ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Desarrollo de un prototipo de red 5G Standalone con servicios *Voice Over New Radio* (VoNR) mediante OpenAirInterface

AUTOR: Jairo Tarro Guerrero

TITULACIÓN: Grado en Ingeniería Telemática

TUTOR: Carlos Ramos Nespereira

DEPARTAMENTO: Ingeniería Telemática y Electrónica (DTE)

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Marta Gil Barba

TUTOR: Carlos Ramos Nespereira

SECRETARIO: Pedro Castillejo Parrilla

Fecha de lectura:

Calificación:

El Secretario,

Agradecimientos

Me gustaría aprovechar esta oportunidad para expresar mi agradecimiento a todas aquellas personas que siempre han confiado en mí y que durante tantos años me han estado apoyando incondicionalmente. No me cabe la menor duda de que sin su apoyo y confianza no lo habría logrado.

A mi abuelo Calixto. Estés donde estés, muchas gracias por enseñarme a no dar por sentadas las cosas que nos rodean e ir siempre más allá, más adentro y descubrir cuales son los mecanismos que mueven el mundo tal y como lo conocemos. Gracias por contagiarme esa curiosidad insaciable y, sobre todo, gracias por apoyarme hasta el último de tus días.

A mis padres por su lucha constante, por todos los sacrificios hechos para que pudiese cumplir mis objetivos siempre en las mejores condiciones, por su confianza inquebrantable, por apoyarme continuamente, aún en mis peores momentos y por darme una educación de la cual me siento orgulloso.

A mis tíos por su infinita paciencia y apoyo continuo. Muchas gracias por abrirme las puertas de vuestra casa, me habéis enseñado mucho a lo largo de estos años, atesoro cada uno de vuestros consejos y momentos compartidos hasta el día de hoy, nada de esto habría sido posible sin vosotros.

A Isabel, gracias por tu infinita bondad y ayuda, especialmente en los momentos más difíciles de estos últimos meses.

A todos mis primos, que de una forma u otra han estado siempre a mi lado, muchísimas gracias.

A mis compañeros de universidad, con los que he compartido innumerables momentos y experiencias y, que, sin lugar a dudas, han hecho esta etapa de mi vida inolvidable.

Finalmente, también quiero agradecer a todos mis profesores su vocación por la enseñanza y esfuerzo para transmitir todos los conocimientos a su alumnado, del cual he tenido el privilegio de formar parte. Especialmente quiero agradecer a mi tutor Carlos su paciencia y ayuda para lograr culminar este proyecto y poder cerrar esta etapa.

A todos vosotros, que habéis sido, sois y seréis siempre parte de mí, de corazón,

MUCHAS GRACIAS.

Resumen

Las redes de comunicaciones móviles constituyen una gran parte de las redes de telecomunicación en nuestros días, no solo por el número de usuarios a los cuales se les presta servicio, sino también por las tecnologías que están en continuo desarrollo y que se sustentan sobre este tipo de redes. Actualmente, la tecnología 5G se ha establecido como la herramienta fundamental en el desarrollo tecnológico de la sociedad y de las comunicaciones móviles tal y como las conocemos.

Este desarrollo ha sido impulsado, en su mayor parte, por la necesidad de una población cada vez más conectada y exigente en cuanto a prestaciones de servicio. Así mismo, no debemos olvidar el continuo crecimiento del *Internet of Things* (IoT), cada día más presente en nuestro entorno y que, sin ninguna duda, es uno de los actores principales en el desarrollo de la tecnología 5G. Tampoco debemos pasar por alto la reciente situación provocada por el COVID-19, el cual ha dejado claras las ventajas de la investigación y el desarrollo en las tecnologías de comunicaciones móviles y la importancia de contar con una infraestructura sólida, escalable y descentralizada.

Una de las características clave de la tecnología 5G y motivo principal por el cual su despliegue es mucho menos costoso que el de redes de generaciones anteriores, es la virtualización de funciones de red o *Network Function Virtualisation* (NFV), que abre todo un abanico de posibilidades para la gestión e implementación de redes. Un claro ejemplo lo encontramos con los servicios virtualizados de voz conocidos como *Voice Over New Radio* (VoNR), que suponen un avance importante con respecto a su predecesor, *Voice Over Long Term Evolution* (VoLTE) en términos de escalabilidad, latencia, seguridad y calidad.

Por lo tanto, viendo la importancia que las redes móviles 5G tienen en nuestros días y que previsiblemente aumentará a lo largo de los años venideros, resulta imprescindible que todos los estudiantes que aspiren a ser titulados en el Grado de Ingeniería Telemática puedan y deban formarse en este tipo de tecnologías. En este contexto resulta muy interesante el desarrollo y despliegue de prototipos de red 5G en entornos virtualizados, con *software* completamente libre, de forma que los estudiantes puedan adquirir todos los conocimientos básicos y, sobre todo, experimentar con este tipo de tecnologías.

Durante la realización de este proyecto se han desplegado tres topologías diferentes de red *5G Standalone* haciendo uso de máquinas virtuales mediante el *software VirtualBox* y contenedores a través del *software Docker*. Sobre los escenarios de red desarrollados, se deben desplegar servicios de voz virtualizados (VoNR) conforme a las especificaciones del *Third Generation Partnership Project* (3GPP) para su aplicación en entornos didácticos. Sin embargo, la madurez actual de las tecnologías y soluciones *open source* no han permitido integrar y desplegar este tipo de servicios.

Se han analizado varias soluciones de código abierto para 5G SA de las cuales, justificando su elección, se han seleccionado:

- *OpenAirInterface* (OAI) para el despliegue del núcleo de red 5G SA.
- $UERANSIM$ y *gnbsim* para la implementación de los componentes gNB y UE de cada escenario.

Así mismo, se han analizado el resto de las tecnologías que han sido necesarias para el despliegue de los distintos desarrollos de este proyecto.

Para finalizar el proyecto, se ha elaborado un conjunto de pruebas que posibilitan el análisis de los prototipos de red desarrollados y así determinar si son válidos para su uso en entornos docentes. Tras la realización de dichas pruebas, se ha concluido que, a pesar de no ofrecer las capacidades necesarias para el despliegue de servicios VoNR, las soluciones analizadas presentan un gran potencial didáctico y cumplen con los objetivos básicos para establecerse como herramientas en entornos docentes. Así mismo se recomienda el seguimiento de estos desarrollos, ya que es posible que en un futuro próximo sí que ofrezcan la capacidad suficiente para implementar servicios de VoNR.

Palabras clave

5G, *5G Stand Alone* (5G SA), *5G Non Stand Alone* (5G NSA), *Voice Over New Radio* (VoNR), *Radio Access Network* (RAN), *New Radio* (NR), *Next Generation Node B* (gNB), *5G Next-gen Core* (5G NGC), *OpenAirInterface* (OAI), $Free5GC$, $UERANSIM$, $gnbsim$, $Kamailio$, *Network Function Virtualisation* (NFV), *Software Defined Networking* (SDN), *User Equipment* (UE), *Docker*, *Open Source MANO* (OSM).

Abstract

Mobile communication networks constitute a large part of today's telecommunication networks, not only because of the number of users served, but also because of the continuously developing technologies that are based on this type of network. Today, 5G technology has established itself as the fundamental tool in the technological development of society and mobile communications as we know it.

This development has been driven, for the most part, by the needs of an increasingly connected and demanding population in terms of service provision. Likewise, we must not forget the continuous growth of IoT, which is increasingly present in our environment and is undoubtedly one of the main players in the development of 5G technology. Nor should we overlook the recent situation caused by COVID-19, which has made clear the advantages of research and development in mobile communications technologies and the importance of having a solid, scalable and decentralised infrastructure.

One of the key features of 5G technology and the main reason why it is much less costly to deploy than previous generation networks is the network functions virtualisation or NFV, which opens up a whole range of possibilities for network management and deployment. A clear example of this is the virtualised voice services known as VoNR, which represent a significant advance over its predecessor, VoLTE, in terms of scalability, latency, security and quality.

Therefore, given the importance that 5G mobile networks have nowadays and that is expected to increase over the coming years, it is essential that all students who aspire to be graduates in the Degree in Telematics Engineering can and should be trained in this type of technology. In this context, the development and deployment of 5G network prototypes in virtualised environments, with completely free *software*, is very interesting, so that students can acquire all the basic knowledge and, above all, experiment with this type of technology.

During this project, three different 5G SA network topologies have been deployed using virtual machines through the *VirtualBox software* and containers through the *Docker software*. On top of the developed network scenarios, virtualised voice services (VoNR) are to be deployed according to 3GPP specifications for application in educational environments. However, the current maturity of *open source* technologies and solutions has not allowed the integration and deployment of such services.

A number of open source solutions have been analysed for 5G SA of which, justifying their choice, the following have been selected:

- OAI for 5G SA network core deployment and,

- *UERANSIM* and *gnbsim* for the implementation of the radio access network.

Likewise, the rest of the technologies that have been necessary for the deployment of the different developments of this project have been analysed.

To conclude the project, a set of tests has been developed to analyse the network prototypes developed and thus determine whether they are valid for use in educational environments. After carrying out these tests, it has been concluded that, despite not offering the necessary capabilities for the deployment of VoNR services, the solutions analysed have a great didactic potential and meet the basic objectives to establish themselves as tools in teaching environments. It is also recommended that these developments be monitored, as it is possible that in the near future they may offer sufficient capacity to implement VoNR services.

Key words

5G, *5G Stand Alone* (5G SA), *5G Non Stand Alone* (5G NSA), *Voice Over New Radio* (VoNR), *Radio Access Network* (RAN), *New Radio* (NR), *Next Generation Node B* (gNB), *5G Next-gen Core* (5G NGC), *OpenAirInterface* (OAI), *Free5GC*, *UERANSIM*, *gnbsim*, *Kamailio*, *Network Function Virtualisation* (NFV), *Software Defined Networking* (SDN), *User Equipment* (UE), *Docker*, *Open Source MANO* (OSM).

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

1G Primera generación de comunicaciones móviles

2G Segunda generación de comunicaciones móviles

3GPP *Third Generation Partnership Project*

3G Tercera generación de comunicaciones móviles

4G Cuarta generación de comunicaciones móviles

5G NGC *5G Next-gen Core*

5G NSA *5G Non Stand Alone*

5G SA *5G Stand Alone*

5G Quinta generación de comunicaciones móviles

AF *Application Function*

AGPL *Affero General Public License*

AMF *Access and Mobility Management Function*

AMPS *Advance Mobile Phone Service*

ARQ *Automatic Repeat Request*

AS *Access Stratum*

AUSF *Authentication Server Function*

CDMA *Code Division Multiple Access*

CN *Core Network*

DRA *Diameter Routing Agent*

DTE Departamento de Ingeniería Telemática y Electrónica

E2E *End to End*

EDGE *Enhanced Data rates for GSM Evolution*

EPC *Evolved Package Core*

EP *Elementary Procedure*

ETSI *European Telecommunications Standards Institute*

FDMA *Frequency Division Multiple Access*

GPG *GNU Privacy Guard*

GPRS *General Packet Radio Services*

GSA *Global mobile Suppliers Association*

GSM *Global System for Mobile Communication*

GTP-U *GPRS Tunnelling Protocol*

HSS *Home Subscriber Server*

HTTP *Hypertext Transfer Protocol Secure*

HTML *Hypertext Markup Language*

IMSI *International Mobile Subscriber Identity*

IMS *IP Multimedia Subsystem*

IP *Internet Protocol*

IWF *Interworking Function*

IoT *Internet of Things*

LTE-M *Long Term Evolution for Machine Type Communication*

LTE *Long Term Evolution*

LTS *Long Term Support*

MAC *Medium Access Control*

MANO *Management and Orchestation*

MMS *Multimedia Messaging Service*

MTC *Machine Type Communications*

NAS *Non Access Stratum*

NB-IoT *Narrowband - Internet of Things*

NEF *Network Exposure Function*

NFVi *Network Function Virtualisation infraestructure*

NFV *Network Function Virtualisation*

NG-AP *NG Application Protocol*

NR NSA *New Radio Non Stand Alone*

NR SA *New Radio Stand Alone*

NRF *Network Repository Function*

NRF *NR Repository Function*

NR *New Radio*

NSA *Non Stand Alone*

NSSF *Network Slice Selection Function*

NS *Network Slicing*

OAI *OpenAirInterface*

OFDMA *Orthogonal Frecuency Division Multiple Access*

OSA *OpenAirInterface Software Alliance*

OSM *Open Source MANO*

PCF *Policy Control Function*

PCRF *Policy and Charging Rules Function*

PDCP *Packet Data Convergence Protocol*

PDU *Protocol Data Unit*

PFCP *Packet Forwarding Control Protocol*

QFI *QoS Flow ID*

QoS *Quality of Service*

RAN *Radio Access Network*

RLC *Radio Link Control*

RRC *Radio Resource Control*

SA *Stand Alone*

SBA *Service Based Arquitecture*

SC-FDMA *Single Carrier - Frecuency Division Multiple Access*

SCTP *Stream Control Transmission Protocol*

SDAP *Service Data Adaptation Protocol*

SDN *Software Defined Networking*

SIM *Subscriber Identity/Identification Module*

SIP *Session Initiation Protocol*

SMF *Session Management Function*

SMS *Short Message Service*

TCP *Transmission Control Protocol*

TDMA *Time Division Multiple Access*

TLS *Transport Layer Security*

UDM *Unified Data Management*

UDP *User Datagram Protocol*

UDR *Unified Data Repository*

UE *User Equipment*

UMTS *Universal Mobile Telecommunication Service*

UPF *User Plane Function*

UPM Univesidad Politécnica de Madrid

URLLC *Ultra-Reliable and Low Latency Communications*

VNF *Virtual Network Function*

VNF *Virtual Network Function*

VoIP *Voice over IP*

VoLTE *Voice Over Long Term Evolution*

VoNR *Voice Over New Radio*

WCDMA *Wideband Code Division Multiple Access*

eMBB *Enhanced Mobile Broadband*

eNB *eNodeB*

gNB *Next Generation Node B*

mMTC *Massive Machine Type Communications*

Capítulo 1

Introducción

Actualmente las redes de comunicaciones móviles son una parte vital de las redes de telecomunicación. Históricamente, la característica que ha diferenciado a este tipo de redes frente a las demás, ha sido la gran flexibilidad que proporciona a los usuarios finales. Sin embargo, con la aparición de 5G la "flexibilidad" es solo una de las muchas características que esta quinta generación de comunicaciones móviles aporta.

La tecnología 5G da cabida a un vasto número de nuevas aplicaciones y casos de uso con unos niveles de exigencia nunca antes vistos. Sin duda las altas velocidades de transferencia de datos, la latencia ultra baja, la excepcional fiabilidad, el bajo consumo energético y la reducción de los costes, han sido solo algunas de las características que han atraído a numerosas industrias, como puede ser la industria del automóvil, empresas de seguridad, industria logística, etc. [1]

Indudablemente, las redes de comunicaciones móviles están transformando no solo la sociedad y nuestros estilos de vida, sino también la industria y todos sus procesos. Este hecho se puede observar a nivel mundial en los datos publicados por *Ericsson* en su último informe de noviembre de 2021. [2]

En la figura 1.1 podemos observar una gráfica con el número de suscriptores móviles en todo el mundo clasificados por la tecnología que les da servicio. En esta figura se puede apreciar que la tecnología 4G sigue siendo la predominante entre los usuarios, incluso frente a 5G.

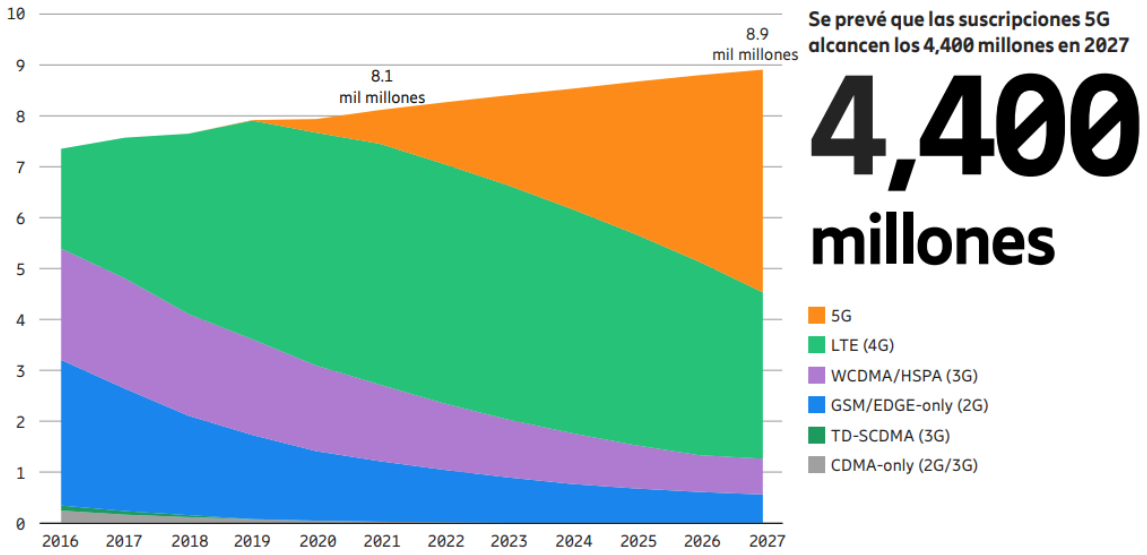


Figura 1.1. Número de suscripciones móviles clasificadas por tecnología [2]

No obstante, y a pesar de estos datos, como vemos en la figura 1.2, el crecimiento de la tecnología 5G presenta una pendiente mucho mayor que la del resto de generaciones anteriores. Esto se traduce a que muy probablemente el número de suscriptores 5G supere a los suscriptores de 4G en un espacio corto de tiempo y, entre las razones principales que causan este efecto, encontramos las mencionadas anteriormente: reducción del coste de despliegue, creación de nuevos mercados, nuevas aplicaciones, casos de uso, la inclusión de la industria y nuevos servicios.

