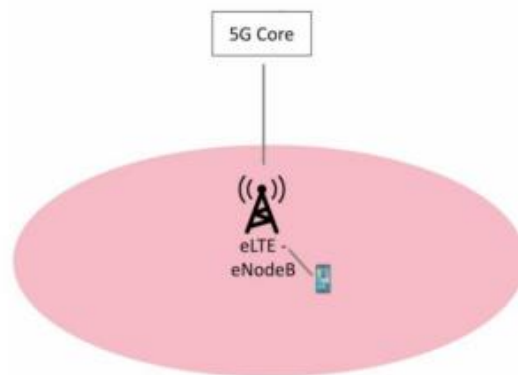


Figure 4 : Option 5 de déploiement de la 5G

c. Avantages de la 5G SA :

Considérée comme la 5G ultime, la 5G nouvelle radio standalone (NR SA) fournira :

- Une meilleure prise en charge de tous les cas d'utilisation et une libération de la puissance de la technologie mobile de la 5G NR.
- Des latences plus faibles, ainsi que de la possibilité de mettre en place la notion de Network slicing
- Des temps d'accès aux ressources radio ultra-rapides et par conséquent un accès plus rapide à des très haut débits pour les communications critiques et la garantie de la qualité de service inhérente à chaque service proposé aux usagers, mais aussi aux entreprises et aux acteurs industriels.
- La capacité et la couverture du réseau sont également plus élevées grâce à l'agrégation de porteuses (Carrier Aggregation) qui apporte une couverture à 25 % de personnes supplémentaires et qui augmente la capacité du réseau de 27 %

II. Réseau 5G Autonome avec Open 5GS et UERANSIM

1. Les prérequis du projet

Dans ce projet, nous allons utiliser **UERANSIM** pour simuler les **UE** et les **RAN 5G**, afin d'effectuer des appels de test à travers notre **Cœur du réseau 5G**. Et **OPEN5GS** pour simuler le cœur du réseau 5G. Nous avons utilisé deux machines virtuelles comme il est recommandé :

- ✓ Une machine Ubuntu 20.04 de RAM 2Go (au minimum) et Disque dure 20Go (au minimum) pour le cœur de réseau open5gs
- ✓ Une autre machine Ubuntu 20.04 de RAM 2Go et Disque dure 20Go pour installer l'UERANSIM

2. Installation de la machine virtuelle

Pour installer nos machines virtuelles nous avons utilisés Oracle VM VirtualBox Manager

- Pour créer une nouvelle machine virtuelle sur VirtualBox, il suffit de cliquer sur le bouton "**Nouvelle**" afin de lancer l'assistant de création d'une VM.



Figure 5 : Installation du nouvelle machine virtuelle

- On entre le nom du système d'exploitation que l'on veut installer (Ubuntu dans notre cas)

