Centre de Recherche sur l'Information Scientifique et Technique

DIVISION SÉCURITÉ INFORMATIQUE

RAPPORT DE STAGE : DÉPLOIEMENT D'UN EPC SUR LE CLOUD

EFFECTUÉ PAR :

TOUMI Nassima

SOUS LA DIRECTION DE :

Dr. BENZAID Chafika,
Associée de Recherche

Table des matières

Chapitre 1

Introduction

1.1 La technologie 4G LTE :

Au vu des besoins grandissants en termes de disponibilité et de performances dans les réseaux de télécommunication, différentes technologies n'ont cessé d'être déployées afin de tenter de répondre à ces besoins : GSM, GPRS, EDGE, UMTS, HSPA, et plus récemment la 4G LTE.

La 4G est la technologie de quatrième génération de téléphonie, elle repose sur la norme LTE (Long Term Evolution) introduite par le consortium 3GPP.

En plus de l'augmentation significative des performances :

— Augmentation de la bande passante.
— Augmentation de la capacité.
— Diminution des temps de latence.
— Meilleure gestion de la mobilité des terminaux.

L'une des principales nouveautés de cette technologie par rappport aux précédentes consiste en l'utilisation de la technologie IP et de l'abandon des circuits commutés, les appels téléphoniques effectués à travers l'infrastructure 4G se font donc par le biais de la voix sur IP (VoIP) [1].

Comme tout réseau mobile, l'architecture d'un réseau 4G LTE comporte trois entités fonctionnelles [2] :

— Le réseau coeur EPC (Evolved Packet Core).
— Le réseau d'accès E-UTRAN.
— L'équipement utilisateur UE.

1.2 La Virtualisation des Fonctions Réseau (NFV) :

La NFV (Network Fuctions Virtualization) est une technologie qui permet aux fournisseurs de service de virtualiser les fonctionnalités logicielles des couches 4 à 7 du modèle IP en faisant abstraction du matériel.

De ce fait, les services réseau ne requièrent plus de machines dédiées et peuvent être déployées en tant que machines virtuelles sur des serveurs.
Ceci offre également aux fournisseurs une plus grande flexibilité, un meilleur contrôle des services réseau, l'optimisation de l'utilisation des ressources, en plus de la réduction des coûts de déploiement et de maintenance. [3]

1.3 Objectif du stage :

L'objectif de ce stage est de virtualiser les fonctions du réseau coeur de l'architecture 4G LTE (EPC) en le déployant sur un cloud.

Chapitre 2

Architecture 4G LTE

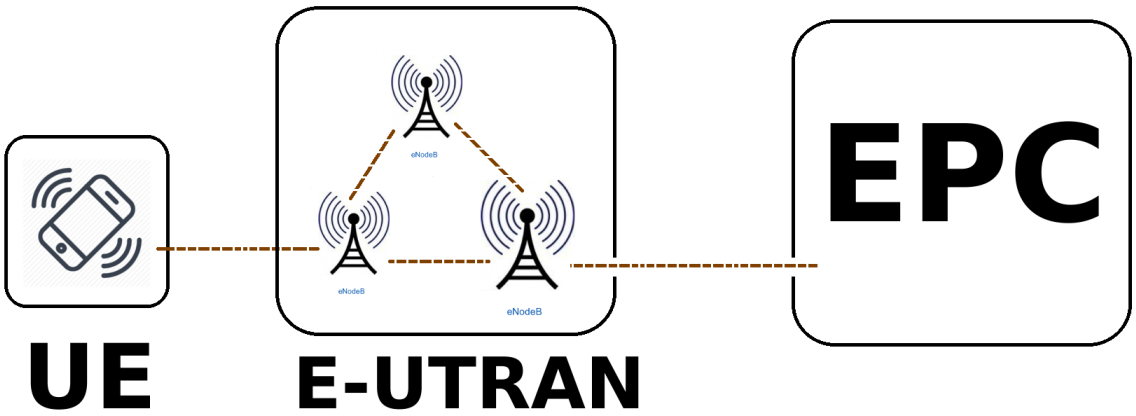


FIGURE 2.1 – Architecture d'un réseau 4G LTE

2.1 UE (User Equipment) :

Est le périphérique au moyen duquel l'utilisateur se connecte au réseau 4G.

2.2 E-UTRAN (Evolved UMTS Terrestrial Radio Access Network) :

Représente le réseau d'accès de l'architecture, son rôle est d'effectuer la transmission des communications radio entre les terminaux mobiles et le réseau coeur (EPC), le réseau E-UTRAN est constitué d'une interconnexion de stations de base (eNodeB) ; lors de sa connexion au réseau, chaque équipement utilisateur est affecté à l'eNB le plus proche. [2]

2.3 EPC (Evolved Packet Core) :

Également appelé SAE (System Architecture Evolution), représente le réseau coeur de l'architecture. Le rôle de l'EPC est d'assurer la gestion des utilisateurs, de la mobilité (handovers), de la qualité de service ainsi que de la sécurité au moyen de composants interconnectés via IP par différentes interfaces.