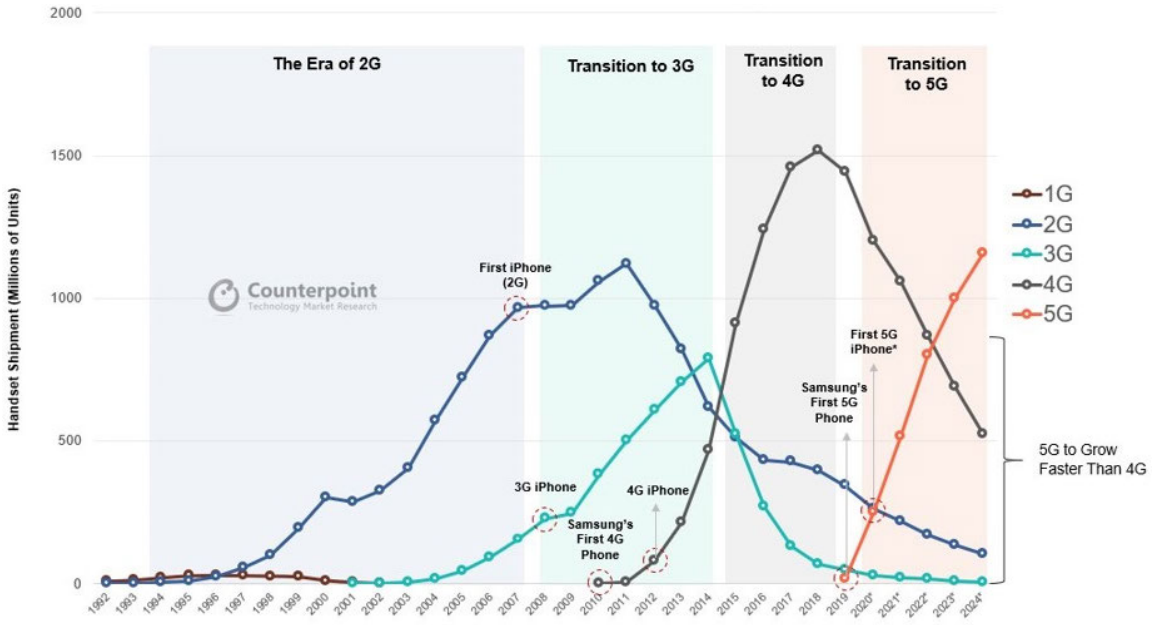


Figura 1.2. Evolución de suscripciones móviles clasificadas por tecnología [3]

Según las previsiones realizadas por *Ericsson*, para el 2027 se estima que habrá 8,8 mil millones de suscripciones en todo el mundo, de las cuales, 4,4 mil millones pertenecerán a 5G. Dibujando estas cifras sobre el mapa mundial, por región arrojarían un resultado como el observado en la figura 1.3.

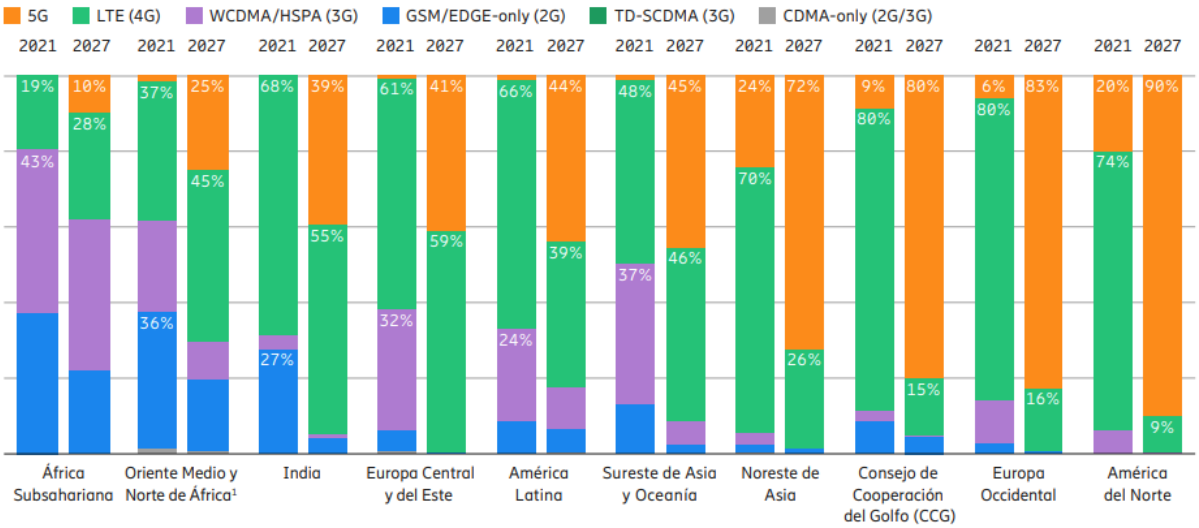


Figura 1.3. Evolución de suscripciones móviles clasificadas por tecnología [2]

1.1. Objetivos

El objetivo principal del proyecto consiste en desarrollar una red *5G Stand Alone* (5G SA) acorde a las especificaciones del *Third Generation Partnership Project* (3GPP) haciendo uso de entornos virtualizados. Además, se persigue el despliegue de servicios de voz sobre 5G, más conocidos como *Voice Over New Radio* (VoNR).

Para cumplir con el objetivo principal del proyecto se han definido los siguientes objetivos específicos, para los cuales se usará exclusivamente herramientas *software* libre.

- Despliegue del núcleo de red 5G mediante el uso del *software* proporcionado por *OpenAirInterface* (OAI).
  - Diseñar escenarios de red que implementen las funcionalidades básicas de una red 5G con propósitos didácticos.
  - Instalar y configurar los elementos *software* que forman una red 5G en un entorno virtual.
  - Realizar pruebas sobre los prototipos de red propuestos con el fin de evaluar la madurez actual de las tecnologías que lo componen y su potencial en escenarios para uso docente.

- Despliegue de las funciones para la red de acceso radio
  - UERANSIM
  - gnbsim
- Despliegue de funciones de red, en nuestro caso, servicios de VoNR.

Finalmente, del desarrollo de este proyecto se espera la integración exitosa de las tecnologías que lo soportan, de forma que se obtenga una arquitectura de red 5G SA funcional y, que pueda ser utilizada en entornos docentes. Concretamente en la asignatura Redes de Comunicaciones Móviles, impartida en la titulación de Grado en Ingeniería Telemática e incluso en asignaturas de postgrado.

1.2. Estructura de la memoria

El objetivo de este apartado consiste en proporcionar una descripción breve de la estructura y los contenidos que se incluyen en la memoria. De esta forma se pretende facilitar al lector el acceso a la información contenida en este documento.

La memoria se divide en los siguientes capítulos:

Capítulo 1 - Introducción

En este capítulo se expone la situación actual a nivel mundial de las redes de comunicaciones móviles, haciendo hincapié en la importancia dentro de la sociedad. Además, se presenta la motivación del estudio y desarrollo del presente proyecto. Finalmente se describe la estructura de la memoria de este.

Capítulo 2 - Marco tecnológico

En este capítulo se introduce al lector dentro del contexto que enmarca este proyecto. Concretamente, se realiza una descripción breve de las tecnologías involucradas o que tienen una relación directa con este. Al cierre del capítulo, se realiza una valoración de las distintas soluciones actuales concluyendo, de forma razonada, que tecnologías han sido seleccionadas para el desarrollo del proyecto.

Capítulo 3 - Especificaciones y restricciones de diseño

En esta sección se presenta de forma concisa las especificaciones y las restricciones de los prototipos de red a desplegar.

Capítulo 4 - Descripción de la solución propuesta

En este capítulo se elabora una descripción detallada de la solución propuesta para cada prototipo de red planteado. Se profundiza en la arquitectura interna de los distintos elementos de los sistemas a implementar, así como su integración. Finalmente, también se encontrarán las tareas a seguir para el despliegue exitoso de los modelos de red propuestos.

Capítulo 5 - Resultados

Este capítulo se centra en la evaluación del comportamiento de los prototipos de red diseñados a través de una serie de pruebas. Las pruebas consisten en explotar las funciones ofrecidas por los distintos elementos que componen la red, así como la captura del flujo de datos generado por la red, analizando a través de él, su comportamiento.

Capítulo 6 - Presupuesto

En esta sección se detalla la inversión requerida para la realización del proyecto. Para el cálculo de los costes, se tendrán en cuenta los recursos *hardware*, *software* y humanos.

Capítulo 7 - Conclusión y trabajos futuros

En este apartado se evalúan los resultados finales obtenidos tanto del desarrollo de la solución, como de los resultados obtenidos a través del análisis llevado a cabo con la batería de pruebas. En base a las conclusiones obtenidas, se definirán las líneas de futuros trabajos.

ANEXOS

Anexo A - Preparación para la instalación En este anexo se incluyen todos los pasos necesarios para la obtención e instalación de los componentes requeridos para el despliegue de los distintos prototipos de red propuestos.

Anexo B - Configuración para desarrollo de escenarios didácticos En este anexo se encuentran todos los pasos de configuración detallados por cada escenario didáctico propuesto en el desarrollo del proyecto.

Anexo C - Obtención de resultados y corrección de errores En este anexo se ubican los pasos y resultados detallados, obtenidos durante el procedimiento de evaluación y obtención de resultados de los diferentes escenarios de red.

Anexo D - Ficheros de configuración Este anexo contiene todos los ficheros de configuración completos de los distintos elementos que componen la red, para cada escenario desarrollado.

Capítulo 2

Marco tecnológico

2.1. Introducción

En este capítulo se abordará la descripción del marco tecnológico en el cual se sitúa el proyecto. Para lograrlo, se realizará un repaso histórico de todas las tecnologías que, a lo largo de los años, se han desarrollado en el área de las comunicaciones móviles. Posteriormente, se realizará una introducción de la tecnología 5G y sus principales especificaciones, es decir: arquitectura, protocolos, características generales, importancia de la tecnología, etc. Por último, se describen las principales implementaciones de la tecnología 5G con sus características y se justificará la elección de cada solución.

2.2. Evolución desde redes 1G, 2G, 3G y 4G. Cambios conceptuales

La primera generación de comunicaciones móviles conocida como 1G vio la luz en 1984 y trajo consigo los servicios de voz a través de las ondas de radio. [4] La tecnología que sustentaba a esta primera generación se denomina *Advance Mobile Phone Service* (AMPS) o Servicio Avanzado de Telefonía Móvil, que se basaba en un sistema de acceso radio conocido como *Frequency Division Multiple Access* (FDMA) o Acceso Múltiple por División en Frecuencia, y solo permitía realizar llamadas de voz.

A pesar del gran avance que supuso esta tecnología, presentaba algunos inconvenientes, entre ellos la escasa o nula seguridad de las comunicaciones que, además presentaban una calidad muy baja. Por otra parte, requería de un gran consumo energético, lo que se traducía en una baja durabilidad de las baterías. En esta primera generación no se incluyeron servicios de datos, es decir, no se trabajó con señales digitales.

La segunda generación de comunicaciones móviles, más conocida como 2G, fue introducida en la década de 1990, con ella se hizo posible la transmisión de señales digitales utilizando dos esquemas en el acceso radio. Estos son: *Time Division Multiple Access* (TDMA) o Acceso Múltiple por División en el Tiempo y *Code Division Multiple Access* (CDMA) o Acceso Múltiple por División de Código. La tecnología que se encuentra detrás de esta generación se denomina *Global System for Mobile Communication* (GSM) o Sistema Global de Comunicación Móvil y fue el estándar móvil más utilizado, además de ser el primero en permitir *roaming* internacional. [4]

El 2G o GSM, incorporó servicios novedosos en aquel momento de los cuales muchos de ellos seguimos utilizando hoy en día: *roaming* internacional, *Multimedia Messaging Service* (MMS), *Short Message Service* (SMS), conferencias telefónicas, entre otros. La principal limitación de GSM reside en la baja velocidad de transferencia de datos, debido a que el núcleo de la red funcionaba bajo conmutación de circuitos. Posteriormente se incorporaron las tecnologías *General Packet Radio Services* (GPRS) y *Enhanced Data rates for GSM Evolution* (EDGE) que habilitaron el núcleo de la red 2G para funcionar bajo conmutación de paquetes. A pesar de estas mejoras, no fue posible hacer compatible esta tecnología con el manejo de datos complejos, como puede ser el contenido multimedia.

En el año 2000, fue introducida la que se conoce como la tercera generación de comunicaciones móviles o 3G. Su principal objetivo fue ofrecer mayores velocidades de transferencia de datos con una inversión mínima en infraestructuras. La aparición de las redes 3G impulsaron el desarrollo de los teléfonos inteligentes, ya que gracias a esta nueva generación se posibilitaba el acceso a la navegación web, correo electrónico, transferencia de contenido multimedia, etc.

La tecnología que mueve a esta tercera generación se denomina en Europa, *Universal Mobile Telecommunication Service* (UMTS) o Sistema Universal de Telecomunicaciones Móviles. [4] UMTS introdujo un nuevo método de acceso denominado *Wideband Code Division Multiple Access* (WCDMA) o Acceso Múltiple por División de Códigos de Banda Ancha, que permite la transmisión simultánea en el dominio del tiempo de múltiples terminales de usuario. Finalmente, cabe destacar que UMTS utilizaba el mismo núcleo de red que usaban GSM/GPRS/EDGE. UMTS empleaba conmutación de circuitos para los servicios en tiempo real, tales como las comunicaciones de voz, y conmutación de paquetes para los servicios de datos.

Justo en el momento en el que 3G comenzaba a implantarse, en el año 2004, se inició también la estandarización del 4G, que hace referencia a la tecnología LTE, toda una nueva generación basada en *Internet Protocol* (IP). La tecnología 4G nace con el objetivo principal de ofrecer servicios de alta velocidad, calidad, capacidad, seguridad y bajo costo para servicios de voz y datos, sobre IP. La razón principal por la que se decidió realizar una transición a "todo IP" fue establecer una plataforma común a todas las tecnologías desarrolladas hasta ese momento. Es por esto por lo que desde 4G todo el tráfico de la red pasa a ser por conmutación de paquetes, empleando para ello el protocolo de red IP y, en consecuencia, el núcleo de red pasa a funcionar exclusivamente en conmutación de paquetes, desapareciendo la parte de conmutación de circuitos

desde esta cuarta generación en adelante.

La red 4G introduce una nueva tecnología de acceso en el enlace radio, concretamente en el enlace descendente se introduce *Orthogonal Frecuency Division Multiple Access* (OFDMA) o Acceso Múltiple por División de Frecuencias Ortogonales y para el enlace ascendente *Single Carrier - Frecuency Division Multiple Access* (SC-FDMA) o Acceso Múltiple por División en Frecuencia con Portadora Única. La diferencia principal entre 3G y 4G radica en la transferencia de datos y en la calidad de la señal, además de que se suprime la conmutación de circuitos dentro del esquema de funcionamiento de la red 4G.

Finalmente, en el año 2017, llega la quinta generación de comunicaciones móviles, cuyo objetivo es ofrecer un nivel de conectividad y cobertura prácticamente sin limitaciones. Inicialmente el 5G está disponible mediante mejoras en las tecnologías previas LTE, LTE-Advance y LTE_Pro. Sin embargo, en el año 2019 se introduce una nueva interfaz radio que da paso al despliegue de redes 5G completamente independientes 5G SA.

En la figura 2.1, podemos observar la evolución de la arquitectura de red desde los inicios con GSM y UMTS, pasando por LTE, hasta llegar a la actualidad con 5G.

