Simulation d'un réseau 5G Core et optimisation du placement et de la sélection dynamique des UPFs

Derroueche Samia[1], Rebhi Assala Nour Elimane[1], Herkat Wifak[1], Zoutat Marwa[1],Larbaoui Yasmine Badr Elhouda[1], Hamed Hiba[1], Boudiaf Fadia[1], Amrouche hakim[1] et Hamani nacer[1*]

[1]Ecole Nationale Supérieure de l'Informatique (ESI Alger), Algiers, Algeria

Abstract

Les applications critiques nécessitent des communications ultra-fiables à faible latence (uRLLC) imposent des défis majeurs en termes d'optimisation de l'infrastructure réseau. Cette étude présente une approche pour la simulation du réseau cœur 5G (5GC) et l'optimisation du placement et sélection des fonctions de plan utilisateur (UPF) en utilisant une plateforme de simulation basée sur Free5GC et UERANSIM dans un environnement conteneurisé. Nous avons développé et évalué des algorithmes d'optimisation du placement et de sélection dynamique des UPFs. L'expérimentation a été menée sur des topologies avec différents scénarios d'optimisation.

Les résultats démontrent que la combinaison de la sélection dynamique et du placement adaptatif des UPFs permet d'atteindre un équilibre optimal entre performances, utilisation des ressources et fiabilité du réseau, validant ainsi l'efficacité de l'approche proposée pour les applications uRLLC dans les réseaux 5G.

Mots-clés: 5G Core (5GC), Architecture orientée services (SBA), UPF (User Plane Function), Network Slicing, NFV (Network Functions Virtualization), SDN (Software-Defined Networking), Edge Computing, MEC (Multi-access Edge Computing), uRLLC (Ultra-Reliable Low Latency Communications), Réseaux 5G privés.

Introduction

L'émergence des réseaux 5G marque une véritable révolution technologique, transformant les paradigmes des communications mobiles et permettant l'émergence d'applications critiques nécessitant des performances extrêmes, telles que les communications ultra-fiables à faible latence (uRLLC). Ces applications, qui englobent des domaines sensibles comme l'automatisation industrielle, la robotique en temps réel, la santé intelligente ou encore les véhicules autonomes, imposent des exigences rigoureuses en termes de latence, de fiabilité et de bande passante. Dans ce contexte, les réseaux privés 5G se positionnent comme une solution adaptée aux environnements industriels et professionnels, en offrant un contrôle accru sur l'infrastructure et les performances du réseau. Au cœur de cette architecture, la fonction de plan utilisateur (User Plane Function - UPF) joue un rôle central. Elle est responsable du traitement et du routage des données utilisateur, de la gestion fine de la qualité de service (QoS), ainsi que de la réduction des délais de transmission. Son positionnement dans l'infrastructure réseau influe directement sur les performances perçues par les applications finales, en particulier celles nécessitant une latence minimale. Cependant, déterminer le placement optimal des UPFs ainsi que leur nombre constitue un défi majeur dans la conception et la gestion des réseaux 5G. Un placement inapproprié peut entraîner une surcharge des nœuds, une latence accrue, ou une inefficacité dans l'allocation des ressources réseau. À l'inverse, une stratégie de placement et de sélection bien pensée peut permettre d'atteindre un équilibre entre performances, coûts d'infrastructure, et résilience du réseau. Dans ce contexte, cet article s'intéresse à la simulation du réseau cœur 5G en mettant l'accent sur l'optimisation de le placement et de la sélection des UPFs. L'objectif est d'évaluer, à travers une approche expérimentale et analytique, l'impact de différentes stratégies de déploiement sur les performances globales du réseau, et d'identifier les conditions permettant de satisfaire les exigences strictes des services uRLLC tout en maintenant une utilisation efficace des ressources.

*Corresponding author: la_rebhi@esi.dz
Received: June 15, 2025, **Published:** Not Yet

Résumé de l'étude théorique

Cette étude théorique approfondie examine l'architecture révolutionnaire et les innovations technologiques des réseaux 5G, avec un accent particulier sur le cœur de réseau 5G (5GC) et ses mécanismes fondamentaux. La cinquième génération de réseaux mobiles représente une avancée majeure par rapport à la 4G/LTE, offrant des débits pouvant atteindre 20 Gbps, une latence réduite à moins d'une milliseconde, et une capacité de connectivité massive

permettant jusqu'à un million de dispositifs par kilomètre carré. Ces performances exceptionnelles ouvrent la voie à de nouveaux cas d'usage critiques tels que les véhicules autonomes, la chirurgie à distance, et les villes intelligentes.
L'architecture 5G repose sur trois piliers principaux : la couche d'accès radio (NG-RAN) avec ses stations de base gNodeB exploitant le Massive MIMO et le beamforming, le réseau de transport (fronthaul, midhaul, backhaul), et le cœur de réseau 5GC. Ce dernier adopte une architecture orientée services (SBA) basée sur des microservices, remplaçant le modèle monolithique de la 4G. Les fonctions réseau clés incluent l'AMF pour la gestion de la mobilité, le SMF pour le contrôle des sessions, et surtout l'UPF (User Plane Function) pour le traitement des données utilisateur. L'UPF, conçu selon le principe CUPS (Control and User Plane Separation), peut être déployé de manière distribuée avec différentes instances (PSA, I-UPF, UL-CL) pour optimiser les performances.
La virtualisation des fonctions réseau (NFV) et les réseaux définis par logiciel (SDN) constituent des technologies clés permettant une gestion dynamique des ressources. Le placement optimal des UPF, particulièrement crucial pour les services URLLC (Ultra-Reliable Low Latency Communications), peut être optimisé grâce à des approches mathématiques (modélisation MILP,NOUP ...) ou des techniques d'intelligence artificielle (apprentissage par renforcement). Le Network Slicing permet quant à lui de partitionner le réseau physique en plusieurs réseaux logiques indépendants (slices), chacun étant optimisé pour des besoins spécifiques (eMBB, uRLLC, mMTC).
L'intégration de l'Edge Computing avec la 5G, via le Multi-access Edge Computing (MEC), permet de déployer des UPF en périphérie du réseau, réduisant ainsi la latence et améliorant l'efficacité. Les réseaux 5G privés, conçus pour des environnements industriels ou institutionnels, offrent des avantages en termes de sécurité, de qualité de service garantie, et d'indépendance opérationnelle. Ces réseaux exploitent pleinement les potentialités du slicing réseau et de l'edge computing pour répondre aux besoins critiques des entreprises.

Choix de Simulateur

La simulation est un outil fondamental pour étudier et optimiser le placement des fonctions UPF dans un réseau 5G. Pour notre projet, nous avons opté pour une solution open-source combinant Free5GC [1], une plateforme de cœur de réseau 5G conforme aux standards 3GPP Release 15+, et UERANSIM [2], un émulateur du réseau radio 5G (RAN). Cette association assure une modélisation complète et réaliste du réseau 5G, du cœur jusqu'à la couche radio [3].