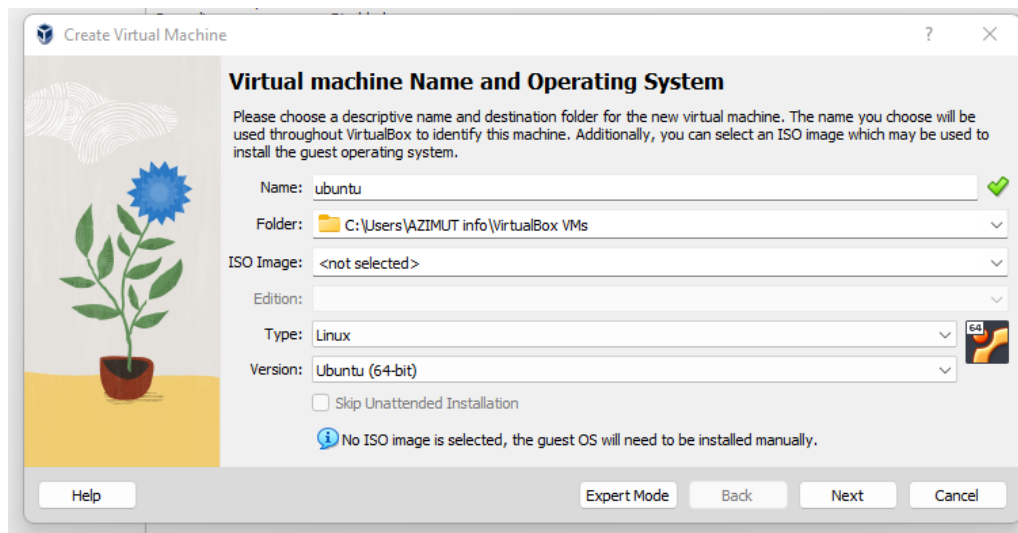


Figure 6 : Configuration du nom du système

- Maintenant l'on choisit la taille de la mémoire et le nombre de processeur

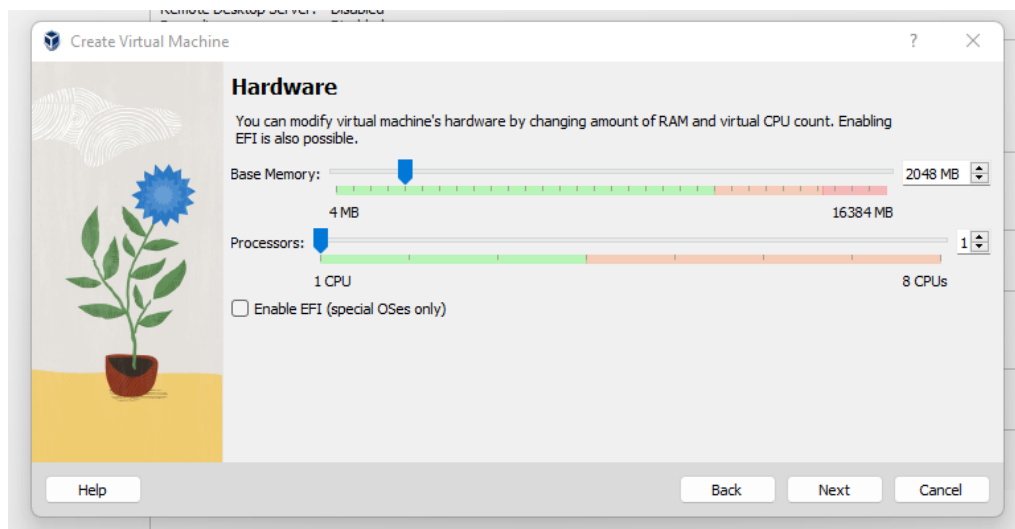


Figure 7 : Configuration de la mémoire et du processeur de la MV

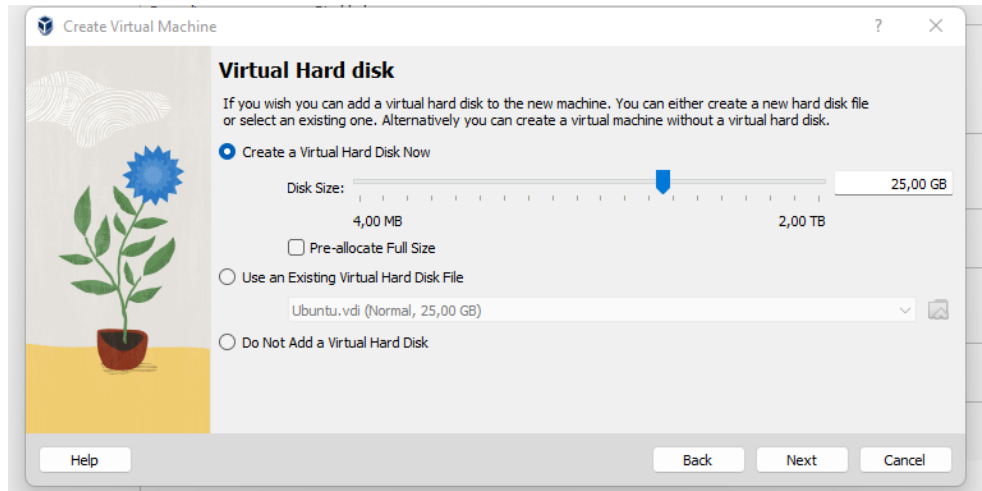


Figure 8 : Configuration du disque virtuel de la machine virtuelle

- Ensuite on ajoute l'image iOS et on démarre l'installation

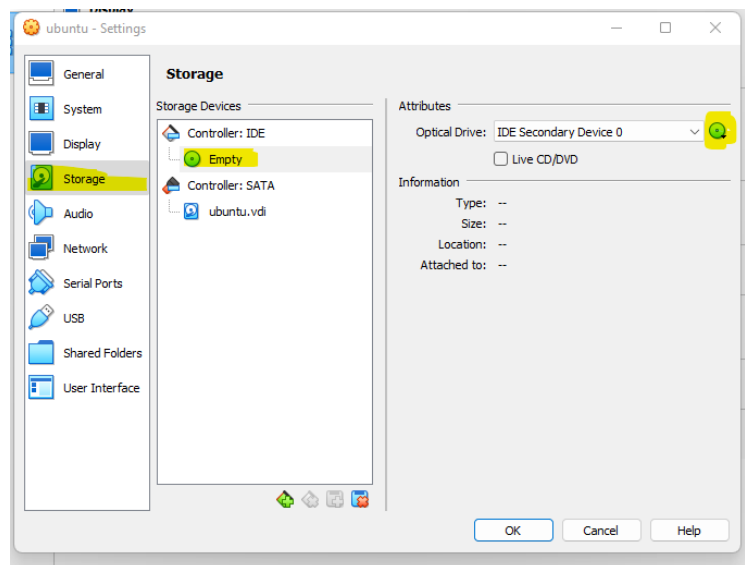


Figure 9 : Ajout de l'image iOS ubuntu

Un fois nos machines virtuelles installés, il faut les mettre dans le même sous-réseau pour établir la connexion entre eux.

Il faut donc :

- Créer un nouveau **NAT NETWORK** sur VirtualBox

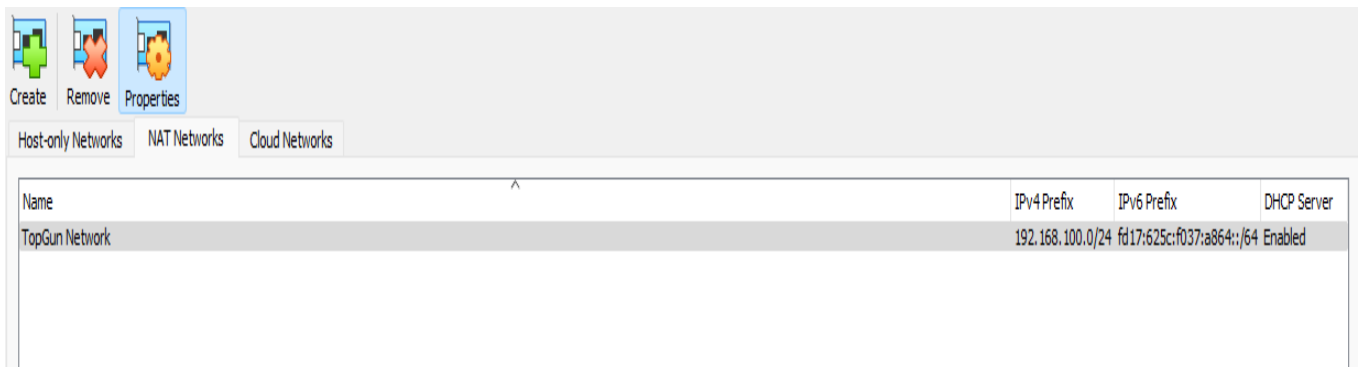


Figure 10 : Création du nouveau NAT

- Un fois le NAT ajouté, l'on ajoute ce NAT aux interfaces des machines

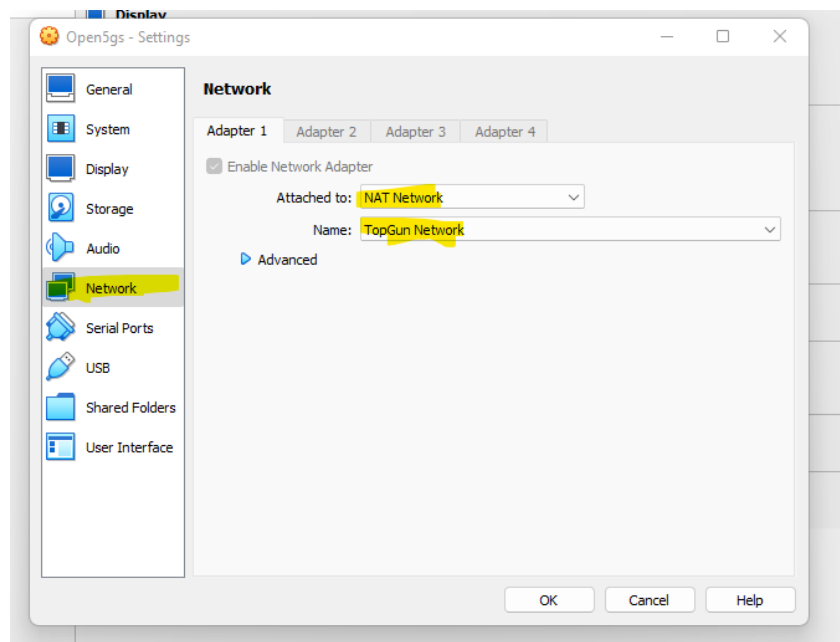


Figure 11 : Ajout du nouveau NAT

- En tapant **ifconfig** dans le terminal on a :

```
open5gs@open5gs-VirtualBox:/$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.4 netmask 255.255.255.0 broadcast 192.168.100.255
    inet6 fe80::61a:4f1b:e91:7e86 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:42:6e:b7 txqueuelen 1000 (Ethernet)
    RX packets 33509 bytes 26821631 (26.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 28188 bytes 2660219 (2.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 12 : Adresse IP de Open5gs

- **Open5GS : 192.168.100.4**
- **UERANSIM : 192.168.100.5**