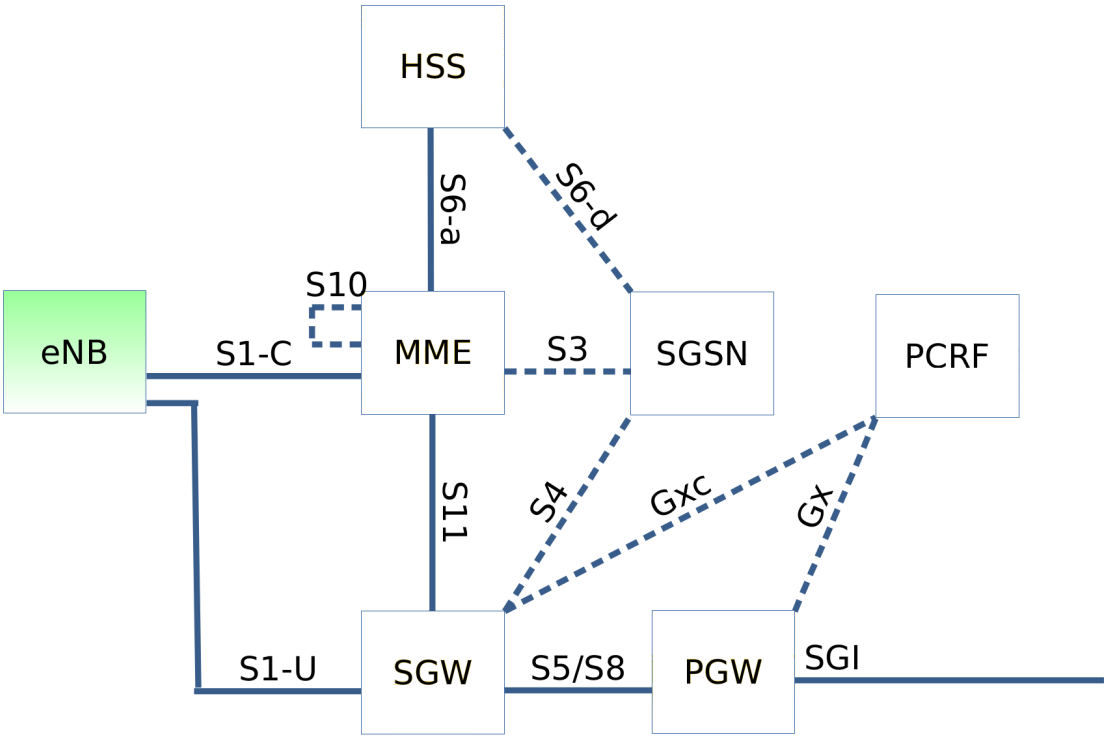


FIGURE 2.2 – Composants d'un EPC

2.3.1 Composants :

• **HSS (Home Subscriber Server) :** Représente la base de données du réseau, le HSS contient toutes les informations d'identification des abonnés, les profils de services fournis, ainsi que les informations de souscription des autres réseaux (3G, GSM, GPRS...). [4]

• **MME (Mobility Management Entity) :** Est l'élément de gestion et de contrôle de l'EPC, le MME effectue le suivi de la mobilité des périphériques, la gestion de la signalisation, l'authentification des abonnés et leurs autorisations, ainsi que la sélection des différentes passerelles en communiquant avec les autres composants.

• **P-GW (Packet Data Network Gateway) :** Représente la passerelle qui permet de connecter le EPC aux autres PDN (réseaux externes), assigne des adresses IP aux UE afin de leur permettre d'accéder à ces réseaux.

• **S-GW (Serving Gateway) :** Fonctionne en tant que routeur, effectue la transmission des paquets et les configuration des tunnels de transmission de données. Le S-GW joue le rôle d'*ancre de mobilité* (mobility anchor), si un UE de déplace et se connecte à un eNB différent, la session entre l'eNB et le S-GW est rompue et ce dernier se connecte au nouvel eNB, le périphérique reste donc connecté à la même passerelle ce qui permet de conserver la même adresse IP et d'éviter l'interruption de la session de l'UE.

Contient également une fonction d'application de politiques (PCEF) qui communique avec le PCRF et génère les CDRs (Charging Data Records) qui serviront à la tarification des communications. [5]

• **PCRF (Policy and Charging Control Element) :** Effectue la gestion des politiques de contrôle et de qualité de service (QoS) ainsi que la tarification, communique avec le PCEF du S-GW afin de bloquer ou d 'autoriser les flux, d 'affecter une QoS par flux, et de taxer chaque flux individuellement.[6]

• **SGSN (Serving GPRS Support Node) :** Représente la passerelle d'interconnexion avec les réseaux GSM, GPRS, EDGE, et UMTS.

2.3.2 Interfaces :

• **S1-C :** Permet de relier le E-UTRAN au MME, transmet les données générées par le protocole du plan de contrôle.

• **S1-U :** Relie le E-UTRAN au S-GW, transmet les données du plan utilisateur une fois le tunnel direct établi.