Architecture du Plan de Contrôle et du Plan Utilisateur via Free5GC

Le plan de contrôle et le plan utilisateur ont été implémentés à l'aide du Free5GC, une plateforme *cloud-native* conforme aux spécifications 3GPP Release 16. Cette architecture permet une dissociation fine des fonctions réseau et un déploiement granulaire à travers des microservices conteneurisés.

Les interfaces standardisées (`N1`, `N2`, `N3`, `N4`, `N6`, `N9`) sont pleinement supportées, assurant une interopérabilité avec des simulateurs d'accès radio. En particulier, la modularité de Free5GC autorise l'intégration de multiples instances UPF, condition préalable à l'expérimentation du placement optimisé des fonctions de transport selon les contraintes de latence et de charge [4].

Émulation de l'Accès Radio et des Terminaux Utilisateurs avec UERANSIM

La couche d'accès radio (`gNB`) et les terminaux utilisateurs (`UE`) sont émulés via UERANSIM, un simulateur protocolaire capable d'établir des sessions PDU avec le cœur Free5GC en mode `SA`. UERANSIM supporte les protocoles NAS, NGAP et GTP-U, et interagit avec les interfaces `N1`, `N2`, et `N3`, assurant ainsi la complétude du plan de signalisation et de transport.

L'intérêt de cette solution réside dans sa légèreté d'intégration et son réalisme fonctionnel pour des scénarios de simulation logicielle du *Core*. UERANSIM ne simule pas les couches physiques ni `MAC/PHY`, ce qui en fait un outil approprié pour l'étude exclusive du comportement logique du réseau et des fonctions du `5GC`.

Son interopérabilité avec Free5GC est confirmée dans plusieurs travaux de bancs de test `5G SA`, notamment pour l'évaluation de la *QoS*, du *slicing*, ou de la gestion des sessions multiples dans des environnements conteneurisés.

Justification du choix

Le couplage **Free5GC + UERANSIM** offre un cadre complet de simulation du réseau 5G, garantissant à la fois la conformité aux standards `3GPP` et la flexibilité d'une plateforme open-source. Leur intégration via des interfaces normalisées facilite la modélisation des interactions entre le cœur réseau et la couche radio, cruciale pour évaluer l'impact des algorithmes d'optimisation de placement UPF.

Leur compatibilité avec Docker permet un déploiement et une orchestration modulaires, adaptés aux scénarios dynamiques du réseau 5G et à la simulation de topologies complexes.

Méthode de Déploiement du Réseau 5G Core

Environnement de Simulation

La mise en œuvre d'un réseau 5G Core expérimental nécessite une approche qui concilie fidélité aux standards industriels et flexibilité de recherche. Notre déploiement s'appuie sur Free5GC, une implémentation open-source reconnue pour sa conformité stricte aux spécifications 3GPP Release 15 et sa popularité dans la communauté académique [5, 6].Cette plateforme facilite l'automation de l'orchestration logicielle et le développement de services, représentant la tendance la plus récente dans le développement des réseaux cœur 5G. L'approche cloud-native adoptée repose sur les principes de conteneurisation pour isoler les fonctions réseau, optimiser la consommation des ressources et simplifier les procédures de déploiement. Les conteneurs offrent de nombreux avantages à la virtualisation des fonctions réseau (NFV) et ont été largement adoptés par les équipes d'ingénierie réseau. Docker a été sélectionné pour sa maturité technologique et son écosystème développé, permettant d'encapsuler chaque fonction réseau dans un conteneur autonome incluant toutes les dépendances logicielles requises. Dans la conteneurisation, l'application réside dans un conteneur abstrait du système d'exploitation et inclut toutes les dépendances runtime nécessaires. Cette configuration est beaucoup plus efficace en ressources, car un conteneur partage le noyau du système d'exploitation entre plusieurs conteneurs. Cette approche garantit la portabilité du déploiement et la reproductibilité dans divers environnements matériels et logiciels [7]. Free5GC-Compose constitue une version Docker Compose de Free5GC pour la stage 3, inspirée par free5gc-docker-compose. Cette distribution maintenue par Orange Open Source fournit une implémentation conteneurisée optimisée avec des configurations pré-validées pour les environnements de test.

Implémentation des Fonctions Réseau

L'architecture mise en place repose sur le paradigme microservices, où chaque fonction réseau (Network Function - NF) se trouve encapsulée dans un conteneur qui est autonome [6, 7], c'est-à-dire que les ressources sont gérées de façon granulaire et que l'évolution de chaque composant est strictement ciblée. De fait, nous avons la possibilité d'effectuer la maintenance, d'appliquer des mises à jour sélectives ainsi que d'analyser les performances respectives de chaque composant.
Le cœur 5G mis en place regroupe les fonctions réseau nécessaires reprises dans les spécifications du 3GPP:

- **Access and Mobility Function (AMF)** : en charge de la gestion de l'accès des équipements utilisateurs ainsi que de leur authentification et mobilité.
- **Session Management Function (SMF)** : chargée du contrôle et de la gestion des sessions PDU, des politiques QoS, ainsi que du routage du trafic utilisateur.
- **User Plane Functions (UPFs)** : acheminement des données utilisateur avec déploiement de plusieurs instances :
  - **PSA-UPF (PDU Session Anchor)** pour l'ancrage des sessions,
  - **I-UPF (Intermediate UPF)** pour l'optimisation des chemins de données (ULCL, *Edge Computing*).
- **Authentication Server Function (AUSF)** : centralise les processus d'authentification et d'autorisation des équipements.
- **Unified Data Management (UDM)** : fournit une gestion des données d'abonnement et de profils utilisateurs unifiée.
- **Network Repository Function (NRF)** : permet la découverte et l'enregistrement des services réseau.
- **Network Slice Selection Function (NSSF)** : responsable de la sélection des tranches de réseau à utiliser selon les besoins applicatifs.
- **Policy Control Function (PCF)** : gère la mise en œuvre des politiques de contrôle de sessions et la gestion des ressources.
- **Charging Function (CHF)** : collecte et consolidation des informations de facturation en temps réel en interaction avec SMF, UPF et PCF.

La figure ci-dessous illustre l'architecture complète du banc d'expérimentation implémenté. Elle met en évidence les couches d'abstraction matérielles et logicielles, l'organisation des fonctions réseau virtualisées et simulée au sein de l'environnement virtualisé.