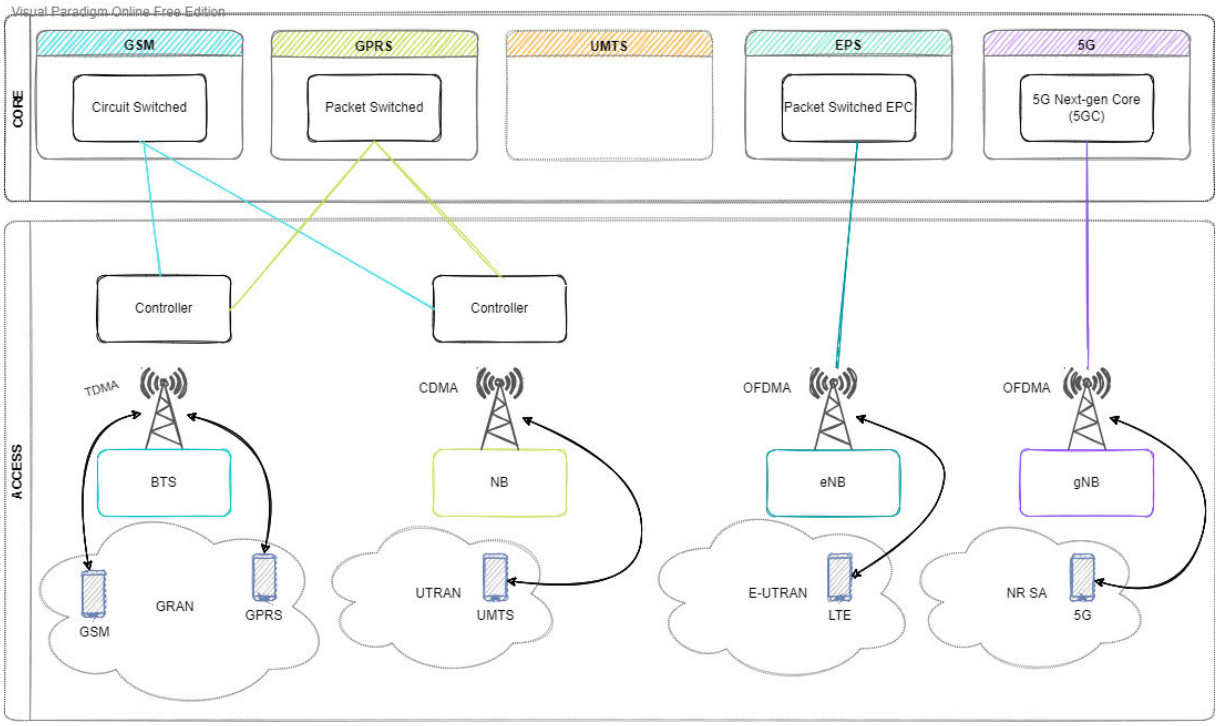


Figura 2.1. Evolución de la arquitectura en las redes de comunicaciones móviles. Elaboración propia, basada en [5][6]

La arquitectura de red 5G se encuentra dividida en dos bloques. En primer lugar, encontramos la red de acceso *New Radio* (NR) y en segundo lugar localizamos el *5G Next-gen Core* (5G NGC) o núcleo de red 5G, el cual añade tres características nuevas con respecto a generaciones anteriores:

1. Se trata de una ***Service Based Arquitecture*** **(SBA)** o Arquitectura Basada en Servicios, que se sustenta gracias al nuevo paradigma introducido con las *Network Function Virtualisation* (NFV).
2. Es compatible con ***Network Slicing*** **(NS)**.
3. Divide por completo el plano de control con respecto al plano de usuario. Esto último se trata de una característica que la tecnología 5G ha heredado de su predecesora (4G), sin embargo, la quinta generación de comunicaciones móviles lleva esta separación de ambos planos a un nivel de madurez mucho más elevado.

Todo esto da lugar a una arquitectura de red como la mostrada en la figura 2.2. En posteriores apartados se analizará en mayor profundidad las características, componentes y funcionalidades de la red 5G.

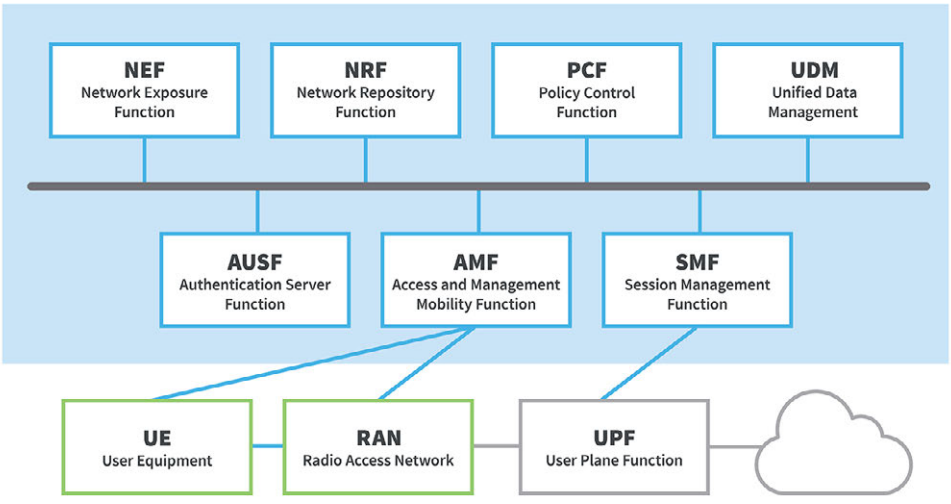


Figura 2.2. Diagrama básico de arquitectura de red 5G [7]

2.3. Tecnología 5G

El término 5G hace referencia al estándar de comunicaciones móviles desarrollado por el 3GPP. Esta quinta generación de comunicaciones móviles se presenta como una mejora de todas sus predecesoras, aunque todas ellas comparten un enfoque similar, ya que persiguen operar en rangos de espectro más amplios y con tasas de datos y capacidades más elevadas, objetivos que continúan siendo claves actualmente. [8]

Los casos de uso más destacados de la tecnología 5G [9], según las especificaciones del 3GPP son:

1. ***Enhanced Mobile Broadband* (eMBB)** o Banda Ancha Móvil Mejorada.
2. ***Ultra-Reliable and Low Latency Communications* (URLLC)** o Comunicación Ultra Confiable y de Baja Latencia.
3. ***Massive Machine Type Communications* (mMTC)** o Comunicación Masiva de Tipo Máquina.

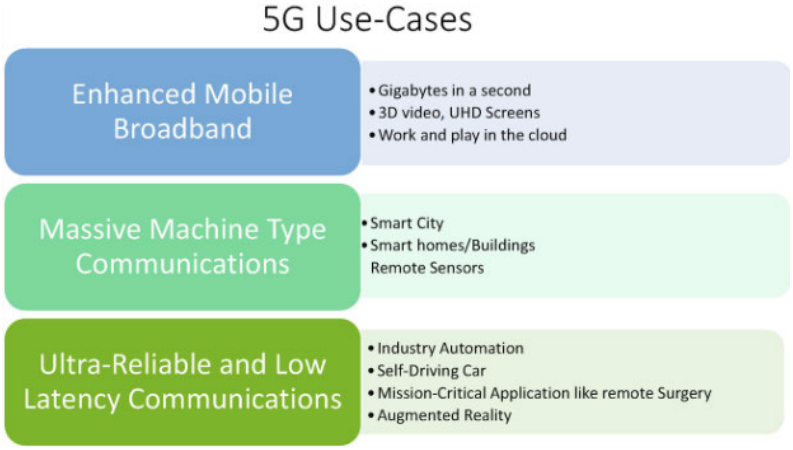


Figura 2.3. Casos de uso previstos por el 3GPP [9].

El núcleo de red y la red de acceso radio de cualquiera de las tecnologías de comunicaciones móviles están estandarizadas por la organización 3GPP. En el caso de la especificación de 5G, el 3GPP comenzó a trabajar en ella a mediados de 2017, partiendo de la base de las Release 13 y 14 para *Long Term Evolution for Machine Type Communication* (LTE-M) y *Narrowband - Internet of Things* (NB-IoT) respectivamente. Estas abordan algunos de los requisitos para las comunicaciones de tipo máquina o *Machine Type Communications* (MTC), tales como: bajo costo de los dispositivos, cobertura extensa y batería de larga duración.

A partir de entonces, el trabajo de estandarización del 5G se dividió en dos fases principales. [8]

La primera, incluyó la estandarización de los bloques fundamentales de 5G, y fue completada en marzo de 2019 (Release 15). Con el objetivo final de acelerar el cronograma de desarrollo de 5G, el 3GPP dividió la Release 15 en tres partes:

- En primer lugar, se estandarizó una arquitectura *New Radio Non Stand Alone* (NR NSA) o Nueva Radio No Independiente que combina dos tecnologías de radio: una nueva tecnología de interfaz radio, denominada *New Radio* (NR) o Nueva Radio, y *Long Term Evolution* (LTE).

- En segundo lugar, se abordó otro tipo de arquitectura radio completamente diferente, denominada *New Radio Stand Alone* (NR SA) o Nueva Radio Independiente, cuya principal diferencia con NR NSA, reside en que para su funcionamiento no es necesario el uso de LTE, mientras que para NR NSA es imprescindible el uso de LTE.
- Por último, el tercer paso para la Release 15 fue incluir más opciones de arquitectura (por ejemplo, la posibilidad de conectar gNBs (5G NodeBs) a un *Evolved Package Core* (EPC) y luego operar con NR y LTE de forma simultánea).

En la segunda fase del desarrollo, a los elementos fundamentales de la red 5G se le agregaron una serie de mejoras adicionales (Release 16). Como resultado, los primeros despliegues 5G finalmente se materializaron a principios de 2019 en Corea del Sur y Estados Unidos. Poco después se lanzaron al mercado los primeros dispositivos móviles compatibles.

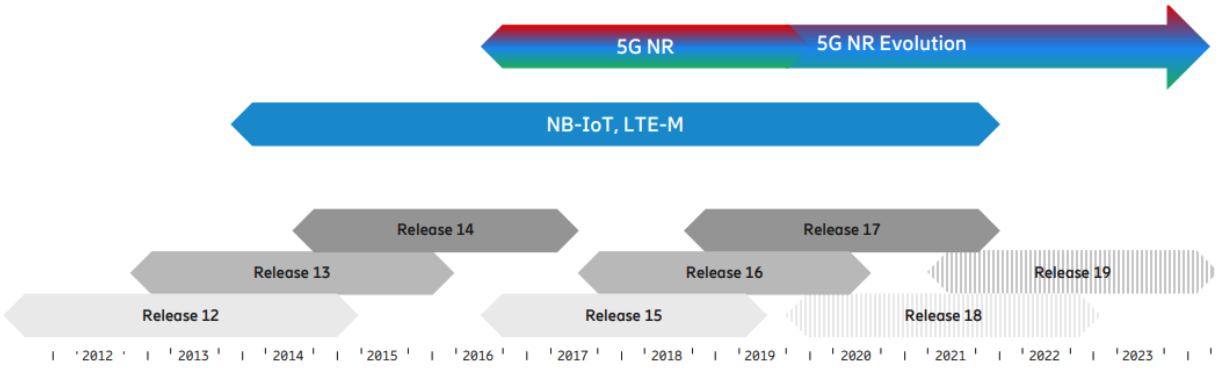


Figura 2.4. Desarrollo de las especificaciones por el 3GPP [8]

2.3.1. Arquitectura de red: Red de acceso radio

Radio Access Network (RAN) o red de acceso radio, se encarga de supervisar las funciones relacionadas con la parte radio de la red 5G, tales como, programación, gestión de recursos radio, protocolos de transmisión, codificación, etc. A fin de poder proporcionar el servicio final a los usuarios.

La implementación de los elementos de la red de acceso radio en las redes 5G son llevadas a cabo a través de *Network Function Virtualisation* (NFV) y *Software Defined Networking* (SDN), esto permite realizar despliegues más flexibles y eficientes que en generaciones anteriores. [10] [11] El 3GPP define varias arquitecturas para el despliegue de redes 5G. Estas arquitecturas se encuentran clasificadas en dos grandes grupos: en el primero de ellos encontramos las redes *5G Non Stand Alone* (5G NSA) y en el segundo las redes *5G Stand Alone* (5G SA). Las redes 5G NSA comparten el *Core Network* (CN) con las redes 4G, es decir, el EPC y los *eNodeB* (eNB), para la parte de acceso radio. Esto implica que ambas tecnologías 5G y 4G trabajan de forma conjunta tal y como podemos observar en la figura 2.5.

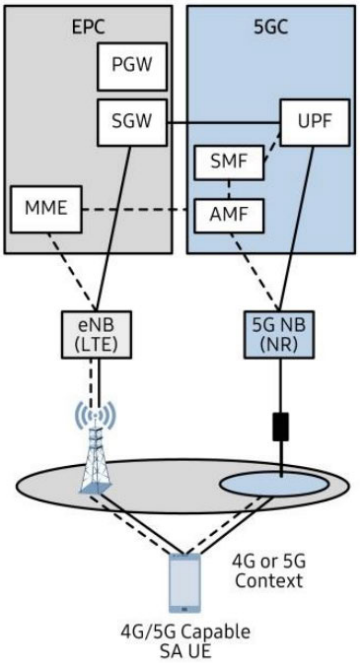


Figura 2.5. Diagrama de red de acceso radio para 5G NSA [12]

Por otra parte, las redes 5G SA son aquellas en las cuales todos los elementos de la red son 5G, es decir, solo prestan servicio a aquellos dispositivos que funcionan íntegramente con la tecnología 5G, sin coexistir con infraestructuras de tecnologías anteriores. Los componentes que conforman este tipo de red son: *5G Next-gen Core* (5G NGC) y *Next Generation Node B* (gNB) tal y como podemos observar en la figura 2.6.

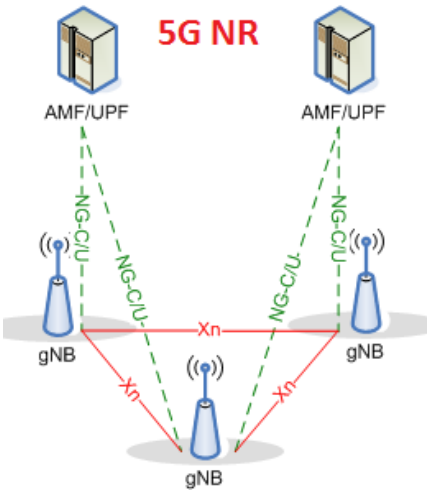


Figura 2.6. Diagrama de red de acceso radio para 5G SA [13]

2.3.2. Arquitectura de red: Núcleo de red

Como se ha mencionado anteriormente, la arquitectura de la red 5G está basada en SBA, incluyendo también al propio núcleo de la red (5G NGC). Por lo tanto, con este nuevo paradigma, en la arquitectura de núcleo de red 5G se refuerza la separación entre el plano de control y el plano de usuario con respecto a generaciones anteriores.

El núcleo de red 5G, se encarga de supervisar las funciones que, a pesar de no estar relacionadas directamente con la red de acceso radio, son necesarias para el funcionamiento de la red en su conjunto. Estas funciones van desde la autenticación de los usuarios y facturación, hasta la configuración de conexiones extremo a extremo.

El hecho de que el núcleo de red se base en una arquitectura SBA, implica que la configuración de red actual puede funcionar en máquinas de propósito general, es decir, ordenadores o servidores con *hardware* comercial. Por lo tanto, se elimina la necesidad de adquirir equipos específicos para cada funcionalidad que se desea añadir a la red. Gracias a esto es posible abaratar enormemente los costes del despliegue de la red 5G, además de proporcionar una amplia flexibilidad y escalabilidad a la infraestructura cuando se desean implementar nuevas funcionalidades y servicios finales. [14]

En la figura 2.7 podemos observar la composición de una red 5G SA, incluyéndose ambas partes: RAN y CN. A continuación, se procederá a la descripción de sus elementos principales [15] [16], así como su función dentro de la arquitectura.

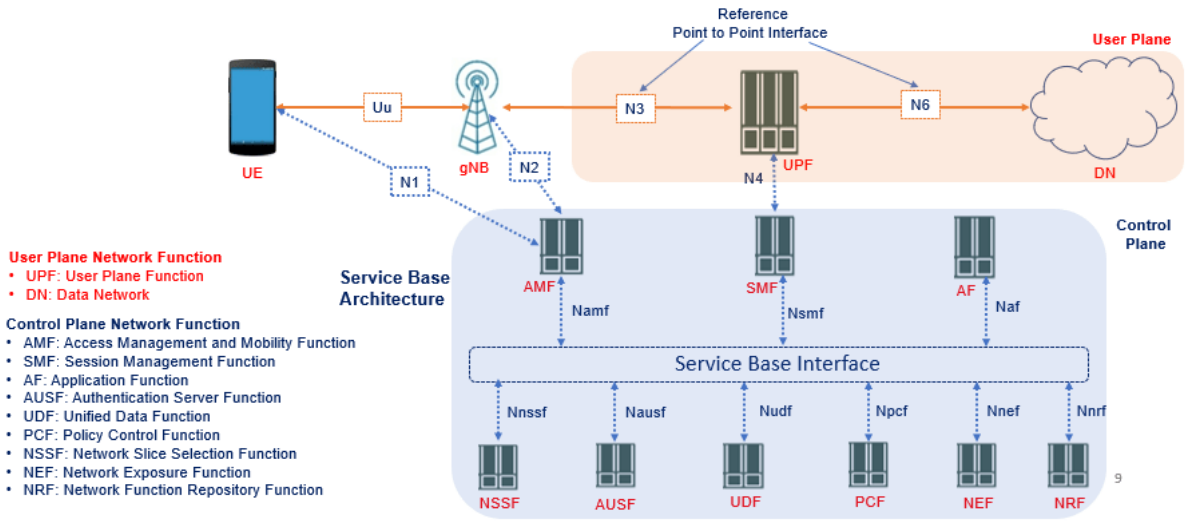


Figura 2.7. Diagrama de arquitectura de red 5G [17]

El plano de usuario está formado por:

- la función de plano de usuario o ***User Plane Function*** **(UPF)**,
- la puerta de enlace entre la red de acceso radio (RAN) y las redes externas como Internet.

Entre sus funciones se encuentran las de encaminamiento y reenvío de paquetes, las funciones de verificación de paquetes, la gestión de *Quality of Service* (QoS), el filtrado de paquetes y las mediciones de tráfico. Las operaciones y procedimientos que se realizan entre el terminal de usuario y la red de acceso radio se denominan *Access Stratum* (AS).

En el otro lado, tenemos un gran número de componentes que en su conjunto conforman el plano de control. A continuación, analizaremos los más relevantes para comprender el funcionamiento interno de la red:

- La función de gestión de sesión o ***Session Management Function*** **(SMF)**, se encarga de la administración de las sesiones de los usuarios. Esto incluye el establecimiento, la modificación y la liberación de las sesiones de los usuarios, además de la asignación de direcciones IP a los equipos de usuarios o *User Equipment* (UE). También se encarga de controlar el cumplimiento de las políticas.
- Por otra parte, encontramos la función de gestión de acceso y movilidad o ***Access and Mobility Management Function*** **(AMF)**, que se ocupa de: la señalización de control entre el núcleo de red y el terminal de usuario, la seguridad de los datos del usuario, la movilidad y la autenticación. Esta operativa entre el terminal de usuario y el núcleo de la red se denomina *Non Access Stratum* (NAS). Cabe señalar que el AMF no maneja por sí mismo algunas de las funciones descritas anteriormente, sino que se comunica con otros elementos (funciones) de la red (por ejemplo, para la autenticación) o directamente se encarga de reenviar los mensajes de señalización a los elementos de red correspondientes.