• **S3 :** Relie le MME au SGSN, permet la transmission des données utilisateur pour la mobilité inter-réseaux 3GPP.

• **S4 :** Relie le SGSN au S-GW, permet le support de mobilité entre les réseaux GPRS et la fonction d'ancrage du S-GW. Dans le cas où un tunnel direct n'est pas établi les données du plan utilisateur sont transmises via cette interface.

• **S5/S8 :** Relie le S-GW et le P-GW, fournit le support des services de paquets de données pour les utilisateurs finaux.

• **S6-A :** Relie le MME au HSS, permet la transmission des données concernant les abonnés pour l'authentification, l'identification des services supportés selon la location de la station mobile, ou la mise à jour de la base de données.

• **S6-D :** Relie le HSS au SGSN, remplit le même rôle que l'interface S6-A.

• **S10 :** Permet de relier les MME du EPC s'il en existe plusieurs, fournit le support de transfert de données utilisateur ainsi que la relocation entre MME.

• **S11 :** Relie le MME et le S-GW, fournit le support de la mobilité entre les deux composants, et permet la transmission des données du plan utilisateur.

• **Gxc :** Relie le PCRF au S-GW, permet le transfert des données des politiques de QoS ainsi que les règles de taxation, cette interface est utilisée uniquement dans le cas où l'interface S5/S8 est baséee PMIP (Proxy Mobile IP).

• **Gx :** Relie le PCRF au P-GW, remplit le même rôle que l'interface Gxc.

• **SGI :** Relie le P-GW aux réseaux extérieurs. [6]

2.3.3 Scénarios de fonctionnement :

Procédure d'attachement de l'UE :

Passe par deux phases :[5]

• **Authentification :**

— L'UE envoie une requête d'authentification au MME contenant les données de l'abonné, notamment l'IMSI (International Mobile Subscriber Identity).
— Le MME à son tour envoie une requête au HSS afin de vérifier l'identité de l'UE.
— Dans le cas où l'UE est authentifié, le MME répond à l'UE via l'eNB.

• **Mise en place d'un tunnel de transmission de données :**

— L'UE envoie au MME une requête de connexion à un réseau IP externe spécifique.
— Le MME envoie une requête au S-GW afin d'établir un tunnel pour l'UE.
— Le S-GW transmet cette information au P-GW qui alloue une adresse IP à l'UE et configure les tunnels de trafic ascendant et descendant entre le réseau externe et le S-GW, puis renvoie l'adresse IP au S-GW.
— Le S-GW configure le tunnel de trafic ascendant entre l'eNB à laquelle l'UE est attachée et le P-GW, contacte le PCRF afin de déterminer la politique de QoS à appliquer pour la session, et transfère l'adresse IP reçue au MME.
— Le MME transmet l'adresse IP à l'UE via l'eNB, l'eNB répond avec son identifiant de tunnel qui est transféré au S-GW pour la configuration du tunnel de trafic descendant.

Procédure de transfert de données via les tunnels :

• **Transfert ascendant (Uplink) :** UE => eNB => S-GW => P-GW => Réseau IP externe

• **Transfert descendant (Downlink) :** Réseau externe => P-GW => S-GW => eNB => UE

Procédure de détachement de l'UE :

- L'UE envoie une requête de détachement au MME.
- Le MME transfère la requête au S-GW afin de supprimer les tunnels de données.
- Le S-GW à son tour transmet le message au P-GW et supprime le tunnel descendant entre l'eNB et le P-GW ainsi que le tunnel ascendant entre le S-GW et le P-GW.
- Le P-GW supprime les identifiants de tunnel de la session en cours, détache l'adresse IP allouée et la replace dans la plage d'adresses IP disponibles.

Gestion d'interconnexion avec les réseaux de génération antérieure :

Au vu de la nouveauté de cette technologie, celle-ci se devait d'offrir un support d'interopérabilité avec les technologies antérieures telles que la 3G, UMTS...etc.

Dans le cas où le terminal supporte une technologie antérieure seulement, les procédures d'attachement, détachement et transfert de données se font via le SGSN au lieu de l'eNB.

Chapitre 3

Implémentation :

3.1 Outils logiciels employés :

Pour le déploiement de cette architecture, nous utiliserons les outils logiciels suivants :

3.1.1 Virtual Box :

Est une solution de virtualisation permettant de créer des machines virtuelles, de sauvegarder leurs états via snapshots...etc.
La machine virtuelle créée a les caractéristiques suivantes :

- **Système d'exploitation :** Ubuntu Server 14.04 64 bits.
- **CPUs alloués :** 3.
- **RAM allouée :** 6.5 Go.
- **Espace de stockage :** 100 Go.

3.1.2 Open Stack :

Est une solution de cloud computing open source qui permet de déployer des infrastructures *en tant que service*, et de gérer de larges plages de ressources de calcul, de stockage et de connectivité via une interface web.

L'installation d'Open Stack sur des distributions Debian telles que Ubuntu se fait à l'aide des scripts Devstack.

3.1.3 Open Air :

Est une implémentation open source des spécifications 3GPP concernant les composants de l'EPC, permet de déployer les fonctions du MME, HSS, S-GW et P-GW.