```
open5gs@open5gs-VirtualBox:/$ ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5) 56(84) bytes of data.
64 octets de 192.168.100.5 : icmp_seq=1 ttl=64 temps=0.552 ms
64 octets de 192.168.100.5 : icmp_seq=2 ttl=64 temps=0.448 ms
64 octets de 192.168.100.5 : icmp_seq=3 ttl=64 temps=0.294 ms
64 octets de 192.168.100.5 : icmp_seq=4 ttl=64 temps=0.384 ms
64 octets de 192.168.100.5 : icmp_seq=5 ttl=64 temps=0.490 ms
^C
--- statistiques ping 192.168.100.5 ---
5 paquets transmis, 5 reçus, 0 % paquets perdus, temps 4092 ms
rtt min/moy/max/mdev = 0,294/0,433/0,552/0,088 ms
```

Figure 13 : Ping entre Open5gs et UERANSIM

Donc nos deux machines arrivent à communiquer

3. But du Projet

Le but du projet Open5GS en utilisant Ueransim est de fournir une solution open source complète pour la mise en place, la gestion et la simulation de réseaux 5G, permettant ainsi le développement et le déploiement d'applications et de services 5G de manière flexible et contrôlée.

4. Introduction à Open5GS

Avant de commencer, nous allons prendre un moment pour comprendre l'architecture de base du logiciel.

Open5GS est un projet open-source qui vise à fournir une solution de réseau de télécommunications 5G (et également 4G) complète et autonome. Il s'agit d'une implémentation logicielle de bout en bout du réseau de communication 5G, basée sur les spécifications du 3rd Generation Partnership Project (3GPP).

Architecture simulée

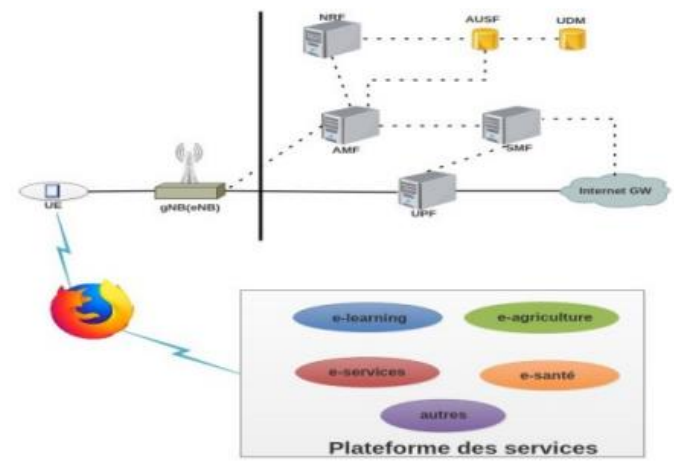


Figure 14 : Architecture du projet a simulée

Le cœur de réseau de la 5G SA fonctionne différemment de celui de la 4G. Il adopte une architecture basée sur les services (SBI). Les fonctions du plan de contrôle sont configurées pour s'enregistrer auprès du NRF (Network Repository Function), qui les aide ensuite à découvrir les autres fonctions du cœur.

Les différentes fonctions du cœur de réseau de la 5G SA interagissent de la manière suivante :

- L'**AMF** (*Access and Mobility Management Function*) gère la connexion et la mobilité des utilisateurs, un rôle similaire à celui du **MME** (*Mobility Management Entity*) dans la 4G. Les **gNB** (*les stations de base 5G*) se connectent à l'AMF.

- L'**UDM** (*Unified Data Management*), l'**AUSF** (*Authentication Server Function*) et l'**UDR** (*Unified Data Repository*) effectuent des opérations similaires à celles du **HSS** (*Home Subscriber Server*) dans la 4G. Ils génèrent des vecteurs d'authentification SIM et stockent les profils des abonnés.

- La gestion de session est entièrement prise en charge par le **SMF** (*Session Management Function*), qui était auparavant la responsabilité du MME/SGWC/PGWC dans la 4G.

- Le **NSSF** (*Network Slice Selection Function*) permet de sélectionner la tranche de réseau appropriée pour les services de l'abonné.

- Le **PCF** (*Policy Control Function*) est utilisé pour la facturation et l'application des politiques pour les abonnés.

En ce qui concerne le plan utilisateur du cœur de réseau de la 5G SA, il est beaucoup plus simple car il ne contient qu'une seule fonction. L'**UPF** (*User Plane Function*) transporte les paquets de données utilisateur entre le gNB et le réseau externe. Il est également connecté au SMF.

À l'exception du SMF et de l'UPF, les fichiers de configuration des fonctions centrales de la 5G SA ne contiennent généralement que les adresses IP/les noms d'interface locale de la fonction et l'adresse IP/le nom DNS du NRF.

4.1 Installation de Open5GS

- Installer Open5GS avec un gestionnaire de paquets **dépendances**

Open5gs@open5gs-VirtualBox: ~# **sudo apt-get update**

- Cette commande met à jour les informations sur les paquets disponibles dans les dépôts logiciels.

Open5gs@open5gs-VirtualBox: ~# **sudo apt-get install make g++ libsctp-dev lksctp-tools iproute2**

- Cette commande installe plusieurs paquets nécessaires pour la compilation et l'exécution d'Open5GS. Plus précisément, elle installe les paquets "make" (outil de compilation), "g++" (compilateur C++), "libsctp-dev" (bibliothèque SCTP pour la gestion des connexions), "lksctp-tools" (outils SCTP pour la configuration du protocole SCTP) et "iproute2" (utilitaires pour la gestion des routes réseau et des interfaces).

Open5gs@open5gs-VirtualBox: ~# snap refresh core (Si core n'est pas installé taper : **snap install core**)

- Cette commande met à jour le noyau (core) du système de gestion de paquets Snap.

Open5gs@open5gs-VirtualBox: ~# snap install cmake - -classic

- Cette commande installe le paquet "cmake" à l'aide du système de gestion de paquets Snap.

➤ Installation de open5gs avec les composants du cœur de 5G SA

Open5gs@open5gs-VirtualBox: ~# sudo apt update

- Cette commande met à jour les informations sur les paquets disponibles dans les dépôts logiciels.

Open5gs@open5gs-VirtualBox: ~# sudo apt install software-properties-common

- Cette commande installe le paquet "software-properties-common", qui est un ensemble d'utilitaires permettant de gérer les sources de logiciels dans Ubuntu.

Open5gs@open5gs-VirtualBox: ~# sudo add-apt-repository ppa:open5gs/latest

- Cette ligne de commande ajoute un référentiel (repository) PPA (Personal Package Archive) spécifique à Open5GS à la liste des sources de logiciels d'un système Ubuntu.

Open5gs@open5gs-VirtualBox: ~# sudo apt update

- Cette commande met à jour à nouveau les informations sur les paquets disponibles, après l'installation de "software-properties-common".

Open5gs@open5gs-VirtualBox: ~# sudo apt install open5gs

- Cette commande installe le paquet "open5gs", qui est le cœur de réseau 5G SA. Elle installe les composants nécessaires pour mettre en place un réseau 5G autonome.

Premièrement nous devons parler est l'**AMF**,

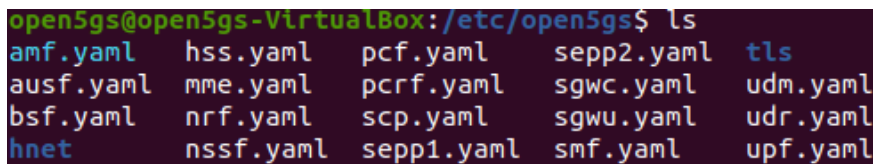
L'**AMF** (*Access and Mobility Function*) est accessible par le gNodeB via l'interface N2. Son rôle principal est de gérer la messagerie NAS (*Non-Access Stratum*) de la 5G. Cette messagerie est utilisée par les appareils/UE pour demander des services de données, gérer les transferts entre les gNodeB lors de leur déplacement dans le réseau, et s'authentifier auprès du réseau.

Par défaut, l'**AMF** est lié à une adresse IP de bouclage, ce qui fonctionne bien si tout est exécuté sur la même machine. Cependant, cela pose problème pour les gNodeB réels ou si nous exécutons **UERANSIM** sur une machine différente.

Ainsi, nous devons configurer l'**AMF** pour qu'il soit lié à l'adresse IP de la machine sur laquelle il fonctionne. Pour ce faire, nous devons modifier le fichier de configuration de l'**AMF**, situé dans **/etc/open5gs/amf.yaml**. En modifiant l'adresse **NGAP** dans ce fichier de configuration, nous pouvons lier l'**AMF** à l'adresse IP de la machine. Dans votre cas, l'adresse IP est **192.168.100.4**.

Open5gs@open5gs-VirtualBox : ~# **cd /etc/open5gs**

Open5gs@open5gs-VirtualBox/etc/open5gs : ~# **ls**

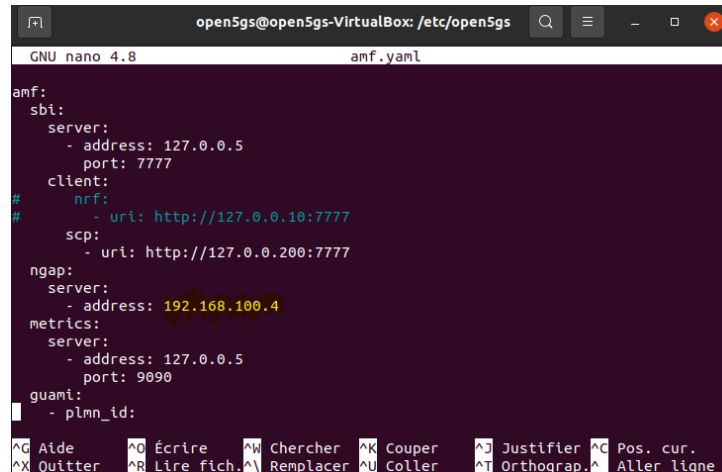