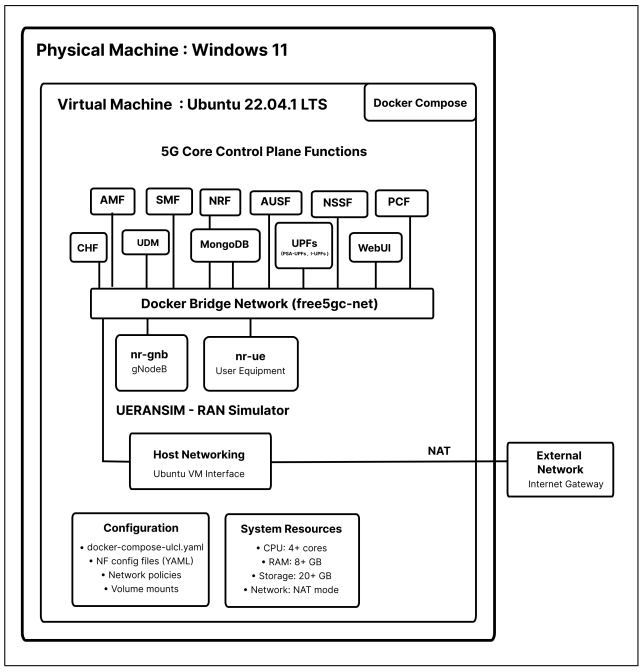

Figure 1: *Déploiement du cœur de réseau 5G Free5GC avec Docker Compose et simulation RAN via UERANSIM dans un environnement virtualisé.*

L'orchestration s'effectue via le fichier `docker-compose-ulcl.yaml` qui définit de manière déclarative l'intégralité des services. Les éléments suivants sont spécifiés pour chaque service déployé:

- l'image Docker et sa version
- les variables d'environnement propres à chaque fonction réseau
- les volumes persistants utilisés pour stocker les fichiers de configuration et autres données
- les dépendances inter-services à travers la directive `depends_on`, afin de garantir l'ordre de démarrage des services critiques
- le réseau virtuel Docker ainsi que l'adressage IP statique ou dynamique attribué à chaque conteneur.

Configuration et Connectivité

Afin de fournir au simulateur du cœur de réseau 5G un environnement virtualisé isolé et reproductible, l'environnement expérimental sera déployé sur une machine **Ubuntu 22.04 LTS** installée sous *VMware Workstation*.

Dans la mise en œuvre d'une architecture conteneurisée Free5GC, plusieurs autres logiciels doivent être installés au préalable, à savoir Docker Engine pour l'exécution des conteneurs, Docker Compose v2 pour l'orchestration multi-conteneurs, et enfin le module noyau `GTP5G`,indispensable au plan utilisateur (UPF).

Le code source de Free5GC-Compose est cloné depuis le dépôt officiel GitHub par la commande suivante:

```
git clone https://github.com/free5gc/
    free5gc-compose.git
```

Cette distribution inclut les Dockerfile pour chaque fonction réseau [5], les fichiers de configurations yaml, les scripts d'automatisation, ainsi que le fichier central `docker-compose-ulcl.yaml` qui orchestre tout le processus de déploiement.

Après une compilation locale des images de Docker, différents types de fonctions réseau sont alors prêts à être instanciés en vue de la conduite des scénarios de validation du cœur 5G.

Pour chaque fonction réseau, un fichier de configuration `YAML` spécifique (`amfcfg.yaml`, `smfcfg.yaml`, `upfcfg.yaml`, etc.) permet un réglage fin des paramètres des éléments suivants :

- **Interfaces réseau** : adresse IP, ports d'écoute, protocoles
- **Identifiants réseau** : PLMN (Public Land Mobile Network), TAI (Tracking Area Identity), DNN (Data Network Name)
- **Relations inter-NF** : points de service et dépendances fonctionnelles
- **Paramètres de sécurité** : clés d'authentification, algorithmes de chiffrement

L'ordre de dépendance retenu pour le démarrage est réputé respecté, en assurant que les services de base (NRF, UDM, AUSF) soient fonctionnels avant le lancement des entités de contrôle (AMF, SMF) et de plan utilisateur (UPF). Un service de sondes de santé permet le suivi permanent des services et le diagnostic des pannes

La connectivité entre conteneurs s'établit sur un réseau Docker bridge nommé `free5gc-net`, offrant un environnement réseau isolé et contrôlé. Cette configuration permet d'isoler le trafic expérimental du réseau hôte, de simuler les interfaces 5G normalisées et d'assurer une flexibilité dans l'attribution des adresses IP.

La stratégie d'adressage combine deux approches :

- La résolution DNS qui utilise les noms de services Docker Compose pour opérer entre conteneurs afin de simplifier la configuration et d'augmenter la maintenabilité.
- L'adressage IP statique en utilisant des adresses fixes pour les fonctions critiques qui requièrent une connectivité prévisible.

Les interfaces logiques 5G sont associées aux spécifications de la 3GPP :

- N1 : Interface NAS (Non-Access Stratum) entre UE et AMF
- N2 : Interface de contrôle entre gNB et AMF
- N3 : Interface de données entre gNB et UPF
- N4 : Interface de contrôle entre SMF et UPF
- N6 : Interface vers les réseaux de données externes

L'hôte met en place des règles iptables et des routes statiques dans le but de simuler les chemins de communications entre le RAN, l'UPF et les réseaux de données extérieures, afin de pallier l'absence d'une infrastructure radio physique.

Validation Fonctionnelle par Simulation End-to-End

La validation de l'intégrité du déploiement s'appuie sur UERANSIM, un simulateur open-source, régi par la licence Creative Commons, interfacé avec Free5GC [2], qui simule gNB et les équipements utilisateur. Le simulateur apparait comme une alternative économique à des équipements radio physiques tout en permettant le respect rigoureux des protocoles techniques 5G.

Le processus de validation des fonctionnalités suit un protocole rigoureux :

1. **Enregistrement initial** : validation de la procédure NAS entre UE et AMF.
2. **Établissement de session** : création d'une session PDU via le SMF et négociation QoS.
3. **Attribution d'adresse** : allocation d'adresse IP par l'UPF à partir du pool défini.
4. **Test de connectivité** : test `ping` entre le UE simulé et les services réseau.

Cette méthode de test permet donc de valider l'ensemble des flux de signalisation 5G SA (Standalone) et de s'assurer du bon orchestration des fonctions réseaux en environnement contrôlé [1, 2]. Les journaux des conteneurs sont surveillés attentivement pour confirmer le déroulement correct des procédures de signalisation et de transmission. Des outils de capture comme tcpdump et Wireshark analysent les flux GTP-U et GTP-C, permettant une inspection détaillée des paquets et protocoles. Le monitoring continu des services s'effectue via des sondes de santé intégrées, facilitant le diagnostic des pannes et l'optimisation des performances. Cette approche de validation complète garantit la fiabilité de l'infrastructure déployée et sa conformité aux standards 5G SA.