Cette solution est toujours en cours de développement et a été testée sur des machines Ubuntu 14.04 avec des noyeaux Linux 3.19 *low latency*.[7]

3.2 Installation et configuration du système d'exploitation :

— **Connectivité :** La VM doit avoir une adresse IP statique.

— **Logiciels :**

— **Interface graphique :** Permet d'ouvrir plusieurs terminaux afin d'ouvrir des sessions SSH et effectuer le suivi des différents composants de l'EPC. L'installation se fait avec la commande (afin de ne pas installer de composants inutiles) :

```
1 sudo apt-get install --no-install-recommends ubuntu-
    desktop
```

— **Firefox :** Afin d'accèder à l'interface *phpmyadmin* de la base de données du HSS.

— **Git :** Afin de télécharger les scripts d'installation de Devstack et de Open Air.

— **IPTables :** Afin de permettre aux instances créées dans Open Stack de se connecter à internet nous devons configurer le firewall avec la commande :

```
1 sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

3.3 Installation et configuration de l'environnement cloud Open Stack :

3.3.1 Installation :

Open Stack peut être installé sur les distributioons Debian avec les scripts Devstack, ils peuvent être téléchargés à partir du dépôt Git avec la commande :

```
1 git clone https://git.openstack.org/openstack-dev/devstack
```

Une fois téléchargés, un fichier de configuration doit être créé :

```
1 cd devstack
2 touch local.conf
3 chmod 664 local.conf
```

Avec le contenu suivant (La variable ADMIN_PASSWORD doit contenir le mot de passe administrateur de la VM) :

```
1 [[local|localrc]]
2 ADMIN_PASSWORD=<mot_de_passe>
3 SERVICE_TOKEN=$ADMIN_PASSWORD
```

```
4  MYSQL_PASSWORD=$ADMIN_PASSWORD
5  RABBIT_PASSWORD=$ADMIN_PASSWORD
6  SERVICE_PASSWORD=$ADMIN_PASSWORD
7  # Services
8  ENABLED_SERVICES=rabbit,mysql,key
9  ENABLED_SERVICES+=,n-api,n-crt,n-obj,n-cpu,n-cond,n-sch,n-novnc
      ,n-cauth
10 ENABLED_SERVICES+=,s-proxy,s-object,s-container,s-account
11 ENABLED_SERVICES+=,g-api,g-reg
12 ENABLED_SERVICES+=,cinder,c-api,c-vol,c-sch,c-bak
13 ENABLED_SERVICES+=,trove,tr-api,tr-tmgr,tr-cond
14
15 ENABLED_SERVICES+=,horizon
16 # Ceilometer
17 ENABLED_SERVICES+=,ceilometer-acompute,ceilometer-acentral,
      ceilometer-collector,ceilometer-api
18 ENABLED_SERVICES+=,ceilometer-alarm-notify,ceilometer-alarm-
      eval
19 # Heat
20 ENABLED_SERVICES+=,heat,h-api,h-api-cfn,h-api-cw,h-eng
21 # Neutron
22 DISABLED_SERVICES=n-net
23 ENABLED_SERVICES+=,q-svc,q-agt,q-dhcp,q-l3,q-meta,q-metering,
      neutron
24 # Neutron - Load Balancing
25 ENABLED_SERVICES+=,q-lbaas
26 # VLAN configuration
27 Q_PLUGIN=ml2
28 ENABLE_TENANT_VLANS=True
29 # GRE tunnel configuration
30 Q_PLUGIN=ml2
31 ENABLE_TENANT_TUNNELS=True
32 Q_ML2_TENANT_NETWORK_TYPE=gre
33 # Logging
34 LOGFILE=$DEST/logs/stack.sh.log
35 SCREEN_LOGDIR=$DEST/logs/screen
36 LOGDAYS=2
37 # Swift
38 SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5
39 SWIFT_REPLICAS=1
40 SWIFT_DATA_DIR=$DEST/data
41 # Tempest
42 enable_service tempest
```

L'installation peut être lancéee à l'aide du script dans le répertoire devstack :

```
1 ./stack.sh
```

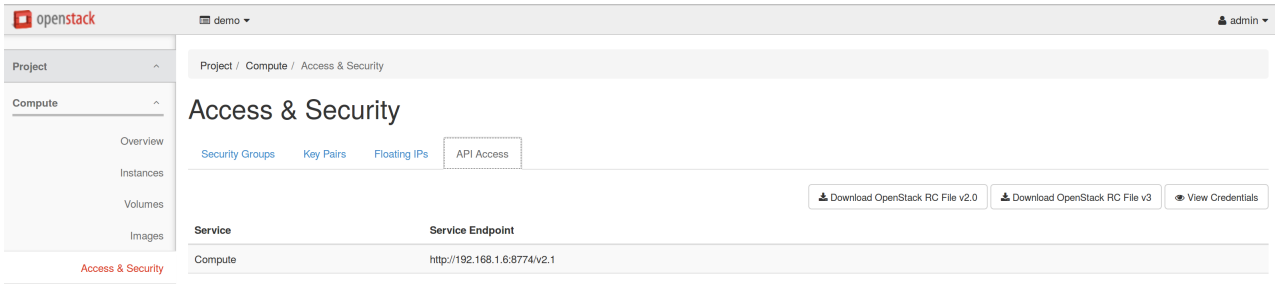
Une fois l'installation terminée, le script affiche l'URL à partir de laquelle nous pouvons accèder à l'interface de gestion (dashboard), l'URL contiendra l'adresse IP de la VM, d'où l'importance de fixer l'adresse IP afin d'accéder à l'interface d'administration.