```
open5gs@open5gs-VirtualBox:/etc/open5gs$ ls
amf.yaml  hss.yaml  pcf.yaml  sepp2.yaml  tls
ausf.yaml mme.yaml  pcrf.yaml sgwc.yaml   udm.yaml
bsf.yaml  nrf.yaml  scp.yaml  sgwu.yaml   udr.yaml
hnet      nssf.yaml sepp1.yaml smf.yaml    upf.yaml
```

Figure 15 : Fichier yaml des fonctions réseaux 4G et 5G

Open5gs@open5gs-VirtualBox:/etc/open5gs\$# **sudo nano amf.yaml**

- Cette commande ouvre le fichier **amf.yaml** avec l'éditeur **nano** en mode superutilisateur et l'on peut maintenant le modifier



```
open5gs@open5gs-VirtualBox: /etc/open5gs
GNU nano 4.8 amf.yaml
amf:
  sbi:
    server:
      - address: 127.0.0.5
        port: 7777
    client:
      nrf:
        - uri: http://127.0.0.10:7777
      scp:
        - uri: http://127.0.0.200:7777
  ngap:
    server:
      - address: 192.168.100.4
  metrics:
    server:
      - address: 127.0.0.5
        port: 9090
  guami:
    - plmn_id:
```

Figure 16 : Changement de l'adresse IP de l'AMF

Pour permettre aux changements de prendre effet, nous allons redémarrer le service AMF d'Open5GS pour que nos changements prennent effet :

Open5gs@open5gs-VirtualBox: / etc/open5gs\$# systemctl restart open5gs-amfd.service

- Cette commande redémarre le service open5gs-amfd.

5. Introduction à UERANSIM

UERANSIM est un simulateur open source de l'infrastructure de réseau principal (core network) de la 5G (**3GPP Release 15**). Il permet de créer et de tester des scénarios de réseau 5G en émulant les différentes fonctions et entités du réseau central, telles que l'AMF, l'UPF, le SMF, le HSS, etc.

5.1 Installation de UERANSIM

Nous utilisons **UERANSIM** comme notre simulateur **UE & RAN**.

Ueransim@Ueransim-VirtualBox: ~# sudo apt-get update

Ueransim@Ueransim-VirtualBox: ~# sudo apt-get install make g++ libsctp-dev lksctp-tools iproute2

Ueransim@Ueransim-VirtualBox: ~# snap refresh core (Si core n'est pas installé taper : **snap install core**)

Ueransim@Ueransim-VirtualBox: ~# snap install cmake - --classic

Nous clonerons le dépôt Github, nous nous y installerons et nous ferons du code source.

Ueransim@Ueransim-VirtualBox: ~# git clone <https://github.com/aligungr/UERANSIM>

- Cette commande clone le dépôt Git hébergé sur GitHub à l'adresse spécifiée.

Ueransim@Ueransim-VirtualBox: ~# cd UERANSIM

- Cette commande permet de se déplacer dans le répertoire UERANSIM qui vient d'être cloné.

Ueransim@Ueransim-VirtualBox:/UERANSIM ~# make

- Cette commande exécute le fichier Makefile présent dans le répertoire UERANSIM. Le fichier Makefile contient des instructions pour la compilation et la construction du projet.

Maintenant, nous attendons que tout soit compilé. Une fois le logiciel installé, nous devons définir les paramètres de base.

Ueransim@Ueransim-VirtualBox:/UERANSIM ~# cd build

Ueransim@Ueransim-VirtualBox:/UERANSIM/build ~# ls -lh