Automatisation de la Création et de la Connexion d'UEs et de gNBs

Dans le cadre de notre simulation du réseau 5G Core, nous avons mis en place une automatisation complète permettant de créer et de connecter plusieurs équipements utilisateurs (UEs) ainsi que plusieurs stations de base (gNBs), afin de contourner les limites de configuration manuelle imposées par défaut dans free5GC, où un seul UE et un seul gNB sont initialement prévus. Pour enrichir la topologie et mieux reproduire des scénarios réalistes à grande échelle, nous avons élaboré une série de scripts capables d'insérer automatiquement les UEs dans la base de données via l'API de la Web Console, tout en générant leurs fichiers de configuration YAML. Chaque UE reçoit un identifiant IMSI unique et est associé de façon dynamique à une adresse IP de gNB, permettant ainsi de répartir plus équitablement la charge sur le réseau. De la même manière, d'autres scripts assurent la génération et la configuration automatique des gNBs, l'attribution des adresses IP nécessaires dans le réseau local du conteneur, ainsi que leur démarrage en arrière-plan avec centralisation des journaux. Enfin, un script d'orchestration global permet de démarrer l'ensemble des UEs et des gNBs en une seule commande, simplifiant grandement la mise en place de scénarios complexes et facilitant l'analyse des performances réseau dans des conditions proches du réel.

Simulation des latences

Pour étudier l'impact du placement des UPFs sur les performances du réseau, nous avons implémenté une approche de simulation de latences à la fois locales (entre conteneurs) et distantes (entre machines).

Génération automatique des latences

Nous avons développé un script Python permettant de :

- Générer aléatoirement des **emplacements de composants** (UPF, AMF, SMF, etc.) dans une surface prédéfinis.
- Calculer la **latence aller-retour** entre chaque paire de composants à partir de leur distance géographique simulée, selon la formule physique suivante [8]:

$$\text{Latence (ms)} = \frac{2 \times \text{distance (km)}}{200\,000} \times 1000 \quad (1)$$

La vitesse de propagation dans la fibre optique est approximée à 200 000 km/s. Un minimum de 0.1 ms est imposé pour rester cohérent avec les exigences URLLC.

Le résultat est exporté dans un fichier `.csv` de la forme suivante :

```
type,source,destination,latency_ms
docker,smf,amf,0.3
host,smf,upf2,1.2
```

Listing 1: *Format du fichier CSV de latences*

Application dynamique des latences

Un second script lit le fichier `.csv` et applique les latences à l'aide de l'outil `tc` (traffic control) à l'intérieur des conteneurs Docker via l'outil `nsenter`. Cela permet d'introduire artificiellement une latence entre deux composants réseau, sans avoir besoin d'un environnement physique distant.

```
- tc qdisc add dev eth0 root
handle 1: prio
- tc filter add dev eth0 protocol
ip parent
1:0 prio 1 u32 match ip dst <IP_DEST>
- tc qdisc add dev eth0 parent
1:1 handle
10: netem delay <latency>ms
```

Listing 2: *Commandes TC pour l'application de latence*

Le script identifie automatiquement :

- l'**IP cible** via `docker inspect`
- le **PID du conteneur source** pour entrer dans son espace réseau
- applique la latence souhaitée

Simulation multi-machines

Pour tester la latence dans un réseau distribué (ex. : UPF sur une machine distante), la latence est également appliquée dans le **mode "host"**, c'est-à-dire entre deux conteneurs Docker déployés sur **des hôtes physiques différents** (connectés en mode `bridged` via VMware) [9].

Chaînage des UPF et Routage Dynamique dans Free5GC

Dans sa version conteneurisée, **Free5GC** permet le chaînage de plusieurs *UPF*. Parmi les atouts de ce simulateur, on trouve un exemple prédéfini de deux UPF enchaînés, servant de base pour étendre la topologie avec de nombreux UPF supplémentaires.

Les fonctions *User Plane Functions (UPF)* dans cette architecture jouent un rôle central dans le traitement efficace du trafic utilisateur. Elles sont réparties selon trois rôles distincts :

- **UPF d'ancrage** : toujours connecté à un *Data Network (N6)*, ce nœud est le point d'entrée et de sortie vers le réseau de données. C'est aussi lui qui attribue une adresse IP à l'UE.
- **UPF intermédiaire (I-UPF)** : il assure un routage dynamique via l'interface *N9*, en intégrant des fonctions de *traffic steering*, c'est-à-dire le choix du chemin vers d'autres UPF.
- **UPF de terminaison** : utilisé lorsque le trafic se termine localement, par exemple vers un serveur MEC, au lieu d'un accès Internet classique.

Configuration et déploiement : Les UPF sont déployés sous forme de conteneurs *Docker*, chacun avec son propre fichier de configuration où sont spécifiés les paramètres réseau, les interfaces et les adresses IP. Le rôle joué par chaque UPF est défini dans le fichier de configuration de la fonction *Session Management Function (SMF)*.

Un autre composant essentiel est le fichier `uerouting.yaml`, qui joue un rôle clé dans la définition de la topologie. Ce fichier permet de :

- Définir des règles de routage spécifiques pour chaque UE (User Equipment) ou groupe d'UEs.
- Activer la fonctionnalité *ULCL (Uplink Classifier)* pour bifurquer dynamiquement les flux vers des UPF spécifiques.

Le fichier `uerouting.yaml` est organisé en deux sections principales :

- `ueRoutingInfo` : liste des UE identifiés par leur IMSI et les chemins associés à chacun.
- `topology` : décrit les connexions logiques entre les nœuds (gNB, UPF, etc.).

Ainsi, la topologie peut être définie au niveau granulaire par UE ou groupe d'UEs. Après lancement de la topologie, il a été constaté que le chemin *GTP-U (User Plane)* s'établit correctement le long de la chaîne d'UPF sélectionnée, assurant un acheminement efficace du trafic entre l'UE et le réseau de données.

```yaml
free5gc-i-upf:
  container_name: i-upf
  image: free5gc/upf:v3.4.2
  command: bash -c "./upf-iptables.sh && ./upf -c ./config/upfcfg.yaml"
  volumes:
    - ./config/ULCL/upfcfg-i-upf.yaml:/free5gc/config/upfcfg.yaml
    - ./config/upf-iptables.sh:/free5gc/upf-iptables.sh
  cap_add:
    - NET_ADMIN
  networks:
    privnet:
      aliases:
        - i-upf.free5gc.org

free5gc-psa-upf:
  container_name: psa-upf
  image: free5gc/upf:v3.4.2
  command: bash -c "./upf-iptables.sh && ./upf -c ./config/upfcfg.yaml"
  volumes:
    - ./config/ULCL/upfcfg-psa-upf.yaml:/free5gc/config/upfcfg.yaml
    - ./config/upf-iptables2.sh:/free5gc/upf-iptables.sh
  cap_add:
    - NET_ADMIN
  networks:
    privnet:
      aliases:
        - psa-upf.free5gc.org
```

Figure 2: *Configuration Docker des conteneurs I-UPF et PSA-UPF dans Free5GC, illustrant leur déploiement modulaire avec des fichiers de configuration dédiés*

```yaml
gtpu:
  forwarder: gtp5g
  # The IP list of the N3/N9 interfaces on this UPF
  # If there are multiple connection, set addr to 0.0.0.0 or list all the addresses
  ifList:
    - addr: i-upf.free5gc.org
      type: N3
    - addr: i-upf.free5gc.org
      type: N9
    - addr: i-upf.free5gc.org
      type: N9
    - addr: i-upf.free5gc.org
      type: N9
```