```
=========================
DevStack Component Timing
=========================
Total runtime          19954

run_process             108
test_with_retry           4
apt-get-update            9
pip_install            2644
restart_apache_server     9
wait_for_service         11
git_timed              11558
apt-get                3233
=========================



This is your host IP address: 192.168.1.6
This is your host IPv6 address: fe80::a00:27ff:fec0:d4ed
Horizon is now available at http://192.168.1.6/dashboard
Keystone is serving at http://192.168.1.6/identity/
The default users are: admin and demo
The password: stack
```

Important : Une fois que le script a été exécuté, la VM ne peut plus être éteinte car sinon la base de données d'Open Stack serait corrompue.

3.3.2 Configuration :

— Téléchargement des données d'authentification Open Stack : Connexion à l'interface web > Compute section > Access security > API Access > Téléchargement des fichiers OpenStack RC pour les projets *demo* et *admin*.



— Récupération des données du fichier Admin RC :

```
1 source admin-openrc.sh
```

— Téléchargement de l'image cloud Ubuntu 14.04 et ajout à OpenStack avec l'outil Glance :

```
1 sudo wget http://cloud-images.ubuntu.com/releases/14.04/
     release/ubuntu-14.04-server-cloudimg-amd64-disk1.img -P
      /var/image
2
3 glance image-create --name "Ubuntu-14.04" --disk-format
     qcow2 --container-format bare --visibility public --
     progress < /var/image/ubuntu-14.04-server-cloudimg-
     amd64-disk1.img --min-disk 4 --min-ram 512
```

— Récupération des données du fichier Demo RC :

```
1 source demo-openrc.sh
```

— Création de la clé à utiliser lors de l'ouverture de sessions SSH sur les instances :

```
1 nova keypair-add kp1 > key
2 chmod 700 key
```

— Récupération de l'identifiant du réseau privé :

```
1 neutron net-list
```

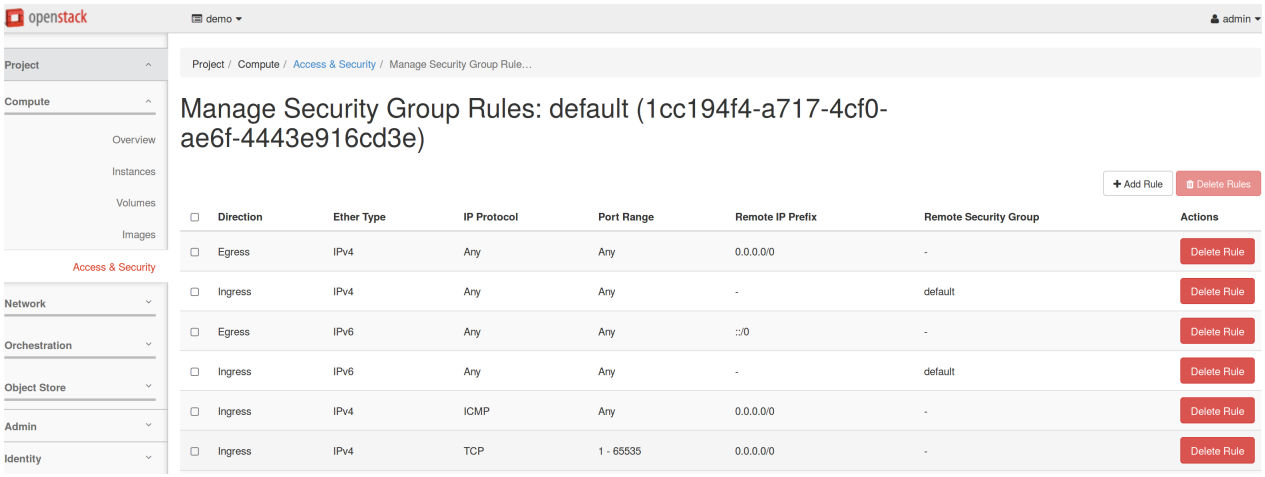
```
stack@stackServer:~/devstack$ neutron net-list
+--------------------------------------+---------+--------------------------------------+
| id                                   | name    | subnets                              |
+--------------------------------------+---------+--------------------------------------+
| 35a19621-81b3-4718-bd47-bae67f8      | private | f49c3a3b-fa8f-49f1-aacb-             |
| 0771d                                |         | 0a9375352bc0 10.0.0.0/24             |
|                                      |         | 03de292b-029c-                       |
|                                      |         | 4c47-b773-88eb28df7b4c               |
|                                      |         | fd71:c1:3086::/64                    |
| ac3296a7-0c77-4e85-8b26-40cccb3      | public  | 7879402c-6797-4bf4-8822-dd4560fa     |
| 84f30                                |         | da62 172.24.4.0/24                   |
|                                      |         | 185eb48b-1985-4230-aa30-6e01111f     |
|                                      |         | afd0 2001:db8::/64                   |
+--------------------------------------+---------+--------------------------------------+
```

Cet identifiant devra être utilisé dans la commande suivante.

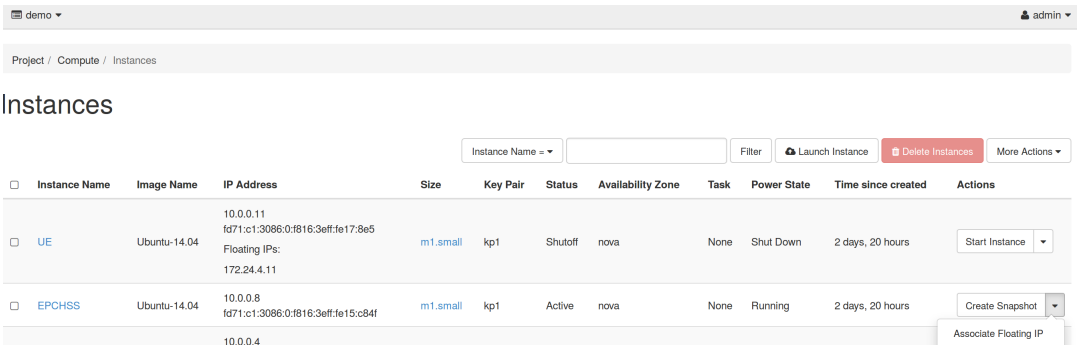
— Création des instances :