El núcleo de red también puede incorporar otras funcionalidades, algunas de las más importantes son:

- ***Policy Control Function*** **(PCF)** o Funciones de control de políticas, este elemento se encarga de las reglas y políticas.
- ***Unified Data Management*** **(UDM)** o Gestión Unificada de Datos, encargada de las credenciales (autenticación y autorización de acceso). El UDM representa la base de datos principal de subscriptores móviles. Se encarga de generar credenciales de autenticación para los dispositivos que desean acceder a la red. Además, permite aplicar cierta QoS al permitir emplear diferentes políticas sobre unos clientes u otros, en función de la información que se tenga almacenada en la base de datos.

- ***Authentication Server Function*** **(AUSF)** o Función de Servidor de autenticación, cuya función es gestionar la autenticación de los elementos conectados a la red. Proceso que emplea las credenciales generadas por el UDM.

- ***Application Function*** **(AF)** o Función de Aplicación, realiza operaciones como acceder a la NEF para recuperar recursos, interacciona con PCF para control de políticas, encaminamiento de tráfico de aplicaciones, exposición de servicios a usuarios finales, etc. Expone la capa de aplicación para interactuar con recursos de red 5G. En resumen, se utiliza para proveer de servicios a los usuarios de la red.

- ***Network Exposure Function*** **(NEF)** o Función de Exposición de Red, se encarga de exponer las capacidades y eventos de la red, tanto interna como externamente. Internamente se comunica con los demás componentes o funciones de red intercambiando información y/o eventos. Externamente aprovisiona de forma segura a las aplicaciones información procedente de los elementos de la red y viceversa.

- ***NR Repository Function*** **(NRF)** o Función de Repositorio de Red, desempeña la labor de proveer información acerca de las funciones de red disponibles. Recibe peticiones de descubrimiento de funciones de red y proporciona de la información necesaria para obtenerlas. Además, también se encarga del mantenimiento de estas funciones de red, es decir, si se añade, se actualiza, o se elimina una instancia de función de red, es trabajo del NRF mantener la información adecuada.

- ***Unified Data Repository*** **(UDR)** o Repositorio Unificado de Datos, almacena la información de las subscripciones, proporcionada por el UDM. Al mismo tiempo, también almacena información de forma estructurada para exponerla (cuando así se requiera) a otras funciones de la red.

- ***Network Slice Selection Function*** **(NSSF)** o Función de Selección de División de Red. Su función consiste en seleccionar la red que prestará servicio a un determinado UE. Además también determina el AMF, de entre todos los disponibles, que será usado por un UE en concreto.

Merece la pena señalar que las funciones más elementales de la red 5G pueden desplegarse bajo diversas configuraciones. Gran parte de las funciones pueden implementarse en un único nodo físico o distribuirse en varios nodos e, incluso, pueden ejecutarse en una plataforma en la nube.

2.3.3. Arquitectura de protocolos

La arquitectura de protocolos para 5G está claramente diferenciada en dos partes, estas son:

1. **Plano de control:** para el intercambio de información de señalización.
2. **Plano de usuario:** para intercambio de datos entre los UE y la red.

2.3.3.1. Plano de control

El plano de control se basa en el intercambio de información mediante NAS entre los terminales de usuario (UE) y el núcleo de red 5G (AMF y SMF), y en el intercambio de información a través de AS en la interfaz de acceso radio (gNB). La información AS consiste en información de señalización de parámetros de la interfaz radio. Por otro lado, la información NAS consiste en información de señalización de procedimientos de movilidad y sesiones de la red, es decir: autenticación, gestión de la movilidad, seguridad de las conexiones, etc. Además de los dos protocolos ya mencionados, en el plano de control se hace uso de otros protocolos: [18][19][20][21]

En la figura 2.8 se muestra la arquitectura de protocolos para el plano de control que se detallará de forma breve a continuación.

- Entre el **UE** y el **gNB:**

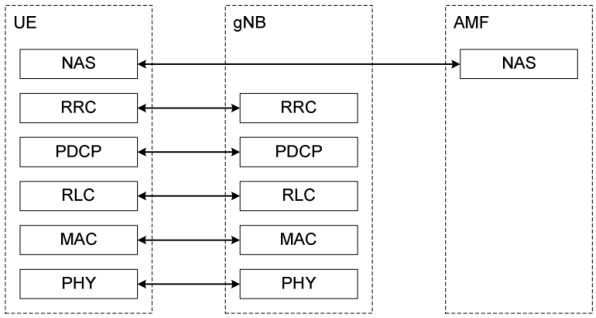


Figura 2.8. Arquitectura del plano de control para red 5G [19]

- ***Radio Resource Control* (RRC):** Encargado del control de la conexión, esto es: establecimiento, reconfiguración, liberación, control de radio y configuración de medidas.
- ***Packet Data Convergence Protocol* (PDCP):** Se encarga de la transferencia de datos, compresión de las cabeceras de información, entrega en secuencia y eliminación de duplicados. Además, se ocupa del cifrado en plano de control, así como de mantener la integridad.
- ***Radio Link Control* (RLC):** Tal y como su nombre indica, su función consiste en controlar el enlace radio: corrección de errores mediante *Automatic Repeat Request*

(ARQ), segmentación y reensamblado, detección de duplicados y números de secuencia independientes de PDCP.

- ***Medium Access Control* (MAC):** Se trata de un conjunto de mecanismos de comunicaciones, a través de los cuales varios "interlocutores" (teléfonos móviles, dispositivos de red, ordenadores, etc.) se ponen de acuerdo para compartir un medio de transmisión.
- **PHY:** Hace referencia a la parte física a través de la cual se transmiten los datos de las comunicaciones (ondas electromagnéticas, fibra óptica, par de cobre, etc.)

- Entre **gNB** y **AMF:**

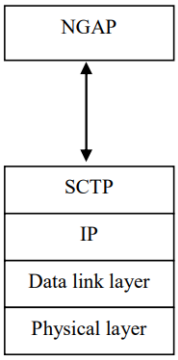


Figura 2.9. Arquitectura del plano de control entre gNB y AMF [22]

- ***NG Application Protocol* (NG-AP):** Consiste en un conjunto de *Elementary Procedure* (EP). Un EP consiste en una unidad de interacción entre un elemento en la red de acceso radio y el AMF. [21]
- ***Stream Control Transmission Protocol* (SCTP):** Se trata de un protocolo de red fiable. Es utilizado para reproducir la infraestructura de telefonía en redes IP. [23]

2.3.3.2. Plano de usuario

En la arquitectura de protocolos del plano de usuario podemos distinguir, por un lado, la arquitectura de protocolos entre UE y gNB y por otro, la arquitectura de protocolos entre gNB y el núcleo de la red, concretamente el UPF. [18] [19] [20] [21]

En las figuras 2.10 y 2.11, se muestra la arquitectura de protocolos para el plano de usuario cuyo detalle se explica a continuación.

- Entre el **UE** y el **gNB**:

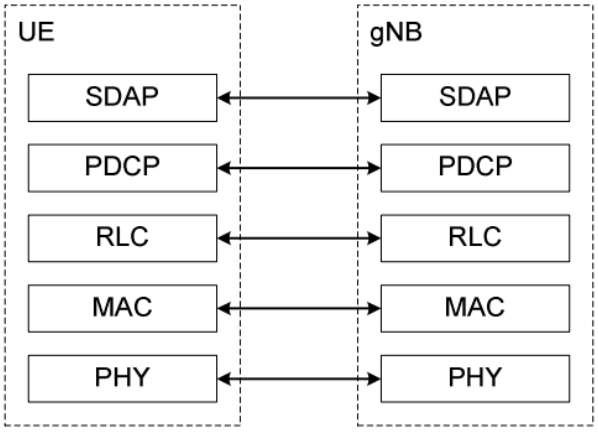


Figura 2.10. Arquitectura del plano de usuario para red 5G [19]

- ***Service Data Adaptation Protocol* (SDAP):** Permite la gestión de flujos QoS: [20].
  1. Mapea un flujo QoS a una sesión con un portador radio adecuado.
  2. Marca *QoS Flow ID* (QFI) de paquetes para asegurar el tratamiento adecuado en los diferentes elementos de la red 5G.
- ***Packet Data Convergence Protocol* (PDCP):** Se encarga de: la transferencia de datos, compresión de las cabeceras de información, entrega en secuencia y eliminación de duplicados. También se ocupa del cifrado del plano de control, así como de mantener la integridad.
- **RLC:** Tal y como su nombre indica, su función consiste en: controlar el enlace radio corrección de errores mediante *Automatic Repeat Request* (ARQ), segmentación y reensamblado, detección de duplicados y números de secuencia independientes de PDCP.
- **MAC:** Se trata de un conjunto de mecanismos de comunicaciones, a través de los cuales varios "interlocutores" (teléfonos móviles, dispositivos de red, ordenadores, etc.) se ponen de acuerdo para compartir un medio de transmisión.
- **PHY:** Hace referencia a la parte física a través de la cual se transmiten los datos de las comunicaciones (ondas electromagnéticas, fibra óptica, par de cobre, etc.)

- Entre el **gNB** y el **UPF**:

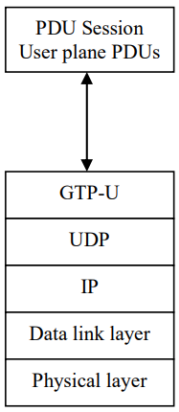


Figura 2.11. Arquitectura del plano de control entre gNB y UPF [21]

- ***GPRS Tunnelling Protocol* (GTP-U):** Se trata de un protocolo de comunicaciones basado en IP y que su utiliza para transportar GPRS. Concretamente, GTP-U es empleado para el transporte de datos de usuario dentro de la red central y entre la red de acceso radio y la red central. [24]

2.4. Virtualización de funciones de red (NFV)

A lo largo del desarrollo del proyecto aparecerán numerosos conceptos, no obstante, existe uno que resulta fundamental, se trata de NFV. [25] [26]

NFV es una arquitectura propuesta por ETSI [27] y aunque no propone ninguna implementación específica, se utiliza para virtualizar funciones de red. Estas se conciben como entidades *software* que se ejecutan en la *Network Function Virtualisation infraestructure* (NFVi).

Puede tratarse de funcionalidades de red, como *routers*, balanceadores de carga, *firewalls*, etc. Sin embargo, también es posible trasladar esta filosofía a las redes móviles, virtualizando funciones como la gestión de usuarios, gestión de la movilidad, gestión de políticas, calidad de servicio, etc.

Estas funcionalidades, desde un punto de vista de alto nivel, son empaquetadas como máquinas virtuales dentro de *hardware* de propósito general. Esto permite que los proveedores de servicios puedan ejecutar su infraestructura en servidores estándar, abandonando por completo el paradigma de generaciones anteriores, donde se hacía necesario emplear hardware propietario o específico para el despliegue de las funciones dentro de la infraestructura de red.

En la figura 2.12 se puede observar la arquitectura de NFV. A continuación, se presenta una descripción breve de los elementos que componen dicha arquitectura:

- ***Virtual Network Function* (VNF):** implementaciones software de las funciones que se ejecutan en la infraestructura NFV.
- ***Network Function Virtualisation infraestructure* (NFVi):** se trata de los elementos de la infraestructura (*hardware*, conexiones de red y otros recursos físicos) de cualquier plataforma que permita ejecutar *software* VNF. Particularmente y para el desarrollo de este proyecto se ha seleccionado la plataforma de gestión de contenedores *Docker*, analizada con mayor detalle en capítulos posteriores.
- **Gestión, automatización y organización de la red o *Management and Orchestation* (MANO):** proporciona el marco para gestionar la infraestructura de NFV e implementar nuevas VNFs. Como veremos más adelante, el ETSI no propone ninguna implementación específica, sino que proporciona una serie de buenas prácticas y define un modelo de arquitectura.

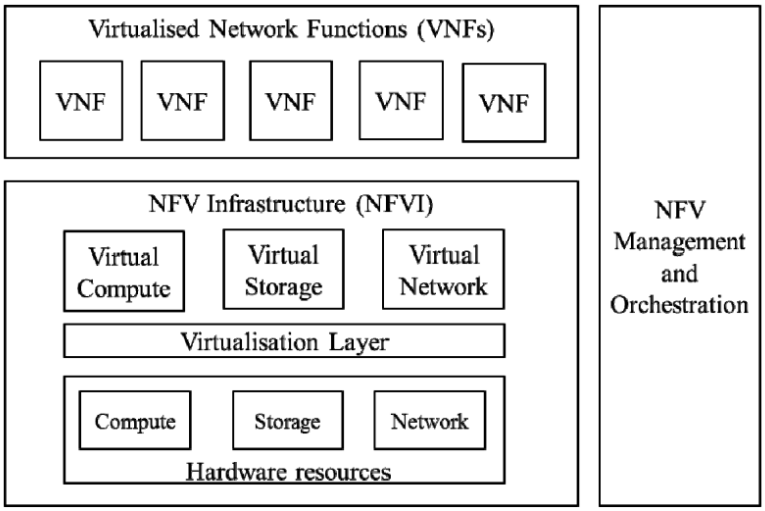


Figura 2.12. Esquema de alto nivel de arquitectura NFV [27]

En resumen, NFV ofrece a los proveedores la flexibilidad para ejecutar VNFs en servidores descentralizados o trasladarlas cuando sea necesario, mejorando en gran medida la operatividad y la eficiencia, reduciendo de forma drástica los costes para los proveedores de servicio.

Además, los servicios desplegados pueden ser fácilmente escalables de forma dinámica, es decir, se puede asignar una cantidad mayor o menor de recursos en función de las necesidades, sin tener que adquirir equipos nuevos o deshacerse de los ya existentes. Por otra parte, la introducción de NFV permite el acceso al mercado de las telecomunicaciones a nuevas empresas más pequeñas e incluso empresas especializadas en el desarrollo *software*. [28]

2.5. Voice over New Radio (VoNR)

Voice Over New Radio (VoNR) es la tecnología que hace posible encapsular toda la información de una llamada de voz sobre el protocolo *Voice over IP* (VoIP). Permite la transmisión de voz haciendo uso del núcleo de red (CN) y la red de acceso radio (RAN) de 5G.

Al igual que su predecesor (VoLTE), VoNR emplea el *IP Multimedia Subsystem* (IMS) para la gestión de llamadas. Así pues, el IMS se establece en las redes móviles como el marco arquitectónico que posibilita la prestación de servicios de comunicaciones multimedia. La principal diferencia entre VoLTE y VoNR, radica en la infraestructura de la red móvil subyacente y en la arquitectura de la red de acceso radio.

Son muchas las ventajas de VoNR, tales como mayor calidad de las llamadas, mayor fiabilidad, escalabilidad, reducción de costes, etc. Sin embrago, si observamos los últimos estudios llevados a cabo acerca de los despliegues de red 5G a lo largo del globo, los números parecen demostrar lo contrario. Tomando como referencia el último reporte de *Global mobile Suppliers Association* (GSA) de octubre de 2021, acerca del progreso a nivel mundial de *Voice Over New Radio* (VoNR), podemos ver que:

Hasta el momento se han catalogado 16 operadores que reconocen públicamente su inversión en VoNR. [29] [30] [31] De estos 16 operadores,

- 8 se encuentran en fase de pruebas, es decir, aún no tienen una infraestructura que de servicio a los usuarios,
- 3 de ellos están planteando implementar VoNR,
- otros 3 están ya implementando esta tecnología,
- uno ofrece un servicio de VoNR limitado como parte de una prueba de mercado y
- otro ha lanzado completamente servicios de VoNR.

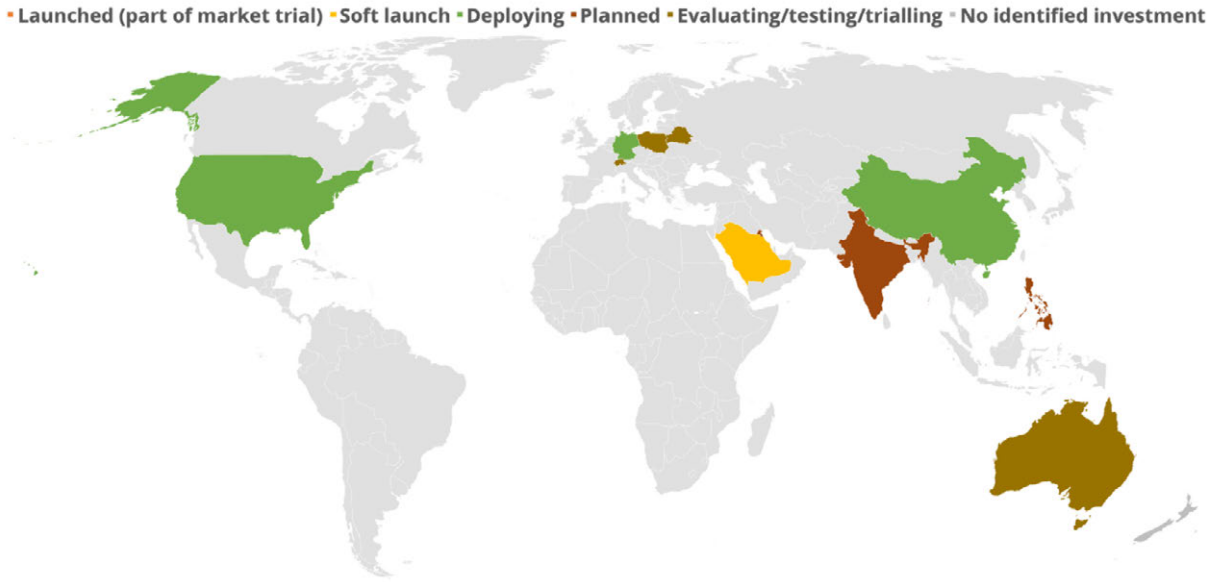


Figura 2.13. Despliegues de VoNR a nivel global en 2021 [31]

La realidad es que la mayor parte de las implementaciones actuales de redes 5G, se encuentran operando como redes NSA. Esto quiere decir que, tal como se ha visto en apartados anteriores, se emplean las infraestructuras de la red 4G para apoyar el desligue de 5G, no obstante, este tipo de despliegues no son apropiados para VoNR.