Figure 3: *Configuration des interfaces N3 (accès radio) et N9 (inter-UPF) pour un I-UPF dans le fichier de configuration de I-UPF, définissant les points de terminaison GTP-U*

```yaml
gtpu:
  forwarder: gtp5g
  # The IP list of the N3/N9 interfaces on this UPF
  # If there are multiple connection, set addr to 0.0.0.0 or list all the addresses
  ifList:
    - addr: psa-upf.free5gc.org
      type: N9

# The DNN list supported by UPF
dnnList:
  - dnn: urllc # Data Network Name
    cidr: 10.60.0.0/16 # Classless Inter-Domain Routing for assigned IPv4 pool of UE
```

Figure 4: *Configuration d'un PSA-UPF avec son interface N9 et le pool d'adresses IP (10.60.0.0/16) pour le réseau de données URLLC*

```yaml
userplaneInformation:
  upNodes:
    gNB1:
      type: AN
      nodeID: gnb.free5gc.org

    I-UPF:
      type: UPF
      nodeID: i-upf.free5gc.org
      sNssaiUpfInfos:
        - sNssai:
            sst: 1
            sd: 010203
          dnnUpfInfoList:
            - dnn: urllc
      interfaces:
        - interfaceType: N3
          endpoints:
            - i-upf.free5gc.org
          networkInstances:
            - urllc
        - interfaceType: N9
          endpoints:
            - i-upf.free5gc.org
          networkInstances:
            - urllc
```

Figure 5: *Déclaration d'un I-UPF dans la configuration SMF avec ses interfaces N3/N9 pour le slice URLLC.*

```yaml
PSA-UPF:
  type: UPF
  nodeID: psa-upf.free5gc.org
  sNssaiUpfInfos:
    - sNssai:
        sst: 1
        sd: 010203
      dnnUpfInfoList:
        - dnn: urllc
          pools:
            - cidr: 10.60.0.0/16
  interfaces:
    - interfaceType: N9
      endpoints:
        - psa-upf.free5gc.org
      networkInstances:
        - urllc
```

Figure 6: *Déclaration du PSA-UPF dans la configuration SMF, incluant son pool DNN (10.60.0.0/16) et interface N9 pour le réseau urllc*

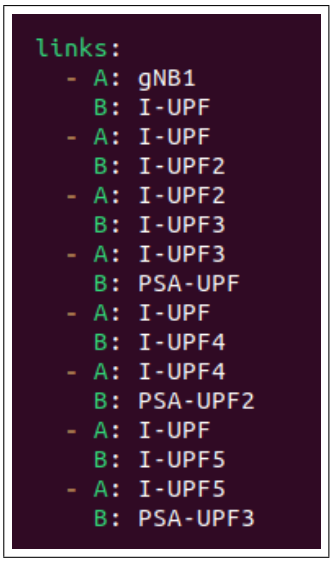

Figure 7: *Schéma des interconnexions entre gNB, I-UPFs et PSA-UPFs -déclaré dans la configuration SMF- pour le routage dynamique du trafic*

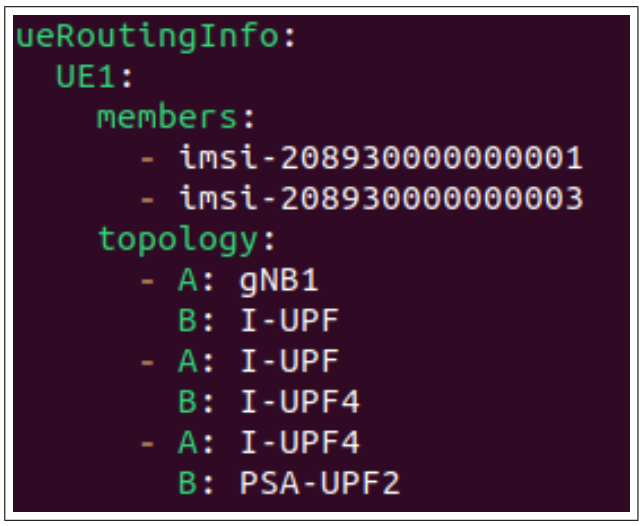

Figure 8: *Schéma des interconnexions entre gNB, I-UPFs et PSA-UPFs pour le routage dynamique du trafic*

Optimisation de placement des UPFs

Algorithme ILP

Le problème de placement optimal des UPFs (User Plane Functions) dans une architecture 5G est formulé ici comme un problème d'optimisation linéaire en nombres entiers (ILP). L'objectif est de minimiser le nombre total de UPFs déployés dans le réseau tout en garantissant une qualité de service, notamment en termes de latence et de capacité.

Algorithm 1 Placement optimal des UPFs avec ILP

Entrées :
Nr : ensemble des nœuds d'accès
Nc : ensemble des emplacements candidats
dr_r : demande de trafic pour chaque $r \in Nr$
$Lrc_{r,c}$: latence entre le nœud r et le candidat c
L_{req} : latence maximale tolérée
Cu : capacité maximale d'un UPF
α : seuil de capacité
Variables de décision :
$x_c \in \{0,1\}$: 1 si un UPF est déployé à c, 0 sinon
$p_{rc} \in \{0,1\}$: 1 si r est affecté à c, 0 sinon

Objectif : Minimiser $\sum_{c \in Nc} x_c$
Contraintes :
for all $r \in Nr$ **do**
$\sum_{c \in Nc} p_{rc} = 1$ ▷ Affectation unique
end for
for all $c \in Nc$ **do**
$\sum_{r \in Nr} dr_r \cdot p_{rc} \leq \alpha \cdot Cu$ ▷ Capacité
end for
for all $r \in Nr$, $c \in Nc$ **do**
$p_{rc} \leq x_c$ ▷ Affectation seulement si UPF actif
$Lrc_{r,c} \cdot p_{rc} \leq L_{req}$ ▷ Latence
end for
Étapes de Résolution :
1. Générer toutes les variables binaires x_c et p_{rc}.
2. Appliquer les contraintes de latence, de capacité et d'affectation unique.
3. Résoudre le programme linéaire en nombres entiers avec un solveur ILP (e.g. CPLEX, Gurobi, PuLP).
4. Obtenir la liste des emplacements activés $x_c = 1$ et les affectations $p_{rc} = 1$.
5. Retourner la solution optimale de placement minimisant le nombre de UPFs tout en respectant les contraintes.

Algorithme RealisticNOUP

C'est une extension de l'algorithme NOUP (Network Optimization for UPF Placement) spécifiquement conçue pour optimiser à la fois le placement et le chaînage des UPFs (User Plane Functions).