```
1 nova boot --flavor m1.small --image Ubuntu-14.04 --nic net
    -id=<Network ID from previous command> --security-group
     default --key-name kp1 <Instance Name>
```

— Ajout de règles aux groupes de sécurité : Ceci permettra d'autoriser le trafic SSH, HTTP, et ICMP vers les instances : Demo project > Compute section > Access security > Security Groups > Manage rules du group *default* > Add rules.



— Association d'adresses IP *flottantes* aux instances pour communiquer avec les réseaux externes : Compute node > Instances > Associate a floating IP pour chaque instance.



Une fois les instances créées, elles peuvent être jointes via leurs adresses flottantes avec SSH en utilisant la clé créée précédemment :

```
1 ssh -i key ubuntu@<Floating IP>
```

3.4 Installation des composants :

— Ajout du FQDN de l'hôte :
Ajout dans le fichier */etc/hosts* de la ligne :

```
1 <Adresse_IP_privee> <nom_d'hote>.openair4G.eur <hostname>
```

— Installation d'un noyeau *low latency* :

```
1 sudo apt-get install linux-lowlatency-lts-vivid
```

Une fois le noyeau installé, l'instance doit être redémarrée.

— Installation de Git et téléchargement d'OpenAir-cn, mais tout d'abord le certificat d'EURECOM doit être ajouté à ceux du système (La commande doit être exécutée en tant que *root*) :

```
1 echo -n | openssl s_client -showcerts -connect gitlab.
    eurecom.fr:443 2>/dev/null | sed -ne '/-BEGIN
    CERTIFICATE-/,/-END CERTIFICATE-/p' >> /etc/ssl/certs/
    ca-certificates.crt
2
3 git clone https://gitlab.eurecom.fr/oai/openair-cn.git
```

Récupération de la version v0.3.2 :

```
1 cd openair-cn
2 git checkout v0.3.2
```

Cette version permet d'installer et de lancer le HSS avec les scripts *build_hss* et *run_hss*, et le MME ainsi que le S-GW et P-GW avec les scripts *build_epc* et *run_epc*.

— Installation des logiciels supplémentaires requis :

```
cd SCRIPTS
./build_hss -i
./build_epc -i
```

— Compilation :

```
./build_hss -c -l
./build_epc -c -l -vV
```

— Modification des fichiers de configuration :

  — */usr/local/etc/oai/hss.conf* : Modification du login et mot de passe SQL.
  — */usr/local/etc/oai/epc.conf* : Modification des adresses IP des différentes interfaces selon la configuration résseau.

— Création de la base de données HSS :

```
./create_hss_database <dbuser> <dbpassword> <hssuser> <
   hsspassword> oai_db
```

— Vérification des certificats MME et HSS :

```
./check_hss_s6a_certificate /usr <hostname>.openair4G.eur

./check_mme_s6a_certificate /usr <hostname>.openair4G.eur
```

— Ajout du FQDN du MME dans la base de données HSS : Dans Firefox, accéder à l'interface phpmyadmin de l'instance (@IP/phpmyadmin), aller à la base de données *oai_db*, puis la table *mmeidentity* , puis modifier le FQDN d'une des lignes en le remplaçant par celui de l'instance.

3.5 Mise en oeuvre :

De préférence sur deux terminaux différents afin d'effectuer un meilleur suivi :