```
ueransim@ueransim-VirtualBox:~/UERANSIM/build$ ls -lh
total 11M
-rwxrwxr-x 1 ueransim ueransim 16K May  6 14:48 libdevbnd.so
-rw-rw-r-- 1 ueransim ueransim 365 May  6 14:48 nr-binder
-rwxrwxr-x 1 ueransim ueransim 343K May  6 14:48 nr-cli
-rwxrwxr-x 1 ueransim ueransim 5.9M May  6 14:48 nr-gnb
-rwxrwxr-x 1 ueransim ueransim 4.4M May  6 14:48 nr-ue
ueransim@ueransim-VirtualBox:~/UERANSIM/build$
```

Figure 17 : Paramètres de base de UERANSIM

5.2 Configuration de UERANSIM

UERANSIM comporte deux parties essentielles, comme tout RAN :

- La première est le **gNodeB**, qui se connecte à notre **AMF** et gère le trafic des abonnés sur notre liaison radio (simulée)
- L'autre est nos abonnés eux-mêmes - les **UEs**.

Les deux sont définis et configurés par des fichiers de configuration dans le répertoire **config/**

– Configuration du gNodeB

Tous les paramètres de notre **gNodeB** sont définis dans le fichier **config/open5gs-gnb.yaml**.

A l'intérieur de ce fichier, nous devons définir les paramètres de notre gNodeB simulé, ce qui signifie pour nous (à moins que vous n'ayez changé le **PLMN** etc.) simplement changer les IPs de liaison auxquelles le gNodeB se lie, et l'IP des **AMFs** (pour nous c'est **192.168.100.4**)

Ueransim@Ueransim-VirtualBox:/UERANSIM ~# sudo nano config/open5gs-gnb.yaml

```
linkIp: 192.168.100.5 # gNB's local IP address for Radio Link Simulation (Usually same with local IP)
ngapIp: 192.168.100.5 # gNB's local IP address for N2 Interface (Usually same with local IP)
gtpIp: 192.168.100.5 # gNB's local IP address for N3 Interface (Usually same with local IP)

# List of AMF address information
amfConfigs:
- address: 192.168.100.4
  port: 38412
```

Figure 18 : Modification du fichier open5gs-gnb. yaml

- ❖ **linkIp** : 192.168.100.5 # adresse IP locale de la gNB pour la simulation de la liaison radio (généralement identique à l'IP locale)
- ❖ **ngapIp** : 192.168.100.5 # adresse IP locale de la gNB pour l'interface N2 (généralement identique à l'IP locale)
- ❖ **gtpIp** : 192.168.100.5 68 # adresse IP locale de la gNB pour l'interface N3 (généralement identique à l'IP locale)
- ❖ **address** : 192.168.100.4 # ip cœur réseau open5gs

– Configuration de l'UE

Nous devons indiquer à notre UE l'adresse IP (192.168.100.5) du gNodeB (en réalité, l'UE scannerait les bandes pour trouver un gNB qui le servirait, mais nous simulons ici).

Ueransim@Ueransim-VirtualBox:~/UERANSIM ~# sudo nano config/open5gs-ue.yaml

```
# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
- 192.168.100.5
```

Figure 19 : Modification du fichier open5gs-ue. yaml

Maintenant, nous devrions être en mesure de démarrer le service gNodeB et de voir la connexion. Mais avant il faut enregistrer l'UE au niveau de Open5GS sur notre navigateur, pour ça il faut :

WebUI vous permet de modifier de manière interactive les données des abonnés. Bien qu'il ne soit pas essentiel de l'utiliser, elle facilite les choses lorsque vous débutez dans l'aventure Open5GS. (Un outil en ligne de commande est disponible pour les utilisateurs avancés).

Node.js est nécessaire pour installer l'interface Web d'Open5GS.

Open5gs@open5gs-VirtualBox: ~# sudo apt update

- Cette commande met à jour la liste des paquets disponibles dans les dépôts apt.

Open5gs@open5gs-VirtualBox: ~# sudo apt install curl

- Cette commande installe le paquet curl en utilisant le gestionnaire de paquets apt. curl est une commande permettant de transférer des données à l'aide de divers protocoles, notamment HTTP et HTTPS.

Open5gs@open5gs-VirtualBox: ~# curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -

- Cette commande utilise curl pour télécharger un script à partir de l'URL https://deb.nodesource.com/setup_18.x et le pipe (|) envoie le contenu du script directement à l'interpréteur de commandes bash.

Open5gs@open5gs-VirtualBox: ~# sudo apt install nodejs

- Cette commande installe le paquet nodejs en utilisant le gestionnaire de paquets apt.

Open5gs@open5gs-VirtualBox: ~# curl -fsSL https://open5gs.org/open5gs/assets/webui/install | sudo -E bash –

- Cette commande utilise curl pour télécharger un script à partir de l'URL <https://open5gs.org/open5gs/assets/webui/install> et le pipe (|) envoie le contenu du script directement à l'interpréteur de commandes bash.

Après l'installation, vous allez accéder à **WebUI** via <http://localhost:9999/> avec :

- **Login** : admin
- **Password** :1423

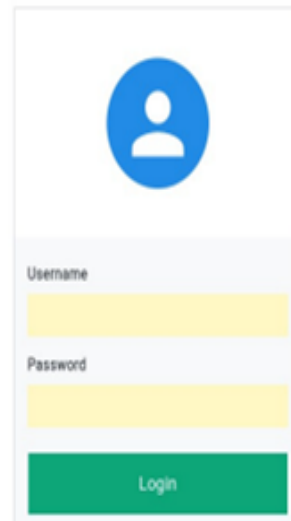
The image shows a web-based login interface. At the top, there is a blue circular icon containing a white silhouette of a person. Below this icon, there are two input fields: the first is labeled 'Username' and the second is labeled 'Password'. Both fields have yellow backgrounds. At the bottom of the form, there is a green rectangular button with the word 'Login' written in white text.

Figure 20 : Interface d'enregistrement

Puis on le renseigne comme suit en fonction des informations obtenues via la commande :

Ueransim@Ueransim-VirtualBox:/UERANSIM ~# sudo nano config/open5gs-ue.yaml