Architecture et Modèle du Système Dans l'architecture 5G, les UPFs peuvent être classés en trois types distincts selon leur rôle dans le traitement du trafic :

- **UL-CL (Uplink Classifier)** : premier UPF de la chaîne, responsable de la classification du trafic montant.
- **I-UPF (UPF Intermédiaire)** : relais pour les chaînes longues.
- **PSA-UPF (PDU Session Anchor)** : UPF terminal,

assurant l'ancrage des sessions.

Notre modèle considère :

- un ensemble de **gNBs** (antennes 5G) avec des demandes de trafic variables ;
- un ensemble de sites candidats pour les UPFs de différents types ;
- les composants du cœur de réseau (*e.g.*, SMF, AMF, etc.).

Formulation du Problème Il s'agit de déterminer, pour chaque gNB, une chaîne d'UPFs optimale respectant les contraintes suivantes :

- chaque chaîne commence par un UL-CL et se termine par un PSA-UPF ;
- la longueur de la chaîne est comprise entre `min_chain_length` et `max_chain_length` ;
- la latence totale ne dépasse pas `max_latency` ;
- la charge de chaque UPF ne dépasse pas `max_upf_capacity`.

L'algorithme **RealisticNOUP** étend l'approche NOUP en introduisant des considérations réalistes liées à la latence et au chaînage des UPFs.

Entrées :

- `gNBs`, `upf_candidates`, `core_components`
- `max_latency`, `max_upf_capacity`
- `min_chain_length`, `max_chain_length`

Sorties :

- `upf_chains`, `selected_upfs`, `upf_load`

Algorithm 2 Algorithme RealisticNOUP

```
INITIALISER distance_matrix, latency_matrix
for chaque gnb dans gNBs do
    chains ← FIND_OPTIMAL_CHAIN(gnb)
    if chains ≠ null then
        upf_chains[gnb] ← chains
        Mettre à jour selected_upfs, upf_load
    end if
end for
OPTIMIZE_CHAINS()
return upf_chains, selected_upfs, upf_load
```

Algorithm 3 FIND_OPTIMAL_CHAIN(gnb)

```
1: ul_cl_upfs ← FIND_UPFS_BY_TYPE(UL_CL)
2: i_upfs ← FIND_UPFS_BY_TYPE(I_UPF)
3: psa_upfs ← FIND_UPFS_BY_TYPE(PSA_UPF)
4: best_chain ← null, best_latency ← ∞
5: for chaque ul_cl dans ul_cl_upfs do
6:     for chaque psa_upf dans psa_upfs do
7:         for num_i_upfs de min_required à max_allowed do
8:             Générer une chaîne candidate
9:             latency ← CALCULATE_CHAIN_LATENCY(chain)
10:            if latency ≤ max_latency et latency < best_latency then
11:                best_chain ← chain
12:                best_latency ← latency
13:            end if
14:        end for
15:    end for
16: end for
17: return best_chain
```

Modèle de Latence Réaliste Le calcul de latence totale prend en compte :

- la distance physique entre les nœuds ;
- le ratio fibre/air dans les connexions ;
- la vitesse de propagation :
  - fibre : 200 000 km/s (corrigée par facteur 1.2) ;
  - air : 300 000 km/s ;
- le délai de traitement selon le type de nœud ;
- une pénalité RTT réaliste.

La latence entre deux nœuds est donnée par :

$$\text{Latency} = \text{PropagationDelay} + \text{ProcessingDelay} \times \text{RTTPenalty} \quad (2)$$

où le délai de propagation dépend de la distance et du type de lien (gNB-UPF ou UPF-UPF).

Sélection et Optimisation des Chaînes Pour chaque gNB, l'algorithme explore toutes les combinaisons valides de chaînes et sélectionne celle avec la latence minimale. Cette approche permet :

- une couverture maximale des gNBs ;
- une utilisation efficace des ressources UPF ;
- une minimisation de la latence bout-en-bout.

Notre implémentation tient aussi compte des zones de déploiement (urbaine, périurbaine, rurale) pour une évaluation fine

Optimisation de séléction des UPFs

Dans l'architecture 5G, une mauvaise répartition des UEs sur des UPFs peut causer une congestion, une latence élevée ou un déséquilibre de charge, notamment dans des scénarios multi-sites (local, edge, cloud).
Or, l'architecture 5G ne spécifie pas une méthode unique d'allocation dynamique des UEs à des UPFs. d'ou l'objectif de l'algorithme suivant est de sélectionner dynamiquement, pour chaque UE, le chemin optimal menant à un UPF, en tenant compte :la latence, la capacité réseau, la distance logique et de la charge actuelle du réseau.

Initialisation et Construction du Score de Coût

INITIALISER
$U \leftarrow \{u_1, u_2, \ldots, u_N\}$ ▷ Ensemble des UE à connecter
$\mathcal{P} \leftarrow \{P_1, P_2, \ldots, P_K\}$ ▷ Ensemble des chemins candidats vers les UPF
$\mathcal{L} \leftarrow$ `matrice_liens_réseau`
$\text{load}(l) \leftarrow 0$ pour tout $l \in \mathcal{L}$ ▷ Charge initiale des liens

PHASE 1: Construction du score de coût réseau pondéré
for chaque lien $l \in \mathcal{L}$ **do**
    $\text{cost}(l) \leftarrow \alpha \cdot \text{lat}(l) + \beta \cdot \text{dis}(l) + \gamma \cdot \text{load}(l) + \delta \cdot \frac{1}{\text{cap}(l)}$
end for
PHASE 2: Évaluation dynamique des chemins vers les UPF
for chaque chemin $P_k \in \mathcal{P}$ **do**
    $\text{cost}(P_k) \leftarrow \sum_{l \in P_k} \text{cost}(l)$ ▷ Coût total d'accès à un UPF
end for
PHASE 3: Attribution dynamique des UE aux UPF
for chaque UE $u_i \in U$ **do**
    $P^*(u_i) \leftarrow \arg\min_{P_k \in \mathcal{P}} \text{cost}(P_k)$ ▷ Sélection UPF optimal
    Assigner u_i au chemin $P^*(u_i)$
end for
PHASE 4: Adaptation en ligne - mise à jour du réseau
for chaque UE assigné u_i **do**
    for chaque lien $l \in P^*(u_i)$ **do**
        $\text{load}(l) \leftarrow \text{load}(l) + 1$ ▷ Mise à jour de la charge
    end for
end for

PHASE 5: Regroupement adaptatif par chemins/UPF
for chaque UPF P_k **do**
    $G_k \leftarrow \{u_i \in U \mid P^*(u_i) = P_k\}$ ▷ Groupe d'UE par UPF
end for
return $\{G_k \mid k = 1, \ldots, K\}$ ▷ Groupes d'UE formés
PHASE 6: Optimisation globale
Définir la fonction d'attribution $f : U \to \mathcal{P}$
for chaque UE u_i **do**
    $f(u_i) \leftarrow P^*(u_i) = \arg\min_{P_k \in \mathcal{P}} \left(\sum_{l \in P_k} \text{cost}(l) \right)$
    où $\text{cost}(l) = \alpha \cdot \text{lat}(l) + \beta \cdot \text{dis}(l) + \gamma \cdot \text{load}(l) + \delta \cdot \frac{1}{\text{cap}(l)}$
end for
return fonction d'attribution optimale f, charges mises à jour $\{\text{load}(l) \mid l \in \mathcal{L}\}$

EXPÉRIMENTATION ET RÉSULTATS

Dans le cadre de cette expérimentation, nous avons simulé le déploiement d'un réseau 5G avancé dans une ville intelligente couvrant une surface de 10×10 km. Ce scénario intègre quatre zones spécialisées — une zone industrielle pour l'automatisation et les véhicules autonomes, une zone hospitalière pour des applications de télé-chirurgie, une zone portuaire dédiée à la logistique connectée, et un cluster technologique orienté R&D.