No hay duda de que se están llevando a cabo grandes esfuerzos por parte de las operadoras y otras compañías de telecomunicaciones para la implementación de redes 5G SA. Sin embargo, aún queda un largo camino ya que, según el último informe emitido por la GSA en enero de 2022, se reconoce que al menos 166 organizaciones utilizan redes 5G para proyectos piloto o para implementaciones completas. De entre estas, se sabe que solo 32 están investigando con redes 5G SA. [32]

Esta situación ha influido de forma directa en la realización de este proyecto, ya que a la hora de encontrar soluciones VoNR de *software* libre para su integración con los escenarios didácticos de red 5G SA propuestos, no se han hallado soluciones estables y válidas para tal propósito. Se está progresando de forma muy rápida, sin embargo, actualmente los esfuerzos no se encuentran focalizados en el desarrollo de soluciones VoNR, sino más bien en avanzar en las soluciones de núcleo de red y red de acceso radio 5G.

2.6. Soluciones software para la implementación propuesta

2.6.1. OPEN5GS

Open5GS es un *software* de código abierto que se distribuye bajo licencia *Affero General Public License* (AGPL) [33] y que implementa las especificaciones 5G del 3GPP. Concretamente nos proporciona una implementación del CN de 4G y 5G, poniendo a nuestra disposición un escenario 5G SA y otro 4G/5G NSA. Para ello emplea el lenguaje de programación C. [34] Es compatible con una amplia variedad de distribuciones *Linux*, como *Debian*, *Ubuntu*, *Fedora* y *CentOS*, así como con *FreeBSD* y *macOS*. Se trata de un software mantenido a través de empresas patrocinadoras: [35]

- Telet Research
- Wavemobile
- NextEPC
- Triple
- StrathSDR
- Skylark Wireless
- Sysmocom
- P1 Security
- ngvoice
- University of Bristol

Además, cuenta con una gran comunidad de usuarios que colaboran para resolver problemas y contribuir al desarrollo de *Open5GS*. Actualmente *Open5GS* cuenta con una implementación del core de red 5G muy completa y dispone de todos los elementos necesarios para realizar un despliegue de red 5G. Sin embargo, existen funcionalidades como VoNR que hoy en día continúan en fase de desarrollo. La plataforma *Open5GS* consta de dos escenarios diferentes de arquitectura del núcleo de red:

1. Open5GS 4G/5G NSA Core
2. Open5GS 5G SA

En la figura 2.14 se puede observar los dos tipos de arquitectura que puede soportar y todos los elementos que las componen.

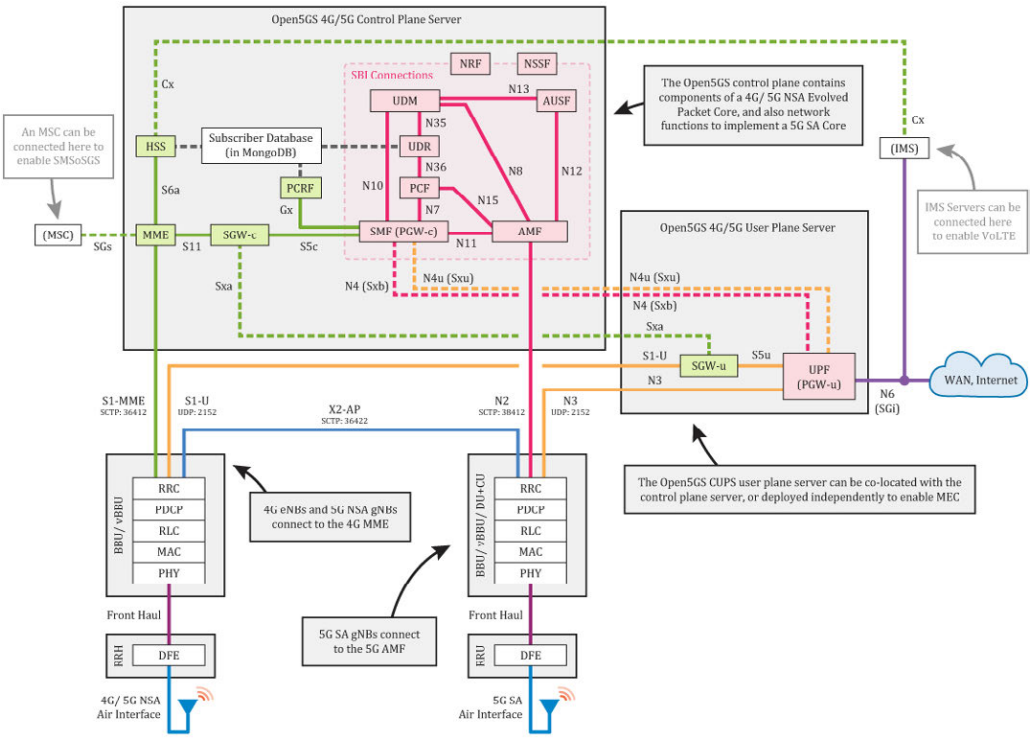


Figura 2.14. Open5GS core architecture [35]

Finalmente, uno de los puntos más desfavorables es la dificultad de adaptación para escenarios no contemplados por *Open5GS*. Por otra parte, soluciones homólogas de *Open5GS* proporcionan una documentación con un mayor nivel de detalle y pruebas para realizar, es decir, una documentación mucho más didáctica.

2.6.2. Free5GC

Free5GC es una solución *open source* para la implementación de las funciones del núcleo de red y de la red de acceso radio 5G, siguiendo los estándares de 3GPP. El *software* proporciona todos los elementos necesarios para el despliegue de una red 5G SA completamente funcional y, aunque no es objeto de estudio para este proyecto, cabe destacar que también posibilita el despliegue de una red 5G NSA.

Free5GC permite el estudio del comportamiento de los distintos elementos de la red 5G ya que proporciona herramientas para la realización de pruebas sobre estos. [36] [37] Además, *Free5GC* posibilita la integración con el software para la simulación de la red de acceso radio *UERANSIM*.

Sin embargo, *Free*5*GC* presenta varios puntos desfavorables con respecto a otras soluciones de características similares:

- La curva de aprendizaje de *Free*5*GC* es más pronunciada, debido a que presenta un mayor nivel de dificultad a la hora de realizar las configuraciones en los distintos componentes de la red 5G.
- De entre todas las soluciones analizadas, presenta un menor grado de madurez y desarrollo.
- Un nivel de documentación menor con respecto a otras soluciones.
- No presenta soluciones para el despliegue de servicios VoNR.

Así pues, queda expuesto que *Free*5*GC* no es una solución que ofrezca las características y el grado de madurez suficientes para postularse como una solución viable para el desarrollo del presente proyecto.

2.6.3. OpenAirInterface (OAI)

OpenAirInterface (OAI) [38] es una plataforma de código abierto que implementa las especificaciones 5G del 3GPP, mantenida por *OpenAirInterface Software Alliance* (OSA). Proporciona una implementación para la red de acceso radio RAN y otra diferente para el núcleo de la red CN:

- **cn5g** [39]: Contiene todos los ficheros necesarios para implementar las funcionalidades básicas de un núcleo de red 5G, para ello OAI ofrece dos posibilidades:
  1. Una primera implementación básica, que cuenta con los elementos indispensables para el CN: AMF, SMF, NRF y UPF
  2. La segunda opción utiliza tres funciones de red adicionales a las proporcionadas para el despliegue básico: UDM, AUSF y UDR
- **Openairinterface5G** [40]: Ofrece todo lo necesario para el despliegue de la red de acceso radio RAN para un despliegue 5G. Concretamente permite la simulación de un gNB en un escenario 5G SA y 5G NSA, además de ofrecer la posibilidad de emular un UE.

OAI permite desplegar escenarios modularizados para estudiar los distintos prototipos de red 5G. Gracias a que se trata de *software* libre, existe un gran número de desarrolladores, investigadores, aficionados, etc. La gran ventaja de esto es que la plataforma se mantiene actualizada y todos los errores son expuestos en la comunidad y subsanados en un espacio corto de tiempo, a la vez que se trabaja en la compatibilidad con otras tecnologías. Finalmente, uno de sus puntos más fuertes es la exhaustiva documentación que OAI ofrece a toda su comunidad, lo cual hace que sea una plataforma ideal para cualquier despliegue de red 5G SA en entornos docentes.

2.6.4. *UERANSIM*

UERANSIM consiste en una implementación de un terminal de usuario 5G (UE) y un gNB de código abierto. Se puede considerar como un teléfono móvil 5G y una estación base, simulados a través de software. Este proyecto resulta muy adecuado para su estudio y para la realización de pruebas sobre una núcleo de red 5G (CN), gracias al gran número de opciones que ofrece para trabajar el despliegue 5G. Al igual que pasaba con OAI, *UERANSIM* es un proyecto de código abierto, por lo que cuenta con una amplia comunidad que trabaja de forma continua en su desarrollo y compatibilidad con otros sistemas. Además, su instalación, adaptación al CN y puesta en marcha, resulta simple al contar con una documentación elaborada. [41] [42] [43]

2.6.5. gnbsim

Gnbsim es un simulador 5G SA que proporciona la capacidad de virtualizar múltiples UEs y gNBs para poder realizar pruebas experimentales en un sistema 5G. A pesar de que el proyecto original de *GitHub* no se encuentra disponible, OAI pone a nuestra disposición los recursos para recuperar su imagen *Docker* [44]. Aun haciendo uso de herramientas muy básicas [44], *gnbsim* permite la configuración de múltiples parámetros que posibilitan la realización de un gran número de experimentos sobre una red 5G. Estas características convierten a *gnbsim* en un buen candidato para la realización de este proyecto.

2.6.6. KAMAILIO

No existen en el mercado muchas soluciones *open source* para la implementación del IMS, además estas se ven aún más reducidas si se incluye el requisito de ser compatible con la integración de tecnologías de comunicaciones móviles (VoLTE o VoNR). Sin embargo, *Kamailio* se presenta como un servidor *Session Initiation Protocol* (SIP) programado en C y de código abierto, que es capaz de gestionar un gran número de configuraciones y llamadas. Es posible utilizarlo para desarrollar grandes plataformas para VoIP. *Kamailio* se estructura tal y como se puede observar en la figura 2.15. [45] [46]

Entre sus ventajas podemos destacar que *Kamailio*:

- Permite comunicaciones asíncronas, a través de *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP) o SCTP.
- Permite comunicaciones seguras, empleando para ello VoIP a través de *Transport Layer Security* (TLS).
- Contempla el desarrollo de una extensión IMS para VoNR.
- Proporciona un alto grado de escalabilidad.

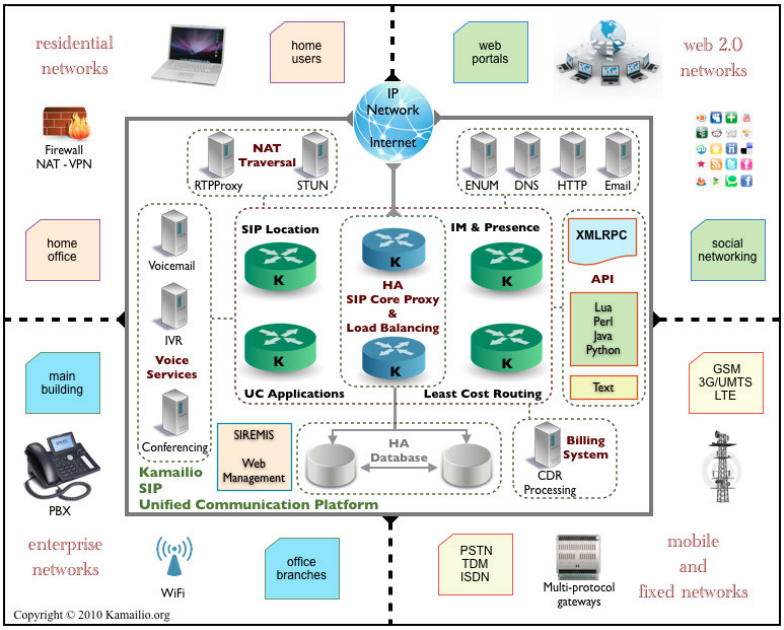


Figura 2.15. Esquema general, Kamailio [45]

2.6.7. Tecnología Docker para virtualización

El término *Docker* se aplica al proyecto *open source* para la organización y gestión de contenedores y las herramientas derivadas de este. Sin embargo, el término *Docker* puede resultar confuso en ocasiones, ya que también puede hacer referencia a *Docker Inc.*, la principal empresa promotora del proyecto. En nuestro caso, empelaremos Docker en referencia a la tecnología.[47]

Docker utiliza el kernel de Linux y sus funciones para dividir los procesos y poder ejecutarlos de forma independiente. El objetivo final de los contenedores es ejecutar varios procesos de forma separada para hacer un uso mucho más eficiente de los recursos disponibles de una máquina.

Las herramientas de contenedores como *Docker*, proporcionada por *Docker Inc.* proveen una arquitectura de implementación basada en imágenes y permite la automatización de la implementación de las aplicaciones (o conjunto de procesos que las constituyen) en un entorno de contenedores.

Como podemos ver en la imagen 2.16 podemos destacar de la arquitectura proporcionada por la tecnología *Docker*:

- Los contenedores incluyen la aplicación y todas sus dependencias.
- Comparten el *kernel* del sistema operativo con los demás contenedores.
- Cada contenedor se ejecuta como proceso aislado en el sistema operativo del *host*.
- Gestión compleja a la hora de escalar un sistema con muchos contenedores.

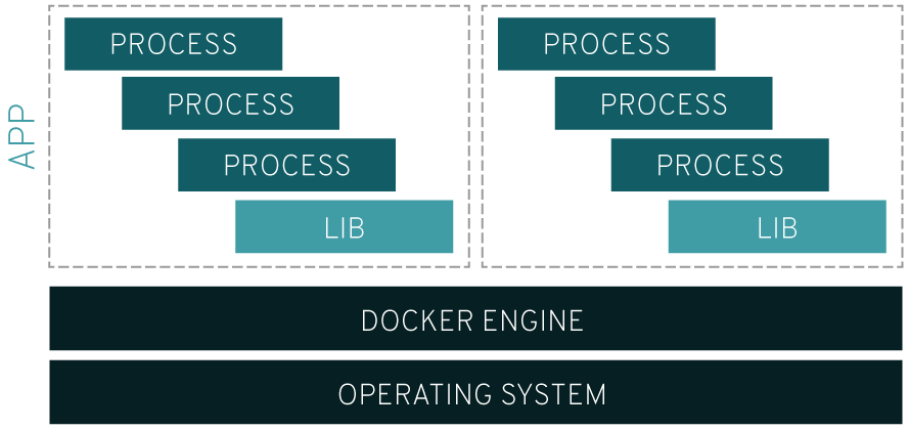


Figura 2.16. Esquema de funcionamiento de Docker [47] [48]

2.6.8. *Open Source MANO* (OSM)

Open Source MANO (OSM) es un marco de gestión y orquestación (MANO), de código abierto para la virtualización de funciones de red NFV. El proyecto ha sido desarrollado bajo la supervisión del ETSI, razón por la cual esta plataforma cumple rigurosamente con las especificaciones de esta organización. [28]

El concepto que propone ETSI acerca de los servicios de red se enfoca en relacionar componentes (VNF) entre sí dentro de entornos virtuales. Precisamente, OSM persigue definir los servicios de red *End to End* (E2E) independientemente de su naturaleza, es decir, sin tener en cuenta que los componentes involucrados sean físicos o virtuales. [49]

En cuanto a la distribución, los desarrolladores prefieren que OSM esté empaquetado en contenedores *software* tipo *Docker* para facilitar su distribución e instalación. El objetivo principal es hacer que este proceso sea lo más sencillo y liviano posible.

Finalmente, uno de los puntos más desfavorables de esta solución lo encontramos en los requisitos *hardware*, que son exigentes a la hora de implementar la solución dentro de máquinas virtuales. Concretamente, los recursos hardware recomendados para la máquina anfitriona de OSM son los siguientes:

	Cantidad mínima	Cantidad Recomendada
Número CPUs	2	2
RAM	6GB	8GB
Capacidad Disco	40GB	40GB

Tabla 2.1: Requisitos hardware para OSM [50]

2.7. Conclusiones

En este capítulo se ha hecho una introducción al marco tecnológico en el que se desarrolla el presente proyecto. En primer lugar, se ha desgranado la evolución de las comunicaciones móviles y sus principales características, a través de las distintas generaciones: 1G, 2G, 3G, 4G hasta llegar a la actualidad con 5G. A continuación, se ha introducido el 5G, que es la tecnología de comunicación móvil de vanguardia en nuestros tiempos.

Posteriormente se ha desarrollado en mayor profundidad la tecnología 5G, describiendo su arquitectura, tanto a nivel de radio como a nivel de núcleo de red y sus protocolos correspondientes. Además, se ha detallado el funcionamiento y el estado actual de VoNR. Adicionalmente se ha proporcionado una visión a alto nivel del funcionamiento de NFV.