```

supi: 'imsi-999700000000001'
# Mobile Country Code value of HPLMN
mcc: '999'
# Mobile Network Code value of HPLMN (2 or 3 digits)
mnc: '70'
# SUCI Protection Scheme : 0 for Null-scheme, 1 for Profile A and 2 for Profile B
protectionScheme: 0
# Home Network Public Key for protecting with SUCI Profile A
homeNetworkPublicKey: '5a8d38864820197c3394b92613b20b91633cbd897119273bf8e4a6f4eec0a650'
# Home Network Public Key ID for protecting with SUCI Profile A
homeNetworkPublicKeyId: 1
# Routing Indicator
routingIndicator: '0000'
# Permanent subscription key

```

Figure 21 : Information de l'UE

Subscriber Configuration

IMSI*

999700000000001

+

Subscriber Key (K)*

465B5CE8B199B49FAA5F0A2EE238A6BC

Authentication Management Field (AMF)*

8000

USIM Type

OPc

Operator Key (OPc/OP)*

E8ED289DEBA952E4283B54E88E6183CA

UE-AMBR Downlink*

1

Unit

Gbps

UE-AMBR Uplink*

1

Unit

Gbps

Figure 22 : Enregistrement de l'UE

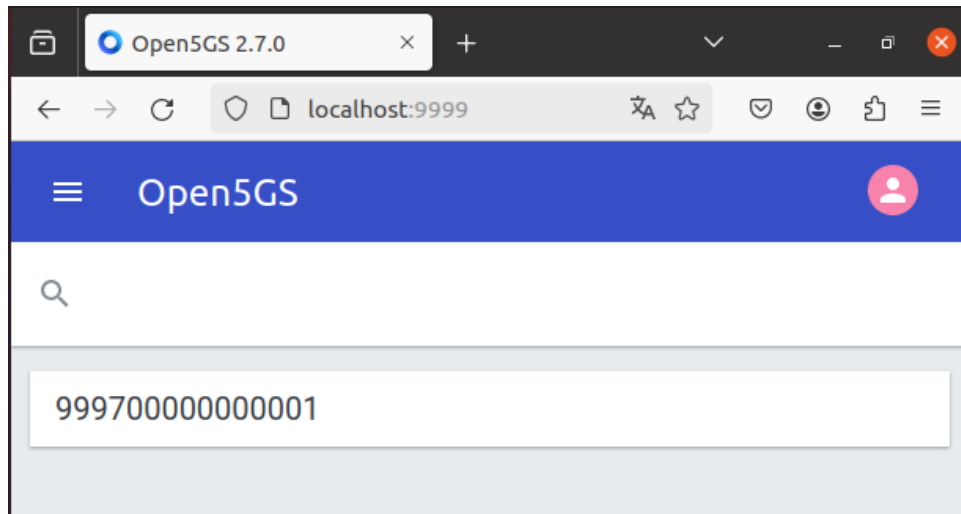


Figure 23 : UE enregistré

Une fois l'UE enregistré, nous démarrons le gNodeB et l'UE

➤ Démarrage des entités (gNB et UE)

○ Démarrage du nr-gNB

Ueransim@Ueransim-VirtualBox:/UERANSIM ~# build/nr-gnb -c config/open5gs-gnb.yaml

```
ueransim@ueransim-VirtualBox:~/UERANSIM$ build/nr-gnb -c config/open5gs-gnb.yaml
UERANSIM v3.2.6
[2024-05-07 01:15:36.055] [sctp] [info] Trying to establish SCTP connection... (192.168.100.4:38412)
[2024-05-07 01:15:36.070] [sctp] [info] SCTP connection established (192.168.100.4:38412)
[2024-05-07 01:15:36.070] [sctp] [debug] SCTP association setup ascId[16]
[2024-05-07 01:15:36.070] [ngap] [debug] Sending NG Setup Request
[2024-05-07 01:15:36.071] [ngap] [debug] NG Setup Response received
[2024-05-07 01:15:36.071] [ngap] [info] NG Setup procedure is successful
```

Figure 24 : Démarrage du gNB

Avec notre gNodeB "On the air", nous allons maintenant connecter un UE et à notre gNodeB simulé. Nous allons laisser le service nr-gnb en marche et ouvrir un nouveau terminal pour démarrer l'UE

○ Démarrage du nr-UE

Ueransim@Ueransim-VirtualBox:/UERANSIM ~# build/nr-ue -c config/open5gs-ue.yaml

```
Ueransim@Ueransim-VirtualBox:~/UERANSIM$ sudo build/nr-ue -c config/open5gs-ue.yaml
[sudo] Mot de passe de ueransim :
UERANSIM v3.2.6
[2024-05-06 23:16:45.612] [rrc] [debug] New signal detected for cell[1], total [1] cells in coverage
[2024-05-06 23:16:45.612] [nas] [info] UE switches to state [MM-DEREGISTERED/PLMN-SEARCH]
[2024-05-06 23:16:45.612] [nas] [info] Selected plmn[999/70]
[2024-05-06 23:16:45.612] [rrc] [info] Selected cell plmn[999/70] tac[1] category[SUITABLE]
[2024-05-06 23:16:45.612] [nas] [info] UE switches to state [MM-DEREGISTERED/PS]
[2024-05-06 23:16:45.612] [nas] [info] UE switches to state [MM-DEREGISTERED/NORMAL-SERVICE]
[2024-05-06 23:16:45.612] [nas] [debug] Initial registration required due to [MM-DEREG-NORMAL-SERVICE]
[2024-05-06 23:16:45.613] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_sig]
[2024-05-06 23:16:45.613] [nas] [debug] Sending Initial Registration
[2024-05-06 23:16:45.613] [nas] [info] UE switches to state [MM-REGISTER-INITIATED]
[2024-05-06 23:16:45.613] [rrc] [debug] Sending RRC Setup Request
[2024-05-06 23:16:45.613] [rrc] [info] RRC connection established
[2024-05-06 23:16:45.613] [rrc] [info] UE switches to state [RRC-CONNECTED]
[2024-05-06 23:16:45.613] [nas] [info] UE switches to state [CM-CONNECTED]
[2024-05-06 23:16:45.622] [nas] [debug] Authentication Request received
[2024-05-06 23:16:45.622] [nas] [debug] Received SQN [000000000141]
[2024-05-06 23:16:45.622] [nas] [debug] SQN-MS [000000000000]
[2024-05-06 23:16:45.634] [nas] [debug] Security Mode Command received
[2024-05-06 23:16:45.634] [nas] [debug] Selected integrity[2] ciphering[0]
[2024-05-06 23:16:45.646] [nas] [debug] Registration accept received
[2024-05-06 23:16:45.646] [nas] [info] UE switches to state [MM-REGISTERED/NORMAL-SERVICE]
[2024-05-06 23:16:45.646] [nas] [debug] Sending Registration Complete
[2024-05-06 23:16:45.646] [nas] [info] Initial Registration is successful
[2024-05-06 23:16:45.646] [nas] [debug] Sending PDU Session Establishment Request
[2024-05-06 23:16:45.647] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_sig]
[2024-05-06 23:16:45.851] [nas] [debug] Configuration Update Command received
[2024-05-06 23:16:45.860] [nas] [debug] PDU Session Establishment Accept received
[2024-05-06 23:16:45.875] [nas] [info] PDU Session establishment is successful PSI[1]
[2024-05-06 23:16:45.929] [app] [info] Connection setup for PDU session[1] is successful, TUN interface[uesimtun0, 10.45.0.10] is up.
```

Figure 25 : Démarrage de l'UE

Une fois la connexion bien établie, nous pouvons voir l'interface l'uesimtun0 dans le terminal :

Ueransim@Ueransim-VirtualBox: ~# ifconfig