Objectif

L'objectif de cette phase d'expérimentation est d'évaluer l'impact d'emplacement des UPFs et l'intégration d'un mécanisme d'optimisation de l'emplacement et de la sélection des UPFs (User Plane Functions) sur les performances globales du réseau 5G simulé. Plus précisément, il s'agit de :
Mesurer les gains apportés par l'optimisation en termes de réduction de la latence de bout-en-bout,
Analyser l'équilibrage de la charge entre les différents UPFs,
Évaluer l'efficacité de la sélection dynamique des UPFs

Topologies testés

Topologie Simple: Cette topologie représente une architecture 5G simplifiée dimensionnée pour 6 utilisateurs simultanés. Elle se compose d'un plan de contrôle essentiel qui gère le réseau, et d'un plan

utilisateur stratifié comprenant quatre UPFs (2 UPF intermédiaires, 2 UPF d'ancrage/terminaison) distribués pour optimiser l'acheminement des données vers un serveur local.

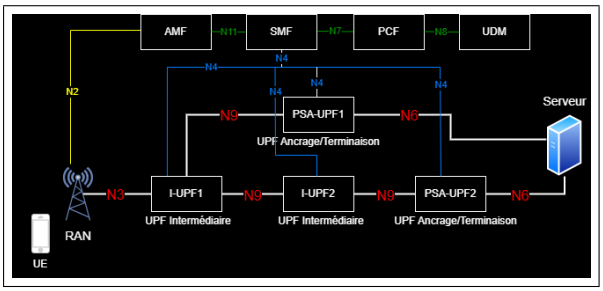

Figure 9: *Topologie simple*

Topologie Complexe: Cette topologie représente une architecture 5G dimensionnée pour 20 util- isateurs simultanés. Elle se compose d'un plan de contrôle essentiel qui gère le réseau, et d'un plan utilisateur stratifié comprenant huit UPFs (5 UPF intermédiaires, 3 UPF d'ancrage/terminaison) distribués pour optimiser l'acheminement des données vers un serveur local.

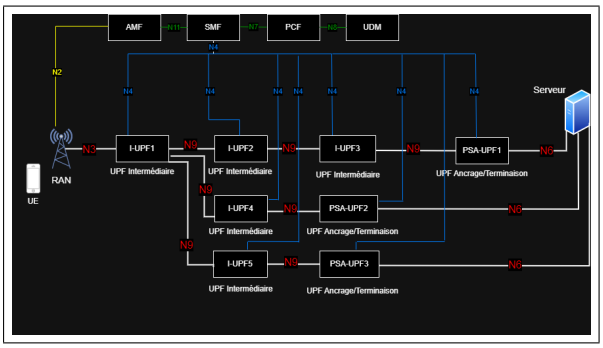

Figure 10: *Topologie complexe*

Scénario testés

Scénario 1 : Topologie par défaut, sans aucun mécanisme d'optimisation.
Scénario 2 : Sélection dynamique de l'UPF optimal pour chaque utilisateur (Utilisation de l'algorithme d'optimisation de sélection).
Scénario 3 : sélection dynamique + placement dynamique des UPF dans le réseau (Utilisation de l'algorithme d'optimisation de sélection et d'emplacement).

Métriques d'évaluation

Latence (ms) : Le temps total nécessaire pour qu'un paquet de données voyage de la source à la destination.
Jitter (ms) : La variation du délai de transmission entre les paquets, mesurant l'instabilité de la latence.
Débit : La quantité de données transmises par unité de temps, généralement exprimée en bits par seconde (bps).
Perte (%) : Le pourcentage de paquets de données qui n'atteignent pas leur destination et sont perdus durant la transmission.
Délai inter-paquet (ms) : L'intervalle de temps entre l'envoi ou la réception de deux paquets consécutifs.

Méthode de collecte et d'évaluation des performances

L'évaluation des performances a été réalisée à travers une série de tests de bout en bout entre un **UE (User Equipment)** et un **serveur de données applicatif**. Le protocole de test repose sur l'émission de requêtes d'interrogation (type requêtes HTTP ou personnalisées), simulant un trafic applicatif URLLC en conditions réalistes.
Les échanges (envoi et réception) entre l'UE et le serveur ont été **capturés en temps réel** à l'aide de l'outil **tcpdump**, générant des fichiers de trace au format **.pcap**. Ces fichiers ont ensuite été analysés hors-ligne à l'aide de **Wireshark** pour une première visualisation, puis traités de manière automatisée via un script Python s'appuyant sur la bibliothèque **Scapy**.
Le script a permis :

- L'extraction des timestamps des paquets envoyés et reçus.
- Le calcul précis de la latence de bout en bout (RTT).
- La mesure de la fiabilité des transmissions (pertes, réémissions, etc.).
- Une analyse de la stabilité et de la congestion du lien, à travers des indicateurs temporels et statistiques dérivés des traces (écart-type des RTT, gigue, délais excessifs, etc.).

Résultats obtenus

Métrique	Scén. 1	Scén. 2-UE1	Scén. 3-UE2
Latence (ms)	21.84	12.68	14.69
Jitter (ms)	10.29	1.75	5.49
Débit	1.36	1.39	1.41
Perte (%)	0.0%	0.0%	0.0%
Délai inter-paquet (ms)	498.36	491.46	488.93

Table 1: *Résultats de performance pour la topologie simple*

Métrique	Scén. 1	Scén. 2	Scén. 3
Latence (ms)	23.1492	20.0945	17.1884
Jitter (ms)	8.4400	6.0942	7.4144
Débit	1.3753	1.3706	1.3651
Perte (%)	0.0%	0.0%	0.0%
Délai inter-paquet (ms)	494.79	495.56	496.48

Table 2: *Résultats de performance pour la topologie complexe*

Analyse et discussion

A partir de la topologie simple: L'implémentation de notre algorithme de sélection dynamique d'UPF, associé dans certains cas à un positionnement adaptatif, a démontré une amélioration significative de la qualité de service dans différents scénarios topologiques. En particulier, les résultats expérimentaux confirment que :

- **La latence moyenne:** est réduite jusqu'à 45%, améliorant la réactivité des applications.
- **La stabilité du réseau (jitter):** s'améliore de manière notable, avec des baisses atteignant 80%, ce qui est crucial pour les flux sensibles.
- **Le débit:** présente une légère augmentation mais plus régulière, traduisant une meilleure fluidité du trafic.
- **Le délai inter-paquet:** est également plus homogène, signe d'une transmission plus contrôlée et fluide.
- Enfin, aucune perte de paquets n'a été constatée dans les cas optimisés, attestant de la fiabilité des chemins choisis.