También se han introducido las mejores soluciones *software* disponibles para la implementación de las arquitecturas propuestas en este proyecto. Para la parte de la simulación de acceso radio se han elegido las soluciones propuestas por $UERANSIM$ y *gnbsim* y para la parte del núcleo de red, se ha decidido emplear la solución proporcionada por OAI. Todo ello empleando una arquitectura basada en contenedores *Docker*, debido a su gran flexibilidad y modularidad.

El motivo principal por la cual se decide usar $UERANSIM$ para la parte de acceso radio (RAN) (que incluye al UE y al gNB) es que se trata de un *software* de código abierto, por lo que cuenta con una amplia comunidad. Además, presenta una excelente compatibilidad con el núcleo de red proporcionado por OAI y ofrece un extenso abanico de herramientas para estudiar y probar el núcleo de red, todo ello conforme con las especificaciones del 3GPP. Por otra parte, el motivo para la elección de *gnbsim*, a pesar de que cuenta con muchas menos herramientas que su homólogo $UERANSIM$, resulta muy útil como implementación inicial y primera toma de contacto con el entorno de desarrollo en este proyecto.

Las razones por las cuales se ha decidido emplear OAI para la implementación de los elementos del núcleo de red 5G son varias, entre ellas las más importantes son:

- Proporciona una solución completamente modular y, por lo tanto, una gran flexibilidad a la hora de seleccionar las funciones del núcleo de red que se desean desplegar/simular en nuestro proyecto.
- $Software$ con una gran comunidad que lo mantiene actualizado, estable y que ofrece un amplio soporte ante cualquier incidencia.
- Se trata de una solución acorde con las especificaciones del 3GPP, de código abierto y soportada por una amplia variedad de tecnologías que permiten junto con su modularidad, añadir un abanico muy amplio de funcionalidades.

Por tanto, por las características que se han expuesto anteriormente, se ha seleccionado la plataforma *OpenAirInterface* (OAI) junto con $UERANSIM$ y *gnbsim* para el desarrollo de este proyecto. Constituyen, por lo tanto, la mejor implementación para el despliegue de prototipos de red 5G SA en entornos docentes, ya que incluye características más que suficientes para su estudio en laboratorios.

Capítulo 3

Especificaciones y restricciones de diseño

En este capítulo se abordarán las especificaciones y restricciones para el diseño de los prototipos de red. En las figuras 3.1, 3.2 y 3.3 pueden verse las arquitecturas de estos escenarios, que se desarrollarán con mayor detalle en el capítulo 4:

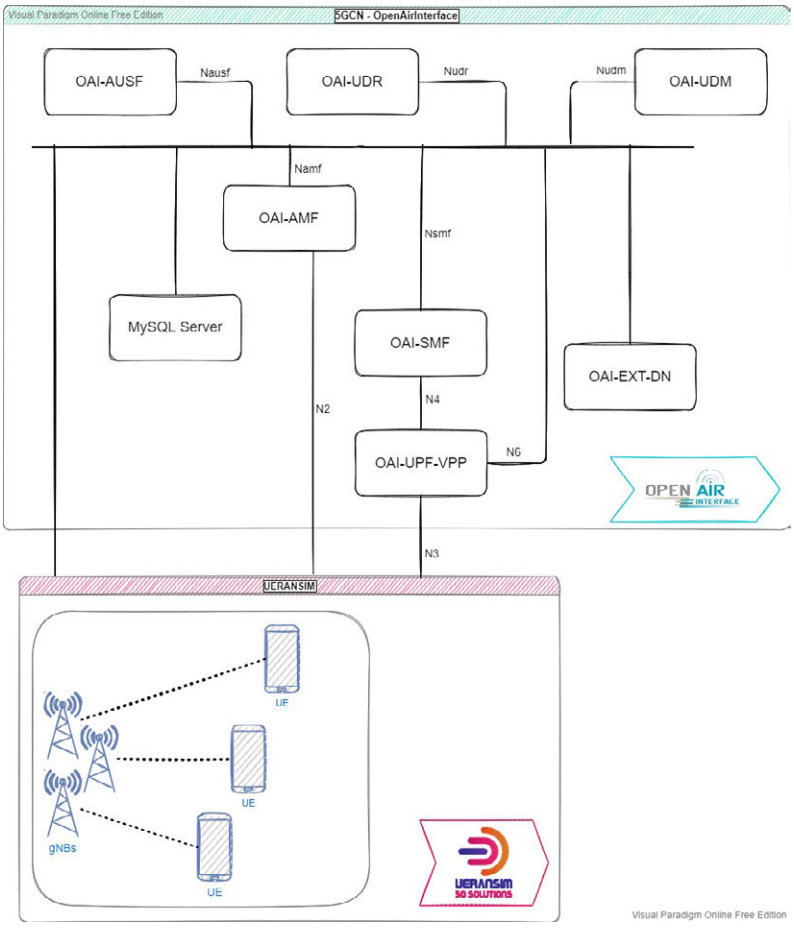


Figura 3.1. Arquitectura del prototipo de red. Escenario I, RAN implementado con *UERANSIM*. Elaboración propia, basada en [43][39][51]

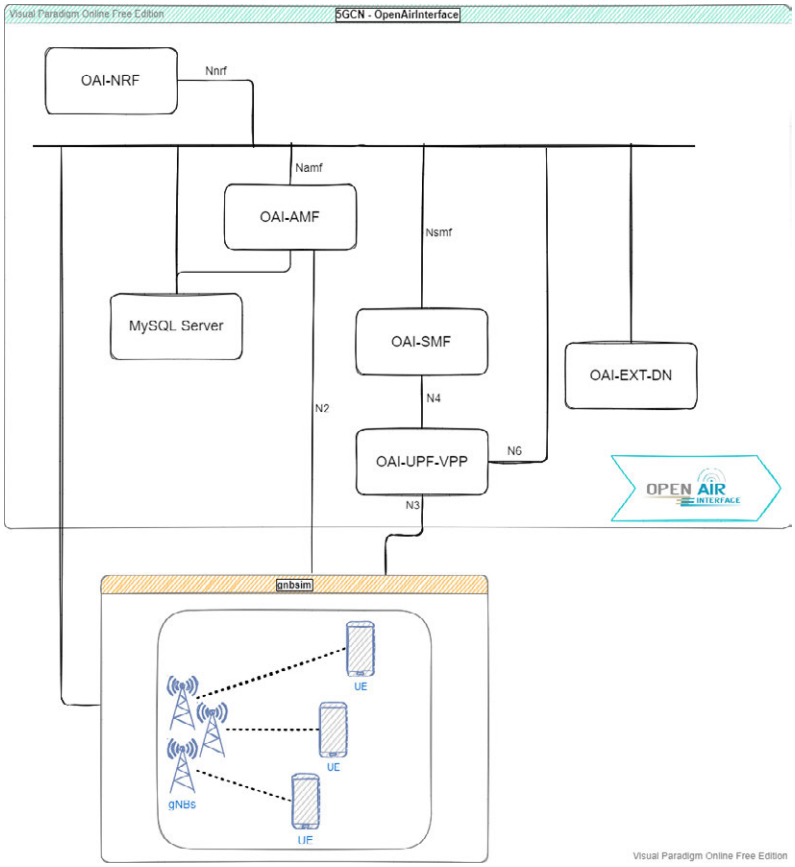


Figura 3.2. Arquitectura del prototipo de red. Escenario II, RAN implementado con gnbsim Elaboración propia, basada en [44][39][51]

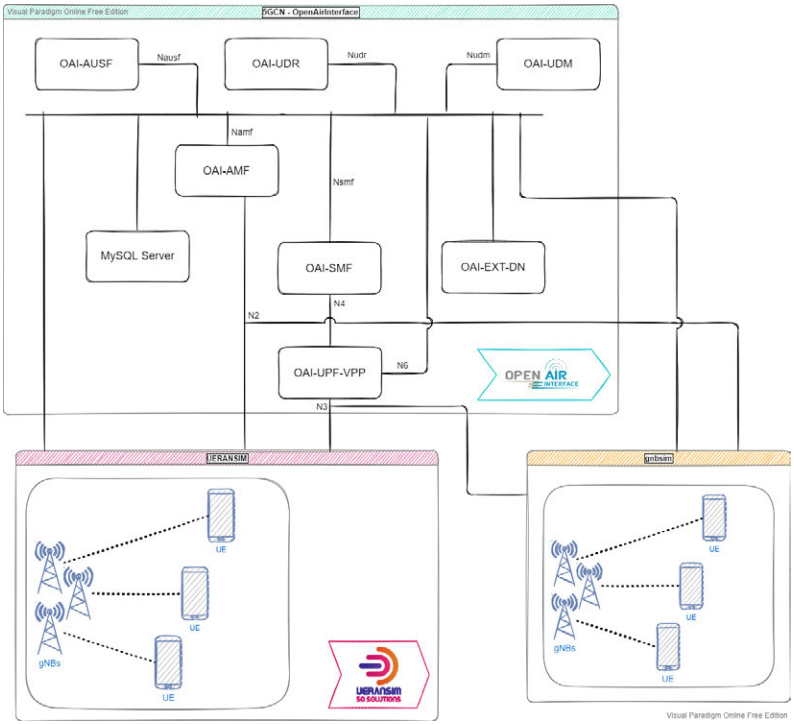


Figura 3.3. Arquitectura del prototipo de red. Escenario III, RAN implementado con UERANSIM y gnbsim Elaboración propia, basada en [44][39][51]

3.1. Especificaciones

- El prototipo de red estará dividido en dos partes claramente diferenciadas, de tal forma que su replicación sea lo más sencilla posible. Una parte se encargará del núcleo de la red 5G (5G NGC) y la otra parte será encargada de la red de acceso radio (gNB y UE)
- Se hará uso del *software* de código abierto *OpenAirInterface* (OAI). Por lo tanto, no es necesaria la adquisición de ningún tipo de licencia y, las capacidades podrán ser ampliadas en cualquier momento, si así se requiere.
- Para la implementación de la parte de acceso radio, se utilizará el *software* proporcionado por *UERANSIM* y *gnbsim*. Al tratarse de *software* de código abierto y licencia para uso no comercial, de nuevo, no será necesaria la adquisición de ninguna licencia. Para un despliegue de uso didáctico *UERANSIM* se postula como la mejor opción frente a *gnbsim*.
- Todo el despliegue se realizará mediante el uso máquinas virtuales, concretamente se hará uso del *software VirtualBox*.
- Se empleará la herramienta *Docker Compose* para el despliegue de todos los componentes del prototipo, tanto de la parte radio como de la parte correspondiente al núcleo de red.

- La máquina virtual que albergará el desarrollo del prototipo deberá ejecutar el sistema operativo de base *Linux Ubuntu Desktop*, concretamente deberá de ejecutar una versión igual o superior a 18.04.4 *Long Term Support* (LTS) de 64 bits.
- El prototipo de red deberá permitir configurarse con diferentes parámetros a fin de poder evaluar y estudiar los distintos escenarios y casos de uso.
- Se deberá permitir la captura de tráfico de red tanto en el plano de usuario (datos) como en el plano de control (señalización) para su posterior visualización y estudio a través de la herramienta $Wireshark$.
- Se deberá permitir evaluar los distintos procedimientos de red: registro, autenticación, desconexión, etc.
- Los terminales de usuario (UE) deberán poder tener acceso a internet.

3.2. Restricciones

El prototipo de red a desplegar cuenta con las siguientes restricciones de diseño:

- Será necesaria la instalación de *Docker* en la versión correspondiente para el sistema operativo de la máquina virtual que lo hospedará.
- Para el desarrollo del prototipo a través de *Docker Compose* será necesario crear una cuenta en $DockerHub$.
- No se podrán realizar llamadas de voz sobre IP VoNR debido a que el software de OAI no ofrece soporte para desplegar esa característica. Con carácter general las soluciones $software$ libres se encuentran aún en un estado muy prematuro en cuanto al soporte de servicios de VoNR.
- Las funcionalidades que ofrecen las herramientas $UERANSIM$ y *gnbsim* son limitadas, por lo que, si se pretende ampliar las capacidades de la red de acceso radio, previsiblemente será necesario buscar nuevas alternativas o desarrollar nuevos *scripts* que amplíen las funcionalidades de estas y cumplan con los estándares 3GPP y con la normativa ETSI.
- Tras la realización de numerosas pruebas, se ha determinado que $UERANSIM$ y $GNBSIM$ pueden integrarse en un mismo escenario, sin embargo, las funcionalidades que ofrecen individualmente no son compatibles entre sí.

Capítulo 4

Descripción de la solución propuesta

4.1. Descripción general

El objetivo de este proyecto consiste en desarrollar una red 5G SA acorde a las especificaciones 3GPP haciendo uso de entornos virtualizados. Concretamente se usarán dos implementaciones RAN diferentes que, junto con el CN proporcionado por OAI, darán lugar a tres escenarios didácticos diferentes:

1. Escenario I: OAI (CN) + *UERANSIM* (RAN). (Figura 4.1)
2. Escenario II: OAI (CN) + gnbsim (RAN). (Figura 4.2)
3. Escenario III: OAI (CN) + *UERANSIM*/gnbsim (RAN). (Figura 4.3)

4.2. Componentes elementales del desarrollo propuesto

En esta sección se identificarán y analizarán de forma breve los distintos elementos e interfaces que aparecerán a lo largo de los diferentes desarrollos de red 5G para entornos didácticos, descritos a continuación:

- *OpenAirInterface*. Se corresponde con las funciones del núcleo de red 5G. Este módulo se identifica concretamente como *cn5g* dentro de los repositorios de OAI. En el podemos encontrar:
  - *Authentication Server Function* (AUSF). Implementa la función de AUSF del núcleo de red 5G.
  - *Unified Data Repository* (UDR). Implementa la funcionalidad del UDR del núcleo de red 5G.

  - *Unified Data Management* (UDM). Se encarga de la función de UDM del núcleo de red 5G.
  - *Access and Mobility Management Function* (AMF). Su función es simular el AMF del núcleo de red 5G.
  - *Session Management Function* (SMF). Implementa la funcionalidad de un SMF del núcleo de red 5G.
  - *User Plane Function* (UPF). Actúa como UPF del núcleo de red 5G.
  - *NR Repository Function* (NRF). Realiza la función del NRF del núcleo de red 5G.

- *UERANSIM*. Se corresponde con la parte de acceso radio de la red 5G, además de incluir la capacidad de simular terminales de usuario (UE). Por lo tanto, este módulo incluye:
  - *Next Generation Node B* (gNB). Se trata de un módulo que implementa las funciones de un gNB propio de una red 5G.
  - *User Equipment* (UE). Se corresponde con el equipo de usuario (equivalentes a un terminal móvil comercial, con características 5G).

- *Gnbsim*. Se corresponde con la parte de acceso radio de la red 5G, además de incluir, al igual que *UERANSIM*, la capacidad para simular terminales de usuario (UE).
  - *Next Generation Node B* (gNB). Módulo que implementa las funciones de un gNB de la red 5G.
  - *User Equipment* (UE). Implementa el/los equipo/s de usuario.

En segundo lugar, podemos identificar una gran cantidad de interfaces, a través de las cuales los distintos componentes del núcleo de la red exponen sus servicios (SBA). [15]

- Interfaz **Nausf:** A través de ella expondrá sus servicios el AUSF.
- Interfaz **Nudr:** Por la cual el UDR expone sus servicios.
- Interfaz **Nudm:** Por la que el UDM expone sus servicios.
- Interfaz **Namf:** El AMF expondrá sus servicios a través de esta interfaz.
- Interfaz **Nsmf:** Se trata de la interfaz mediante la cual el SMF expone sus servicios.

Así mismo también encontramos otras interfaces como: [15]

- Interfaz **N2:** Se trata del punto de referencia entre RAN y el AMF.
- Interfaz **N3:** Su función es la de punto de referencia entre RAN y el UPF.
- Interfaz **N4:** Consiste en el punto de referencia entre SMF y el UPF.
- Interfaz **N6:** Hace la función de punto de referencia entre UPF y la red de datos externa.

4.3. Entornos de red 5G SA desarrollados

A continuación, se mostrarán en mayor profundidad las distintas implementaciones, para ello se presentarán los diagramas correspondientes junto con la configuración de red de cada escenario.

4.3.1. Escenario I

En esta sección analizaremos el primero de los despliegues desarrollados para este proyecto. Este escenario se compone, tal y como podemos observar en la figura 4.1, de todos los elementos esenciales para el núcleo de la red CN proporcionados por OAI junto con los elementos que componen la red de acceso radio (RAN), proporcionados por *UERANSIM*.