```
./run_hss
./run_epc
```

3.5.1 Lancement du HSS :

Une fois l'interface S6-A initialisée, le HSS tente de se connecter au MME périodiquement :

```
1  07/23/16,09:16:44.808566 DBG Core state: 1 -> 2
2  07/23/16,09:16:44.839922 NOTI Local server address(es):
     10.0.0.8{---L-}
3  07/23/16,09:16:44.843285 DBG epchss.openair4G.eur: Connecting
     ...
4  07/23/16,09:16:44.843638 DBG 'STATE_CLOSED' -> '
     STATE_WAITCNXACK' 'epchss.openair4G.eur'
5  07/23/16,09:16:44.850273 DBG Core state: 2 -> 3
6  Initializing s6a layer: DONE
7  07/23/16,09:16:44.855925 DBG Prepared 1 sets of connection
     parameters to peer epchss.openair4G.eur
8  07/23/16,09:16:44.857335 DBG Connecting to TCP 10.0.0.8(3870)
     ...
9  07/23/16,09:16:44.863095 DBG TCP connection to 10.0.0.8(3870)
     failed: Connection refused
10 07/23/16,09:16:44.863879 DBG Connection to 'epchss.openair4G.
     eur' failed: All connection attempts failed, will retry
     later
```

3.5.2 Lancement du MME + P-GW + S-GW :

Après une période d'initialisation, nous pouvons constater que la connexion avec le HSS a été établie :

```
1     Initializing SPGW-APP task interface: DONE
2     ================== Statistics ==================
3  | Global | Since last display |
4     UE | 0 | 0 |
5  Bearers | 0 | 0 |
6  ================== Statistics ==================
7           | Global | Since last display |
8     UE | 0 | 0 |
9  Bearers | 0 | 0 |
10 'STATE_OPEN' <-- 'FDEVP_CNX_MSG_RECV' (0x7f5b70001960,84) '
     epchss.openair4G.eur'
11    RCV from 'epchss.openair4G.eur': (no model)0/280 f:R--- src
        :'epchss.openair4G.eur' len:84 {C:264/1:28,C:296/1:21,C
        :278/1:12}
12 Iterating on rules of COMMAND: 'Device-Watchdog-Request'.
13    No Session-Id AVP found in message 0x1641ba0
```

```
14    SENT to 'epchss.openair4G.eur': 'Device-Watchdog-Answer
        '0/280 f:---- src:'(nil)' len:96 {C:268/l:12,C:264/l:28,C
        :296/l:21,C:278/l:12}
15    Sending 96b data on connection {----} TCP from
        [10.0.0.8]:53990 (32<-33)
16  Peer timeout reset to 30 seconds (+/- 2)
17  'epchss.openair4G.eur' in state 'STATE_OPEN' waiting for next
      event.
18    ================== Statistics ==================
19  | Global | Since last display |
20    UE | 0 | 0 |
21    Bearers | 0 | 0 |
22    ================== Statistics ==================
23          | Global | Since last display |
24    UE | 0 | 0 |
25  Bearers | 0 | 0 |
26  ================== Statistics ==================
27  | Global | Since last display |
28    UE | 0 | 0 |
29    Bearers | 0 | 0 |
30    'STATE_OPEN' <-- 'FDEVP_PSM_TIMEOUT' ((nil),0) 'epchss.
        openair4G.eur'
31  Peer timeout reset to 30 seconds
32    'epchss.openair4G.eur' in state 'STATE_OPEN' waiting for
        next event.
33  SENT to 'epchss.openair4G.eur': 'Device-Watchdog-Request'0/280
      f:R--- src:'(nil)' len:84 {C:264/l:28,C:296/l:21,C:278/l
      :12}
34    Sending 84b data on connection {----} TCP from
        [10.0.0.8]:53990 (32<-33)
35  'STATE_OPEN' <-- 'FDEVP_CNX_MSG_RECV' (0x7f5b70000b40,96) '
      epchss.openair4G.eur'
36    RCV from 'epchss.openair4G.eur': (no model)0/280 f:---- src
        :'epchss.openair4G.eur' len:96 {C:268/l:12,C:264/l:28,C
        :296/l:21,C:278/l:12}
37    Iterating on rules of COMMAND: 'Device-Watchdog-Answer'.
38  Peer timeout reset to 30 seconds (+/- 2)
39  'epchss.openair4G.eur' in state 'STATE_OPEN' waiting for next
      event.
```

Le HSS affiche également une notification de connexion :