```
uesimtun0: flags=369<UP,POINTOPOINT,NOTRAILERS,RUNNING,PROMISC> mtu 1400
inet 10.45.0.11 netmask 255.255.255.255 destination 10.45.0.11
inet6 fe80::bcf8:f76f:4522:a67f prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 6 bytes 400 (400.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 26 : uesimtun0

Avant les tests, nous allons activer le routage et le NAT sur nos deux machines virtuelles

➤ **Pour Open5G**

Open5gs@open5gs-VirtualBox: ~# echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

- Cette commande active le routage IP en écrivant la valeur 1 dans le fichier /proc/sys/net/ipv4/ip_forward

Open5gs@open5gs-VirtualBox: ~# sudo sysctl -p

- Cette commande recharge les paramètres système à partir des fichiers de configuration.

Open5gs@open5gs-VirtualBox: ~# sudo iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o ogstun -j MASQUERADE

- Cette commande ajoute une règle au tableau de filtrage NAT d'iptables. La règle spécifie que tout paquet provenant du sous-réseau 10.45.0.0/16 et qui ne sort pas par l'interface ogstun doit être soumis à une opération de masquerade (MASQUERADE).

Open5gs@open5gs-VirtualBox: ~# sudo iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o enp0s3 -j MASQUERADE

- Cette commande ajoute une autre règle au tableau de filtrage NAT d'iptables. La règle est similaire à la précédente, mais elle spécifie que les paquets sortants provenant du sous-réseau 10.45.0.0/16 et qui ne sortent pas par l'interface enp0s3 doivent subir une opération de masquerade (MASQUERADE).

➤ **Pour UERANSIM**

Ueransim@Ueransim-VirtualBox: ~# echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

Ueransim@Ueransim-VirtualBox: ~# sudo sysctl -p

Ueransim@Ueransim-VirtualBox : ~# sudo iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o uesimtun0 -j MASQUERADE

Ueransim@Ueransim-VirtualBox : ~# sudo iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o enp0s3 -j MASQUERADE

III. TEST

Ping de UE vers Google via son interface

Ueransim@Ueransim-VirtualBox: ~# ping 8.8.8.8 -I uesimtun0

```
ueransim@ueransim-VirtualBox:~$ ping 8.8.8.8 -I uesimtun0
PING 8.8.8.8 (8.8.8.8) from 10.45.0.8 uesimtun0: 56(84) bytes of data.
64 octets de 8.8.8.8 : icmp_seq=1 ttl=108 temps=186 ms
64 octets de 8.8.8.8 : icmp_seq=2 ttl=108 temps=207 ms
64 octets de 8.8.8.8 : icmp_seq=3 ttl=108 temps=131 ms
64 octets de 8.8.8.8 : icmp_seq=4 ttl=108 temps=148 ms
64 octets de 8.8.8.8 : icmp_seq=5 ttl=108 temps=116 ms
64 octets de 8.8.8.8 : icmp_seq=6 ttl=108 temps=115 ms
64 octets de 8.8.8.8 : icmp_seq=7 ttl=108 temps=114 ms
64 octets de 8.8.8.8 : icmp_seq=8 ttl=108 temps=138 ms
64 octets de 8.8.8.8 : icmp_seq=9 ttl=108 temps=160 ms
64 octets de 8.8.8.8 : icmp_seq=10 ttl=108 temps=115 ms
64 octets de 8.8.8.8 : icmp_seq=11 ttl=108 temps=202 ms
64 octets de 8.8.8.8 : icmp_seq=12 ttl=108 temps=121 ms
64 octets de 8.8.8.8 : icmp_seq=13 ttl=108 temps=115 ms
^C
--- statistiques ping 8.8.8.8 ---
13 paquets transmis, 13 reçus, 0 % paquets perdus, temps 12018 ms
rtt min/moy/max/mdev = 114,397/143,698/206,885/33,173 ms
```

Figure 27 : Ping Google

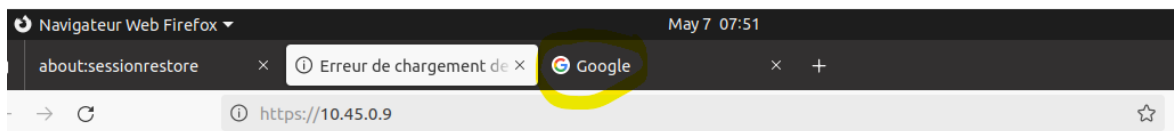
En lançant Wireshark en mode super utilisateur sur notre terminal, l'on peut voir comment se passe l'échange des paquets

Fichier Editer Vue Aller Capture Analyser Statistiques Telephonie Wireless Outils Aide						
Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=1/256, ttl=64 (reply in 2)
2	0.186476840	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=1/256, ttl=108 (request i...
3	1.001294214	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=2/512, ttl=64 (reply in 4)
4	1.280119665	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=2/512, ttl=108 (request i...
5	2.002045934	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=3/768, ttl=64 (reply in 6)
6	2.218533120	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=3/768, ttl=108 (request i...
7	3.001868419	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=4/1024, ttl=64 (reply in ...)
8	3.181101418	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=4/1024, ttl=108 (request ...)
9	4.002233661	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=5/1280, ttl=64 (reply in ...)
10	4.139652528	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=5/1280, ttl=108 (request ...)
11	5.002772256	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=6/1536, ttl=64 (reply in ...)
12	5.121775888	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=6/1536, ttl=108 (request ...)
13	6.002131014	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=7/1792, ttl=64 (reply in ...)
14	6.400289780	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=7/1792, ttl=108 (request ...)
15	7.003193854	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=8/2048, ttl=64 (reply in ...)
16	7.569438189	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=8/2048, ttl=108 (request ...)
17	8.003071022	10.45.0.9	8.8.8.8	ICMP	84	Echo (ping) request id=0x000f, seq=9/2304, ttl=64 (reply in ...)
18	8.515286287	8.8.8.8	10.45.0.9	ICMP	84	Echo (ping) reply id=0x000f, seq=9/2304, ttl=108 (request ...)
19	8.878416930	fe80::2e83:90ea:2ef... ff02::2		ICMPv6	48	Router Solicitation

Figure 28 : Wireshark-ping google

L'on peut aussi ouvrir google sur notre navigateur via l'interface Uesimtun0 avec la commande :

Ueransim@Ueransim-VirtualBox: ~# `firefox --bind-address= 10.45.0.9 https://www.google.com`



La connexion a échoué

Une erreur est survenue pendant une connexion à 10.45.0.9.