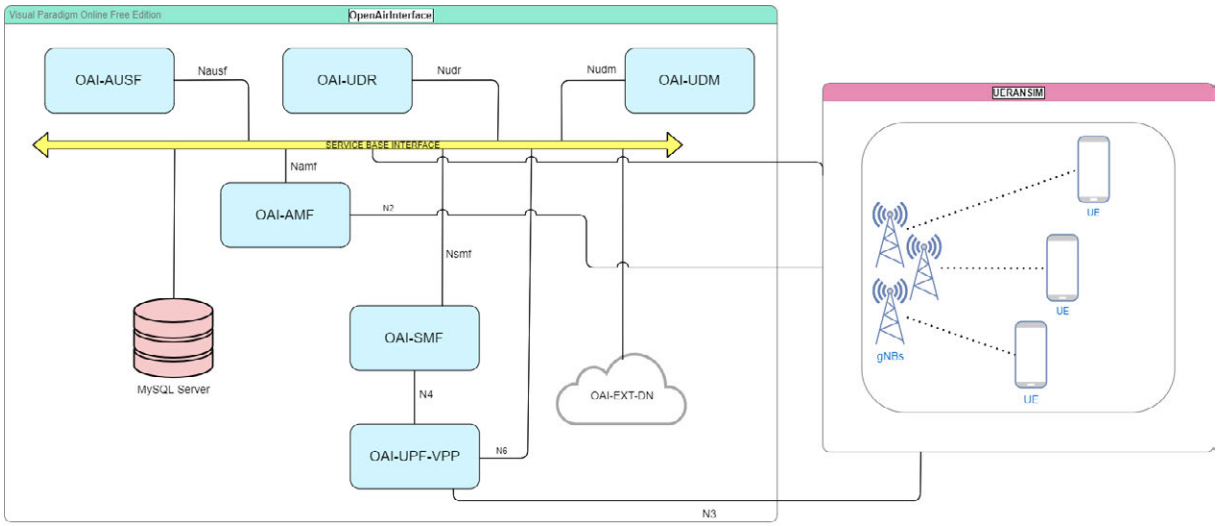


Figura 4.1. Arquitectura de red a desplegar. Elaboración propia, basada en [43][39][51]

En la tabla 4.1 se muestran las decisiones de diseño adoptadas con respecto al direccionamiento de los distintos elementos del escenario. Para este primer despliegue se ha decidido mantener la configuración propuesta por OAI para la integración de *UERANSIM* con su núcleo de red.

Segmento	Prefijo de red	Componente	Dirección IP
CORE	192.168.70.0/24	NRF	192.168.70.130
		MySQL	192.168.70.131
		AMF (N2)	192.168.70.132
		SMF (N4)	192.168.70.133
		UDR	192.168.70.136
		UDM	192.168.70.137
		AUSF	192.168.70.138
		UPF	192.168.70.202
RAN (N3)	192.168.72.0/24	gNB	192.168.72.141
SGi-LAN	192.168.73.0/24	N6	192.168.73.135

Tabla 4.1: Direccionamiento IP de los distintos componentes del escenario

4.3.2. Escenario II

En esta sección se analiza con más detalle el segundo de los despliegues llevados a cabo para el desarrollo de un entorno didáctico.

Este nuevo escenario, se compone del núcleo de red proporcionado por OAI, sin embargo, en esta ocasión se han eliminado los elementos: AUSF, UDR, UDM. Y en su lugar, ha sido añadido uno nuevo elemento, el NRF. Finalmente, en esta ocasión, se empleará la solución propuesta por *gnbsim* para la parte de acceso radio (RAN). Todo esto dará como resultado una arquitectura idéntica a la observada en la figura 4.2.

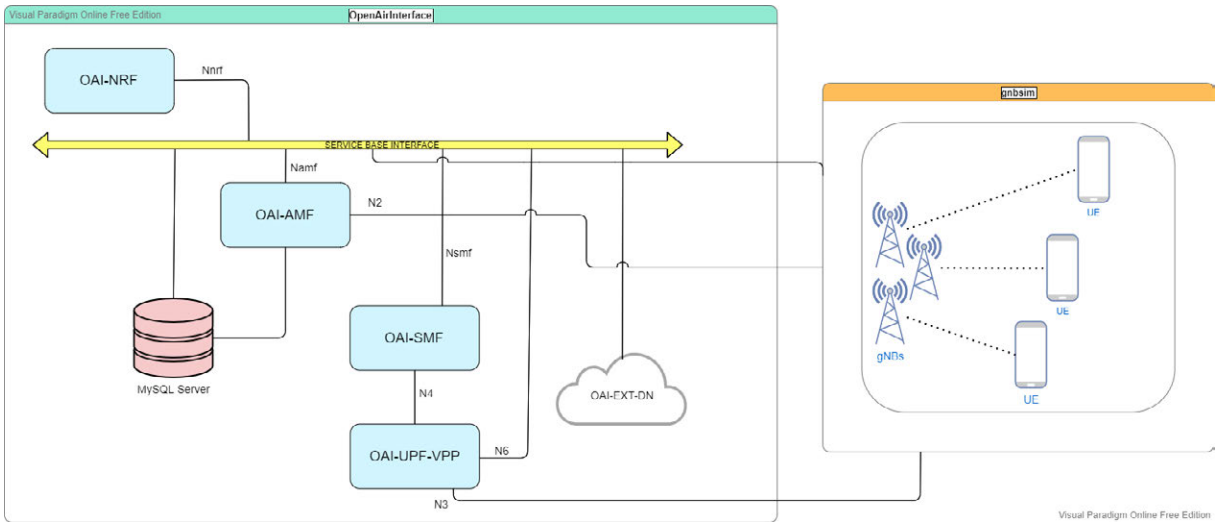


Figura 4.2. Arquitectura de red a desplegar. Elaboración propia, basada en [43][39][51]

En la tabla 4.2 es posible observar las decisiones de diseño adoptadas para la implementación de este nuevo escenario. De nuevo se ha decidido mantener la recomendación de configuración propuesta por OAI, para garantizar la compatibilidad y la buena integración de los elementos

del núcleo de red con la parte de acceso radio proporcionada por *gnbsim*.

Segmento	Prefijo de red	Componente	Dirección IP
CORE	192.168.70.0/24	NRF	192.168.70.130
		MySQL	192.168.70.131
		AMF (N2)	192.168.70.132
		SMF (N4)	192.168.70.133
		UPF	192.168.70.134
		N6	192.168.70.135
RAN (N3)	192.168.70.128/26	gNB	192.168.70.136
		N3 gNB1	192.168.70.156
		N3 gNB2	192.168.70.157
		N3 gNB3	192.168.70.158
		N3 gNB4	192.168.70.159
		N3 gNB5	192.168.70.160

Tabla 4.2: Direccionamiento IP de los distintos componentes del escenario II

4.3.3. Escenario III

Finalmente, para culminar el desarrollo de los prototipos de red para entornos didácticos, se ha decidido desarrollar una implementación que aúne las distintas soluciones para el acceso radio (RAN) presentadas en este proyecto, de tal forma que se logre una integración total de ambas, tal y como es posible observar en la figura 4.3.

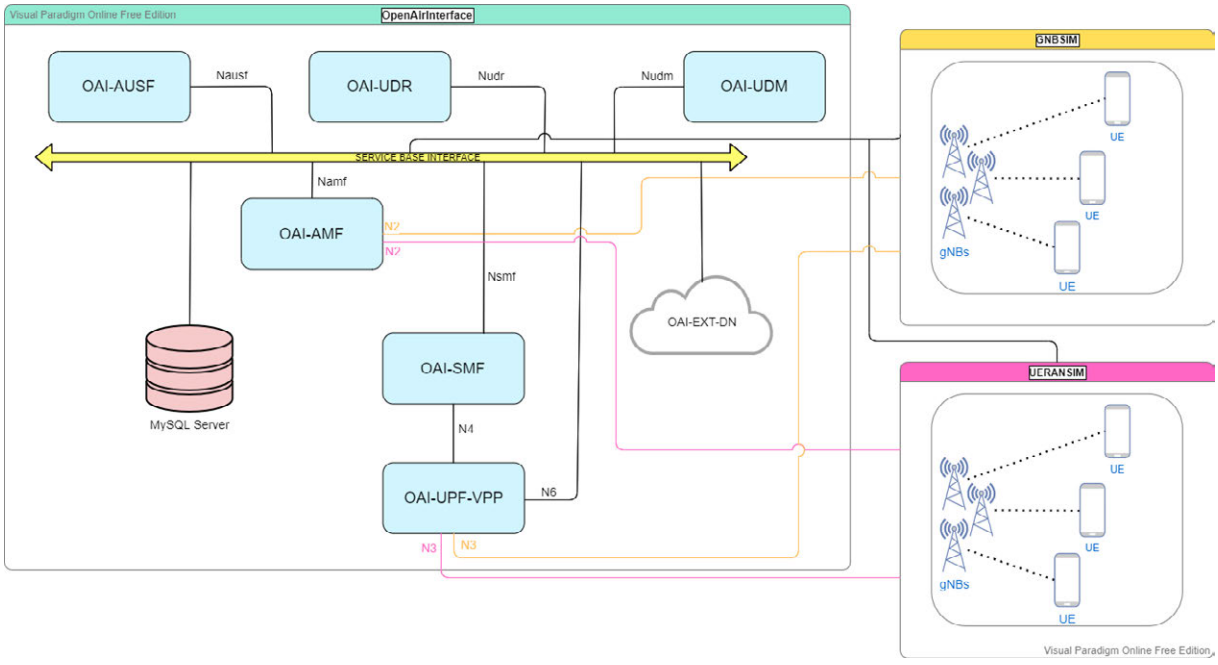


Figura 4.3. Arquitectura de red a desplegar. Elaboración propia, basada en [43][39][51]

En la tabla 4.3 es posible observar las decisiones de diseño adoptadas para la implementación de este último escenario. El direccionamiento propuesto garantiza la compatibilidad y la buena integración de los elementos del núcleo de red con la parte de acceso radio proporcionada por *gnbsim* y $UERANSIM$.

Segmento	Prefijo de red	Componente	Dirección IP
CORE	192.168.70.0/24	NRF	192.168.70.130
		MySQL	192.168.70.131
		AMF (N2)	192.168.70.132
		SMF (N4)	192.168.70.133
		UDR	192.168.70.136
		UDM	192.168.70.137
		AUSF	192.168.70.138
		UPF	192.168.70.202
RAN GNBSIM (N3)	192.168.70.128/26	gNB	192.168.70.139
		gNB1	192.168.70.140
		gNB2	192.168.70.141
		gNB3	192.168.70.142
		gNB4	192.168.70.143
		gNB5	192.168.70.144
RAN UERANSIM (N3)	192.168.72.0/24	gNB	192.168.72.141
SGi-LAN	192.168.73.0/24	N6	192.168.73.135

Tabla 4.3: Direccionamiento IP de los distintos componentes del escenario III

4.4. Estación de Trabajo

El despliegue del prototipo de red se llevará a cabo en un entorno de trabajo compuesto por una única estación portátil, en la cual a través de máquinas virtuales se construirán los escenarios de red propuestos. Las especificaciones *hardware* de la máquina física sobre la cual se desarrollará el trabajo, son las presentadas en la tabla 4.4.

Procesador	Memoria RAM	Capacidad de Almacenamiento	Otras características
Intel © Core™ i7-6700HQ	16GB DDR4	1TB HDD + 256GB SSD	Capacidades de la máquina aumentadas con respecto a capacidades de fábrica

Tabla 4.4: Hardware instalado en la estación de trabajo [52]

Como se ha mencionado anteriormente, la estación de trabajo ejecutará varias máquinas virtuales, una por cada escenario de red planteado. Cada máquina virtual será dotada de los siguientes recursos *hardware* detallados en la tabla 4.5.

Número de vCPUs	Cantidad de vRAM	Capacidad de vDisk
2	4GB	20GB

Tabla 4.5: Recursos asignados por máquina virtual

Estos recursos son suficientes para realizar todas las tareas sin ningún tipo de limitación. Así mismo, dentro de cada máquina virtual se ejecutarán en *Docker*, los contenedores de OAI junto con el contenedor correspondiente a la parte RAN determinada para cada escenario.

4.5. Preparación del entorno de desarrollo

Antes de realizar la instalación y configuración de los distintos componentes *software* tanto de OAI, como *gnbsim* y *UERANSIM*, es necesario realizar una serie de pasos previos para preparar la máquina en la cual se desarrollará el trabajo. Para un mayor detalle de las acciones a llevar a cabo y pasos precisos a seguir puede consultarse el ANEXO A de la memoria.

4.5.1. Configuración inicial de cada máquina virtual

En primer lugar, es necesario realizar la instalación del *software* de virtualización apropiado para albergar cada máquina virtual sobre la cual se desarrollarán los distintos escenarios de red 5G, en este caso emplearemos *Oracle VirtualBox* en su versión 6.1.34.

En segundo lugar, y una vez se ha instalado correctamente *VirtualBox* se instalará el sistema operativo sobre el cual trabajaremos. En esta ocasión, se ha decidido que el sistema operativo sea *Ubuntu* en su versión 20.04.3 LTS de 64 bits. En esta máquina se llevará acabo el despliegue y estudio de todos los escenarios de red 5G SA, por lo que la instalación del *kernel* debe ser la mínima, esto es, que solo los componentes básicos e imprescindibles para el buen funcionamiento de la máquina sean instalados. Esta decisión se toma en base a la optimización máxima del entorno, persiguiendo aprovechar la totalidad de las capacidades de la máquina anfitriona, para permitir que todos los escenarios de red se ejecuten con el mejor desempeño y posibilitando el uso de este desarrollo en máquinas con recursos *hardware* más limitados.

Una vez que se ha finalizado el proceso de instalación y configuración del *kernel*, se puede dar por concluida la configuración inicial necesaria en la máquina. A continuación, se realizará la configuración de los elementos *software* necesarios para establecer el entorno didáctico sobre el cual se desplegarán los distintos escenarios de red.

4.6. Instalación de *OpenAirInterface*

En esta sección se muestran las tareas necesarias para realizar la instalación de los módulos *software* de OAI que se van a utilizar para la implementación de los diferentes prototipos de red 5G SA. Previo al proceso de instalación, es necesario identificar cuáles son los módulos *software* necesarios para instalar OAI. En este caso, se necesita únicamente uno:

- **cn5g:** Contiene todos los ficheros para implementar los elementos básicos del núcleo de una red 5G.

4.6.1. Preparación para la instalación

En primer lugar, es necesario preparar la máquina para la instalación. Para ello, se buscarán y actualizarán todas las dependencias y paquetes que se encuentren disponibles para el sistema operativo. Una vez actualizado, es necesario instalar una serie de componentes *software*, previamente a la instalación de los elementos proporcionados por OAI.

4.6.1.1. Instalación de Docker y Docker-compose

Puesto que todos los elementos del despliegue de red se van a desarrollar utilizando tecnología basada en contenedores, es necesario instalar la versión del gestor de contenedores *Docker*, más adecuada al sistema operativo de la máquina virtual. En este caso la versión óptima es la 5:19.03.9 ubuntu-focal.

Después de haber completado las tareas de instalación de *Docker*, debemos añadir la herramienta *Docker-compose*, que resulta imprescindible para la ejecución de varias aplicaciones simultáneas basadas en *Docker*. A continuación, procedemos con el registro en la plataforma *Docker*, para lo que debemos crear una cuenta, a fin de realizar operaciones *Pull* de pasos posteriores. El detalle de las tareas de selección e instalación de *Docker*, así como del proceso de registro en la plataforma, puede consultarse en el ANEXO A sección A.1.0.1.

Finalizadas satisfactoriamente las operaciones de instalación y registro de *Docker*, es necesario tener en la máquina instalado el siguiente *software* (ver ANEXO A sección A.1.1):

- *Python*.
- *Wireshark*.

En el caso de *Python*, OAI recomienda instalar una versión igual o superior a la 3.6, para este proyecto se hará uso del paquete de *Python* en su última versión, es decir la 3.9. [53] Por otra parte, para la instalación del software *Wireshark* no existe ningún requerimiento de versión

específica, por lo que se instalará la última versión estable disponible, que en el momento de desarrollo de este proyecto es la 3.6.5.

Como última acción a realizar en este apartado y siguiendo las recomendaciones que se encuentran en los portales web, tanto de OAI como de *Docker*, se debe habilitar el reenvío desde contenedores *Docker* a redes externas (ver ANEXO A sección A.1.0.2). [53] [54]

4.6.1.2. Operaciones "*Pull*" sobre las imágenes base

En este apartado se indican las imágenes que son necesarias para el desarrollo del proyecto y que deberemos obtener a través de *Docker*.

- Imágenes básicas.
  - Ubuntu:focal
  - mysql:latest
- Componentes del núcleo de red proporcionados por OAI: Estas imágenes se encuentran alojadas en la cuenta de "rdefosseoai" (debe tenerse presente que puede cambiar en el futuro). [55]
  - oai-amf:latest
  - oai-nrf:latest
  - oai-spgwu-tiny:latest
  - oai-smf:latest
  - oai-udr:latest
  - oai-udm:latest
  - oai-ausf:latest
  - oai-upf-vpp:latest
  - oai-nssf:latest
  - trf-gen-cn5g:latest

Para obtener más detalles acerca de este proceso, puede consultarse el A sección A.1.2.

4.6.1.3. Configuración y despliegue de los contenedores

Después de finalizar el proceso de obtención de las imágenes de los elementos del núcleo de red 5G que proporciona OAI, el siguiente paso consiste en comprobar que los ficheros de configuración de cada elemento tienen los parámetros adecuados para los escenarios previstos.

El detalle completo de los ficheros de configuración de los componentes usados durante el desarrollo de este proyecto se encuentra disponible en el ANEXO D.

A continuación, desde OAI se ofrecen dos posibilidades para desplegar los contenedores o imágenes obtenidas de los elementos del núcleo de red:

1. Despliegue mini
   - Si se requiere NRF, el orden de despliegue será:
     a) MySQL
     b) OAI-NRF
     c) OAI-AMF
     d) OAI-SMF
     e) OAI-UPF
   - Si no es necesario NRF
     a) MySQL
     b) OAI-AMF
     c) OAI-SMF
     d) OAI-UPF
2. Despliegue básico
   - Si se requiere NRF, el orden de despliegue será:
     a) MySQL
     b) OAI-NRF
     c) OAI-AMF
     d) OAI-SMF
     e) OAI-UPF
   - Si no es necesario NRF
     a) MySQL
     b) OAI-AMF
     c) OAI-SMF
     d) OAI-UPF

En base al escenario que se esté implementando y a los requerimientos de este, se hace necesario seguir una secuencia de despliegue u otra. Sin embargo, para todas las implementaciones se debe de seguir una serie de pasos previos, cuyo detalle se encuentra recogido a lo largo del B.