```
'STATE_CLOSED' -> 'STATE_WAITCNXACK' 'epchss.openair4G.eur'
Prepared 1 sets of connection parameters to peer epchss.openair4G.eur
Connecting to TCP 10.0.0.8(3870)...
epchss.openair4G.eur: Connection established, {----} TCP,#9->10.0.0.8(3870)
SENT to 'epchss.openair4G.eur': 'Capabilities-Exchange-Request'0/257 f:R--- src:'(nil)' len:200 {C:264/l:28,C:296/l:21,C:278/l:12,C:257/l:14,C:266/l:12,C:269/l:20,C:267/l:12,C:299/l:12,C:260/l:32,C:265/l:12}
'STATE_WAITCNXACK' -> 'STATE_WAITCEA' 'epchss.openair4G.eur'
RCV from 'epchss.openair4G.eur': (no model)0/257 f:---- src:'epchss.openair4G.eur' len:200 {C:268/l:12,C:264/l:28,C:296/l:21,C:278/l:12,C:257/l:14,C:266/l:12,C:269/l:20,C:267/l:12,C:260/l:32,C:265/l:12}
Connected to 'epchss.openair4G.eur' (TCP,soc#9), remote capabilities:
  'Capabilities-Exchange-Answer'
   Version: 0x01
   Length: 200
   Flags: 0x00 (----)
   Command Code: 257
   ApplicationId: 0
   Hop-by-Hop Identifier: 0x671B13E1
   End-to-End Identifier: 0x5F2C6373
    {internal data}: src:epchss.openair4G.eur(20) rwb:(nil) rt:2 cb:(nil),(nil)((nil)) qry:0x7f9e28000a30 asso:0 sess:(nil)
    AVP: 'Result-Code'(268) l=12 f=-M val='DIAMETER_SUCCESS' (2001 (0x7d1))
    AVP: 'Origin-Host'(264) l=28 f=-M val="epchss.openair4G.eur"
    AVP: 'Origin-Realm'(296) l=21 f=-M val="openair4G.eur"
    AVP: 'Origin-State-Id'(278) l=12 f=-M val=1469265446 (0x57933626)
    AVP: 'Host-IP-Address'(257) l=14 f=-M val=10.0.0.8
    AVP: 'Vendor-Id'(266) l=12 f=-M val=0 (0x0)
    AVP: 'Product-Name'(269) l=20 f=-- val="freeDiameter"
    AVP: 'Firmware-Revision'(267) l=12 f=-- val=10200 (0x27d8)
    AVP: 'Vendor-Specific-Application-Id'(260) l=32 f=-M val=(grouped)
      AVP: 'Auth-Application-Id'(258) l=12 f=-M val=16777251 (0x1000023)
```

```
28          AVP: 'Vendor-Id'(266) l=12 f=-M val=10415 (0x28af)
29        AVP: 'Supported-Vendor-Id'(265) l=12 f=-M val=10415 (0
             x28af)
30    No TLS protection negotiated with peer 'epchss.openair4G.eur
      '.
31    'STATE_WAITCEA' -> 'STATE_OPEN' 'epchss.openair4G.eur'
32    SENT to 'epchss.openair4G.eur': 'Device-Watchdog-Request
      '0/280 f:R--- src:'(nil)' len:84 {C:264/l:28,C:296/l:21,C
      :278/l:12}
33    RCV from 'epchss.openair4G.eur': (no model)0/280 f:---- src
      :'epchss.openair4G.eur' len:96 {C:268/l:12,C:264/l:28,C
      :296/l:21,C:278/l:12}
34    RCV from 'epchss.openair4G.eur': (no model)0/280 f:R--- src
      :'epchss.openair4G.eur' len:84 {C:264/l:28,C:296/l:21,C
      :278/l:12}
35    SENT to 'epchss.openair4G.eur': 'Device-Watchdog-Answer
      '0/280 f:---- src:'(nil)' len:96 {C:268/l:12,C:264/l:28,C
      :296/l:21,C:278/l:12}
```

Les composants continuent de communiquer en attendant la connexion d'eNodeB et/ou de périphérique utilisateur.

Chapitre 4

Conclusion

Au cours de ce stage, nous-nous sommes familiarisés avec la technologie 4G LTE, son fonctionnement, son architecture, ainsi que les différents outils logiciels utilisés pour le déploiement des fonctions de l'EPC sur cloud (Open Stack, Open Air).

Nous avons pu constater que la virtualisation des fonctions de l'EPC était possible et nécessitait relativement peu de ressources, sans avoir recours à des machines dédiées ou de logiciels propriétaires.

La NFV est donc une technologie qui permet aux fournisseurs de service d'optimiser les ressources, et de disposer d'une grande flexibilité de déploiement et de gestion.

Bibliographie

[1] François-Xavier Wolf Yannick Bouguen, Éric Hardouin. *LTE et les réseaux 4G*. Eyrolles, 2012. ISBN : 978-2-212-12990-8.

[2] http://www.tutorialspoint.com/lte/lte_network_architecture.htm.

[3] http://searchsdn.techtarget.com/definition/network-functions-virtualization-NFV.

[4] Germine Seide. *Planification d'un réseau de quatrième génération à partir d'un réseau de troisième génération*. PhD thesis, Université de Monréal, 2011.

[5] Aman Jain. Building and benchmarking mme and hss in sdn based lte-epc. Master's thesis, Indian Institute of Technology Bombay, 2015.

[6] ETSI. 3GPP TS 23.0.002 version 13.5.0 Release 13. Standard, 3GPP, April 2016.

[7] https://gitlab.eurecom.fr/oai/openair-cn.