- Le site est peut-être temporairement indisponible ou surchargé. Réessayez plus tard ;
- Si vous n'arrivez à naviguer sur aucun site, vérifiez la connexion au réseau de votre ordinateur ;
- Si votre ordinateur ou votre réseau est protégé par un pare-feu ou un proxy, assurez-vous que Firefox est autorisé à accéder au Web.

Réessayer

Figure 29 : Ouverture de google

Ping de UE vers YouTube via son interface

```

ueransim@ueransim-VirtualBox:~$ ping youtube.com -I uesimtun0
PING youtube.com (216.58.215.46) from 10.45.0.8 uesimtun0: 56(84) bytes of data.
64 octets de par21s17-in-f14.1e100.net (216.58.215.46) : icmp_seq=1 ttl=110 temps=261 ms
64 octets de par21s17-in-f14.1e100.net (216.58.215.46) : icmp_seq=2 ttl=110 temps=268 ms
64 octets de par21s17-in-f14.1e100.net (216.58.215.46) : icmp_seq=3 ttl=110 temps=291 ms
64 octets de par21s17-in-f14.1e100.net (216.58.215.46) : icmp_seq=4 ttl=110 temps=313 ms
64 octets de par21s17-in-f14.1e100.net (216.58.215.46) : icmp_seq=5 ttl=110 temps=260 ms
^C
--- statistiques ping youtube.com ---
6 paquets transmis, 5 reçus, 16.6667 % paquets perdus, temps 5009 ms
rtt min/moy/max/mdev = 260,484/278,538/313,162/20,529 ms

```

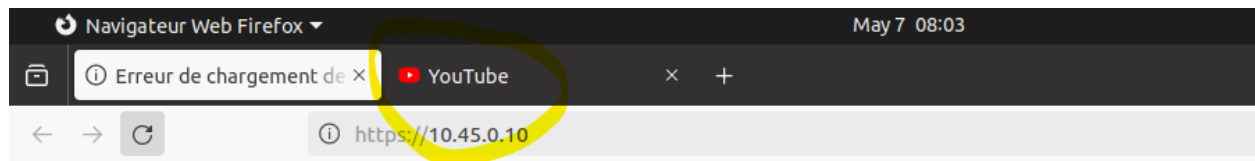
Figure 30 : Ping Youtube

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::2e9c:767c:86c...	ff02::2	ICMPv6	48	Router Solicitation
2	12.028632371	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=1/256, ttl=64 (reply in 3)
3	12.331712924	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=1/256, ttl=110 (request in ...)
4	13.030074545	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=2/512, ttl=64 (reply in 5)
5	13.327533408	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=2/512, ttl=110 (request in ...)
6	14.030756617	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=3/768, ttl=64 (reply in 7)
7	14.359619394	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=3/768, ttl=110 (request in ...)
8	15.030251822	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=4/1024, ttl=64 (reply in ...)
9	15.411664464	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=4/1024, ttl=110 (request in ...)
10	16.030562475	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=5/1280, ttl=64 (reply in ...)
11	16.414170957	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=5/1280, ttl=110 (request in ...)
12	17.030859423	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=6/1536, ttl=64 (reply in ...)
13	17.375704270	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=6/1536, ttl=110 (request in ...)
14	18.030619615	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=7/1792, ttl=64 (reply in ...)
15	18.364270412	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=7/1792, ttl=110 (request in ...)
16	19.031377814	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=8/2048, ttl=64 (reply in ...)
17	19.415346238	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=8/2048, ttl=110 (request in ...)
18	20.031562390	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=9/2304, ttl=64 (reply in ...)
19	20.439463755	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=9/2304, ttl=110 (request in ...)
20	21.048533406	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=10/2560, ttl=64 (reply in ...)
21	21.384664153	216.58.215.46	10.45.0.10	ICMP	84	Echo (ping) reply id=0x0010, seq=10/2560, ttl=110 (request in ...)
22	22.048972633	10.45.0.10	216.58.215.46	ICMP	84	Echo (ping) request id=0x0010, seq=11/2816, ttl=64 (reply in ...)

Figure 31 : Wireshark-Ping YouTube

L'on peut aussi ouvrir google sur notre navigateur via l'interface Uesimtun0 avec la commande :

Ueransim@Ueransim-VirtualBox: ~# firefox --bind-address= 10.45.0.10 https://www.youtube.com



La connexion a échoué

Figure 32 : Ouverture de YouTube

NB : A chaque fois que l'on redémarre le gNodeB et l'UE, l'adresse IP de uesimtun0 change.

CONCLUSION

En conclusion, notre projet Open 5GS avec l'utilisation d'UERANSIM a été une expérience enrichissante et fructueuse pour nous. Grâce à cette combinaison, nous avons pu explorer et étudier en profondeur les fonctionnalités et les capacités de la technologie 5G.

L'utilisation d'Open 5GS nous a permis de mettre en place une infrastructure de réseau 5G complète et de simuler un environnement réaliste pour tester et déployer diverses fonctionnalités. Nous avons pu configurer des abonnés, gérer les connexions, effectuer des appels et examiner les mécanismes de sécurité de la 5G. Cela nous a donné une compréhension pratique des protocoles et des procédures impliqués dans le déploiement d'un réseau 5G.

UERANSIM s'est révélé être un outil précieux pour simuler les équipements d'abonné. Grâce à ses fonctionnalités flexibles, nous avons pu créer et contrôler des utilisateurs virtuels, émuler leur comportement et évaluer les performances du réseau. Cela nous a permis de tester efficacement différentes configurations et scénarios, et d'optimiser nos stratégies de déploiement pour une meilleure performance et efficacité.

Ce projet nous a également permis de développer nos compétences en matière de gestion de projet, de résolution de problèmes et de collaboration. Travailler avec des technologies émergentes telles que la 5G et les outils open source comme Open 5GS et UERANSIM nous a donné une perspective précieuse sur les défis et les opportunités de l'avenir des réseaux de communication.