Cette variation s'explique par le fait que chaque utilisateur a emprunté une chaîne distincte de fonctions UPF, déterminée en fonction de critères tels que la latence estimée, la charge des UPFs, et la distance topologique par rapport à la station de base (gNB). Ainsi, l'algorithme a sélectionné pour UE1 une chaîne plus optimale en termes de latence (12.68 ms), tandis qu'UE2 a obtenu une chaîne légèrement moins performante (14.69 ms), mais toujours bien meilleure que celle du scénario non optimisé (21.84 ms).

A partir de la topologie complexe: En comparant les résultats on remarque que:

- La **latence** diminue clairement entre les scénarios : elle est la plus élevée sans optimisation (Scénario 1) et la plus faible avec les deux algorithmes (notamment le Scénario 3).
- Le **jitter** suit la même tendance, montrant une meilleure stabilité du trafic dans les scénarios optimisés.
- Le **débit** reste relativement stable, ce qui montre que l'amélioration ne se fait pas au détriment du volume de données transmises.
- Le **délai inter-paquet** devient plus régulier dans les scénarios optimisés, surtout avec le placement dynamique, traduisant un meilleur ordonnancement des paquets.
- **Aucune perte de paquet** n'a été constatée, ce qui confirme la fiabilité du réseau dans tous les cas.

Ces résultats confirment les observations précédentes concernant l'efficacité de l'algorithme de sélection dynamique d'UPF. De plus, la combinaison de cette sélection avec un positionnement adaptatif des fonctions UPF permet une réduction encore plus significative de la latence, améliorant ainsi la qualité de service globale du réseau.

Surveillance et Visualisation du Système

Afin de garantir une observation fine et continue du comportement du réseau simulé, une solution de monitoring en temps réel a été mise en place, combinant des outils de collecte de métriques système et réseau avec une plateforme de visualisation dynamique.

Surveillance

La surveillance du système repose sur la collecte automatique de métriques provenant des différentes couches de l'infrastructure :

- **Node Exporter :** installé sur la machine hôte, cet agent permet de surveiller l'état du système (charge CPU, utilisation mémoire, activité disque, etc.).
- **cAdvisor (Container Advisor) :** intégré avec Docker, il expose des métriques détaillées au niveau des conteneurs, telles que l'utilisation des ressources (CPU, mémoire, I/O) pour chaque service virtualisé (gNB, UPF, Core 5G, etc.).
- Toutes les métriques sont centralisées et enregistrées dans une base de données temporelle via Prometheus, qui interroge périodiquement les exportateurs et stocke les données sous forme de séries temporelles.

Visualisation

Pour permettre une analyse visuelle intuitive et interactive, l'outil Grafana a été utilisé en complément de Prometheus. Il permet de :

- Créer et configurer des tableaux de bord personnalisés pour chaque composant (UE, UPF, Core, etc.).
- Générer automatiquement des graphiques de performance réseau, tels que l'évolution de la

latence, la charge des UPFs, ou encore la répartition du trafic.
- Suivre en temps réel le comportement du réseau simulé, identifier les points de congestion et observer les effets des différentes politiques de sélection/placement des UPFs.

L'intégration de ces outils assure ainsi une visibilité complète sur l'environnement de test, tout en fournissant un support empirique pour l'analyse des résultats et la validation des hypothèses expérimentales.

Conclusion

Cette étude a permis de démontrer la faisabilité et l'efficacité d'une approche de simulation complète du réseau 5G Core, en mettant l'accent sur le placement optimisé et la sélection dynamique des fonctions de plan utilisateur (UPF). Grâce à l'intégration de Free5GC et UERANSIM dans un environnement conteneurisé, nous avons pu modéliser fidèlement les mécanismes du 5GC et reproduire des scénarios réalistes de trafic dans des contextes critiques comme les applications uRLLC. Les algorithmes proposés, basés sur des approches ILP et RealisticNOUP, ont permis d'identifier des configurations de placement des UPFs réduisant significativement la latence et améliorant la stabilité du réseau, sans compromettre le débit ni la fiabilité. Les performances mesurées dans des topologies simples et complexes ont confirmé que l'association d'une sélection dynamique des UPFs à un placement adaptatif constitue une solution robuste pour répondre aux exigences des réseaux 5G critiques. Ces résultats ouvrent la voie à des travaux futurs portant sur l'intégration de méthodes d'intelligence artificielle pour une optimisation en temps réel, ou encore sur l'étude de scénarios multi-slices et interdomaines dans des réseaux 5G privés.

References

[1] Free5GC Team. *Free5GC: Open Source 5G Core Network*. Accédé le: 2024-01-15. 2023. URL: https://free5gc.org/.

[2] UERANSIM. *UERANSIM: Open Source 5G UE and RAN (gNodeB) Simulator*. Accédé le: 2024-01-15. 2023. URL: https://github.com/aligungr/UERANSIM.

[3] M. Sallouha et al. "Open-source 5G testbeds: Opportunities, Challenges, and the Road Ahead". In: *IEEE Communications Surveys & Tutorials* 24.2 (2022), pp. 1294–1324. DOI: 10.1109/COMST.2022.3155217.

[4] B. Han, S. Yi, and Y. Zhang. "Free5GC-based Testbed for Exploring End-to-End Slicing in 5G Networks". In: *Proc. of IEEE NetSoft*. 2021, pp. 123–130. DOI: 10.1109/NetSoft51509.2021.9492602.

[5] S. Iqbal and J. M. Hamamreh. "A comprehensive tutorial on how to practically build and deploy 5G networks using open-source software and general-purpose, off-the-shelf hardware". In: *RS Open Journal on Innovative Communication Technologies* (2021). PDF.

[6] A. B. N. Linh et al. *Analysing open-source 5G core networks for TLS vulnerabilities and 3GPP compliance*. PDF. 2023.

[7] I. Rojas Pérez. *5G-enabled edge deployments for emerging real-time services*. PDF. 2024.

[8] A. S. Tanenbaum and D. J. Wetherall. *Computer Networks*. 5th. Chapitre sur les délais de transmission. Pearson, 2010.

[9] Dirk Merkel. "Docker: Lightweight Linux Containers for Consistent Development and Deployment". In: *Linux Journal*. Vol. 2014. 239. 2014.