Resulta imprescindible su lectura y la realización de los pasos que allí se recogen, antes de avanzar con las siguientes secciones.

4.6.2. Despliegue del escenario I: CN OAI + RAN *UERANSIM*

Una vez se han completado todos los pasos previos de instalación y preparación (ver Anexos A y B) de la propuesta integrada con *UERANSIM*, se puede continuar con la ejecución de los distintos componentes del núcleo de red, así como de la propia solución *software* de *UERANSIM*. Para ello será necesario abrir una consola de mandatos en el directorio "./oai-cn5g-fed/docker-compose" y ejecutar los siguientes comandos:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-basic-vpp-nrf.yaml up -d
```

Tabla 4.6: Comando para el despliegue del núcleo de red 5G proporcionado por OAI

Si se desea arrancar los componentes del núcleo de red y capturar su tráfico, deberá de ejecutarse la siguiente instrucción:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-basic-vpp-nrf.yaml up -d --capture
/tmp/oai/docker-compose-basic-vpp-nrf/docker-compose-basic-vpp-nrf.pcap
```

Tabla 4.7: Comando para el despliegue del núcleo de red 5G proporcionado por OAI con captura inicial

Finalmente se procede con la puesta en marcha del contenedor *Docker* de *UERANSIM*, para ello se debe de ejecutar la siguiente sentencia:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-ueransim-vpp.yaml up -d
```

Tabla 4.8: Sentencia para iniciar el contenedor *Docker* de *UERANSIM*

Completadas estas acciones, se concluye la parte del despliegue para el escenario para *UERANSIM*. En el capítulo 5 se profundizará con mucho más detalle en el funcionamiento de los distintos elementos.

4.6.3. Despliegue del escenario II: CN OAI + RAN Gnbsim

Después de haber completado todos los pasos previos de instalación y preparación (ver Anexos A y B) de la propuesta integrada con *gnbsim*, se puede continuar con la ejecución los distintos componentes del núcleo de red, así como del propio *software* de *gnbsim*. Para ello será necesario abrir una consola de mandatos en el directorio "./oai-cn5g-fed/docker-compose" y ejecutar los siguientes comandos:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# python3
./core-network.py --type start-basic --fqdn no --scenario 1
```

Tabla 4.9: Comando para el despliegue del núcleo de red 5G proporcionado por OAI

Si se desea arrancar los componentes del núcleo de red y capturar su tráfico, se deberá de ejecutar la siguiente instrucción:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# python3
./core-network.py --type start-basic --fqdn no --scenario 1 --capture
/tmp/oai/Tabla-gnbsim/Tabla-gnbsim.pcap
```

Tabla 4.10: Comando para el despliegue del núcleo de red 5G proporcionado por OAI con captura inicial

Finalmente se procede con la puesta en marcha del contenedor *Docker* de *gnbsim*, para ello se debe de ejecutar la siguiente sentencia:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-gnbsim.yaml up -d gnbsim
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# sleep 40
```

Tabla 4.11: Sentencia para iniciar el contenedor *Docker* de *gnbsim*

Completadas estas acciones, concluye la parte de despliegue para el escenario para *gnbsim*. En el capítulo 5 se profundizará con más detalle en el funcionamiento de los distintos elementos a través de capturas de tráfico de red y de visualización de ficheros *.log*.

4.6.4. Despliegue del escenario III: CN OAI + RAN *UERANSIM* y *gnbsim*

Tras haber completado todos los pasos previos para el desarrollo de las propuestas integradas con $UERANSIM$ y *gnbsim*, es posible implementar el escenario final. Para comenzar, se deben lanzar los distintos componentes del núcleo de red, así como de los propios $software$ de *gnbsim* y $UERANSIM$, tal como hemos visto en secciones anteriores. Para ello será necesario abrir una consola de mandatos en el directorio "./oai-cn5g-fed/docker-compose" y ejecutar los siguientes comandos:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-basic-vpp-nrf.yaml up -d
```

Tabla 4.12: Comando para el despliegue del núcleo de red 5G proporcionado por OAI

Finalmente se procede con la puesta en marcha de los contenedores *Docker* de *gnbsim* y $UERANSIM$ (en ese orden), para ello se deben de ejecutar las siguientes sentencias:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-gnbsim.yaml up -d gnbsim
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# sleep 40
```

Tabla 4.13: Sentencia para iniciar el contenedor *Docker* de *gnbsim*

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker-compose -f
docker-compose-ueransim-vpp.yaml up -d
```

Tabla 4.14: Sentencia para iniciar el contenedor *Docker* de *UERANSIM*

Completadas estas acciones, concluye la parte de despliegue para el escenario final que integra las redes de acceso radio tanto de *gnbsim* como de $UERANSIM$. En el capítulo 5 se profundizará con más detalle en el funcionamiento de los distintos elementos a través de capturas de tráfico de red y de visualización de ficheros *.log*.

4.7. Integración de VoNR sobre la red 5G SA

Tal y como se mencionó en el capítulo 3, sección 3.2, la implementación de los servicios de VoNR sobre una red virtualizada se encuentra en una fase muy prematura de desarrollo, ya que la escasez de soluciones de código abierto, sumada a las complicaciones de compatibilidad entre distintos desarrolladores y plataformas, hace que sea inviable el desarrollo de una red 5G SA con servicios de VoNR en entornos virtualizados. Sin embargo, a pesar de las limitaciones actuales, en esta sección se describen las posibles formas de implementar una solución VoNR sobre una red 5G SA virtualizada, así como la forma en que se integran los diferentes elementos vistos hasta el momento en esta memoria.

4.7.1. Implementación de UDM con funcionalidades *Home Subscriber Server* (HSS) para IMS

Esta implementación añade elementos con una doble funcionalidad para el despliegue de los servicios ofrecidos por 5G, entre ellos VoNR. La solución consiste en implementar un UDM que sea capaz de realizar funciones de HSS, junto con un PCF que también pueda realizar operaciones de *Policy and Charging Rules Function* (PCRF).

La principal ventaja de este método de implementación la encontramos en que no es necesario adaptar ningún elemento de la red IMS, pudiéndose integrar con una red 5G de forma rápida y sencilla. Para poder llevar esto a cabo es necesario contar con un elemento que sea capaz de encaminar paquetes del protocolo *diameter* (*Diameter Routing Agent* (DRA)).

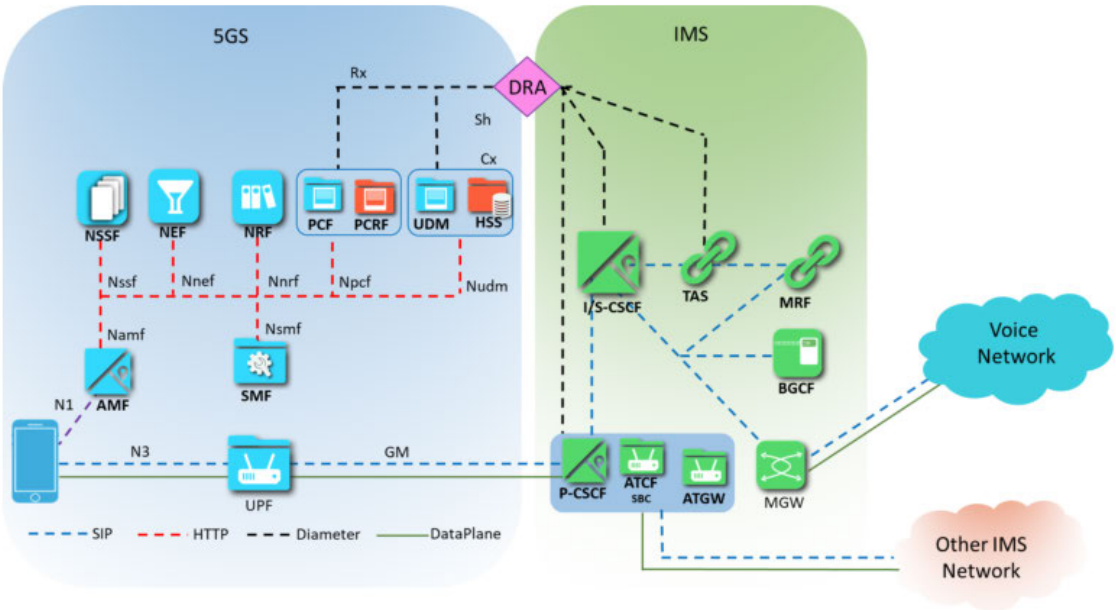


Figura 4.4. Arquitectura de implementación de UDM con funcionalidades HSS para IMS [56]

4.7.2. Implementación de un *Interworking Function* (IWF) para la traducción de paquetes *diameter* a *Hypertext Transfer Protocol Secure* (HTTP), integrando el IMS con el 5G CN

En este caso, la solución propuesta, pasa por la implementación de un nuevo elemento, el IWF, situado entre el núcleo de red 5G y el IMS. Su función consiste en "traducir" los mensajes o paquetes en protocolo *diameter* procedentes del IMS a paquetes HTTP que serán enviados al núcleo de red 5G.

La principal ventaja de este modelo reside en que no es necesaria la modificación de la arquitectura del IMS ni la del núcleo de red 5G. Sin embargo, será necesario configurarlos para el correcto funcionamiento con el nuevo elemento introducido (IWF).

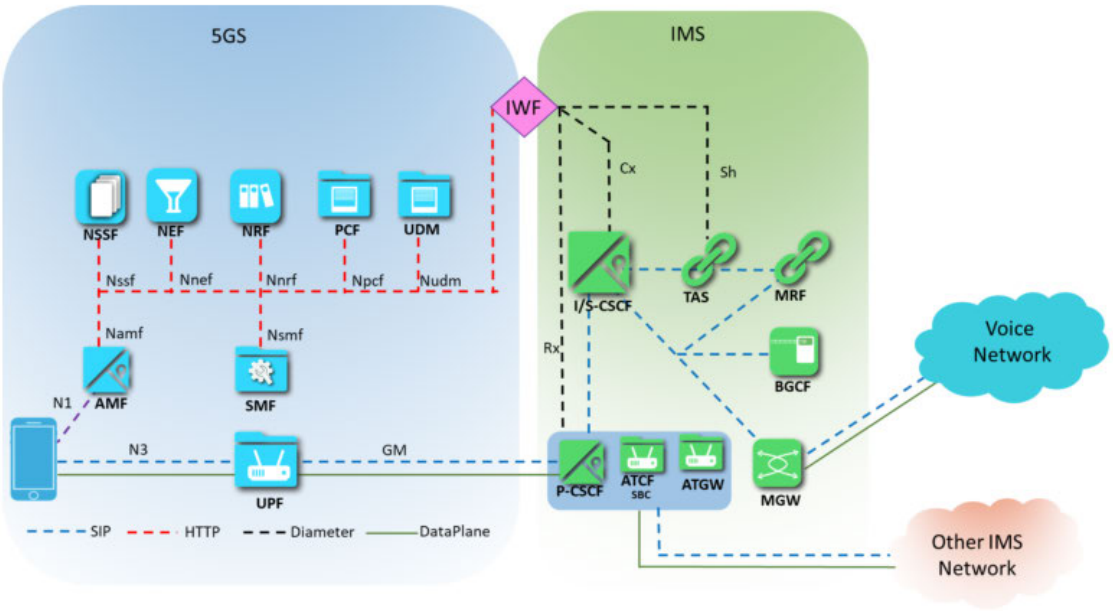


Figura 4.5. Implementación de un *Interworking Function* (IWF) para la traducción de paquetes *diameter* a HTTP, integrando el IMS con el 5G CN [56]

4.7.3. Adaptación de la red IMS para su compatibilidad con HTTP y los elementos del núcleo de red 5G

Esta última solución propuesta se basa en la actualización de los componentes de la red IMS para funcionar bajo el protocolo HTTP y, de esta forma, ser compatible con los elementos del núcleo de red 5G.

Este método resulta especialmente interesante dada su sencillez cuando se pretende construir una red 5G propia, partiendo de cero. Sin embargo, el principal punto débil es encontrar una solución IMS que permita realizar las configuraciones necesarias para conseguir la adaptación completa con el 5G CN.

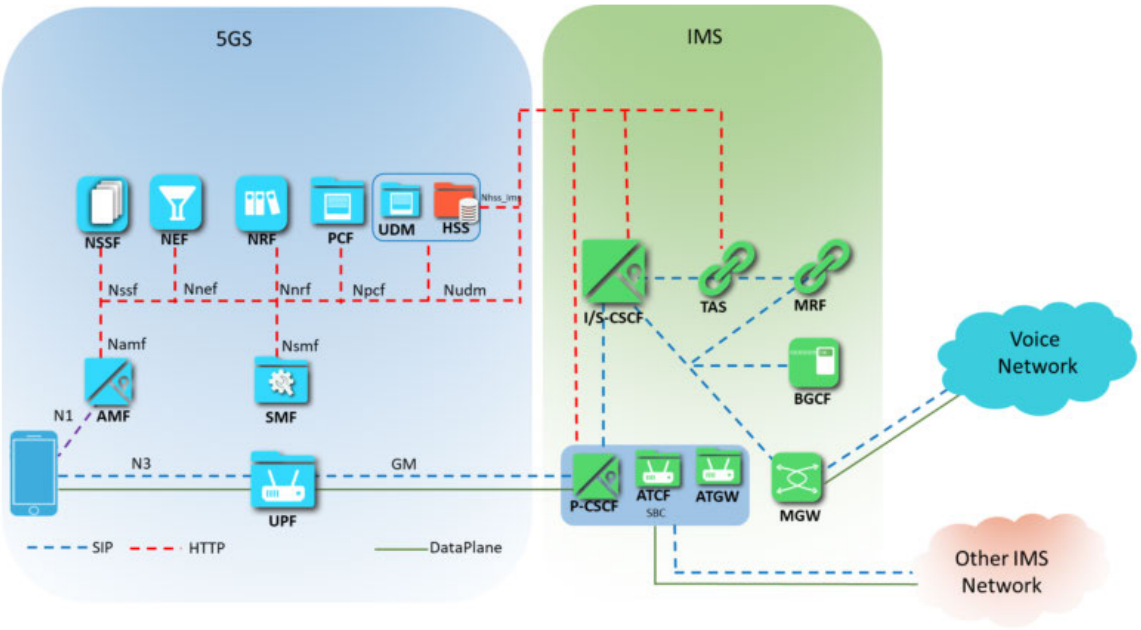


Figura 4.6. Implementación de un *Interworking Function* (IWF) para la traducción de paquetes *diameter* a HTTP, integrando el IMS con el 5G CN [56]

Capítulo 5

Resultados

5.1. Descripción de las pruebas realizadas y los resultados obtenidos

En este capítulo se definen las pruebas a las que se han sometido las distintas soluciones propuestas para evaluar sus capacidades y su correcto funcionamiento. Estas pruebas también servirán para validar las soluciones para entornos didácticos, así como evaluar el nivel de cumplimiento con las especificaciones del proyecto.

5.1.1. Escenario basado en *gnbsim*

Se llevará acabo una batería de pruebas que consisten en la captura y análisis de tráfico de red. Lo que se refleja en los siguientes apartados, es una síntesis de todas las pruebas realizadas. Todo el detalle de los comandos empleados, así como la secuencia de acciones seguidas se encuentran registradas en el ANEXO C.

5.1.1.1. Despliegue de los elementos de red

Después de haber completado todos los pasos previos para el desarrollo de la propuesta integrada con *gnbsim*, se puede continuar avanzando con el lanzamiento de los distintos componentes del núcleo de red, así como del propio *software gnbsim*. En primer lugar, previo al lanzamiento de los elementos del núcleo de red 5G, se debe ejecutar *Wireshark* para obtener la captura de tráfico generado por los componentes.

A continuación, se debe abrir una consola de comandos en el directorio "./oai-cn5g-fed/docker-compose" para ejecutar las sentencias de lanzamiento de los componentes del núcleo de red 5G. Cuando todos los elementos del núcleo se encuentren operativos, se podrán ejecutar *gnbsim*

(ver ANEXO C sección C.1.1.1). Una vez se hayan cargado con éxito todos los componentes del escenario, se puede detener la captura de *Wireshark* para observar si el intercambio de paquetes y la comunicación entre los distintos elementos se ha realizado de forma correcta y acorde a los estándares del 3GPP. La captura obtenida hasta este punto debe ser similar a la de la figura 5.1.

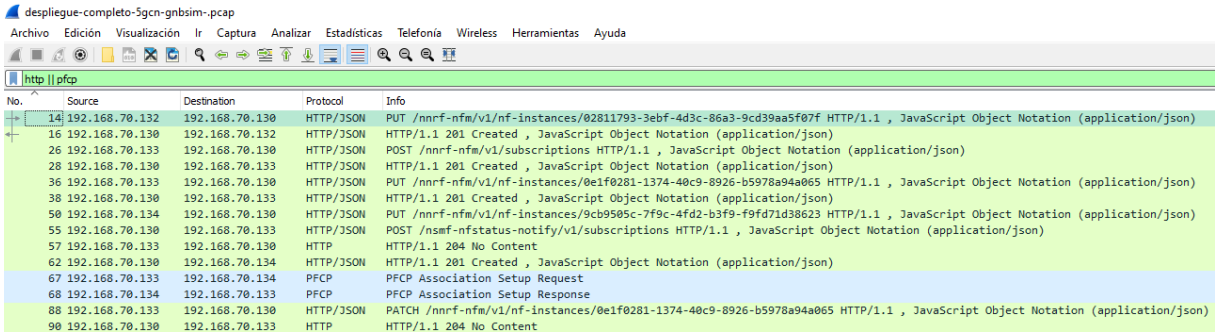


Figura 5.1. Tráfico intercambiado entre los elementos funcionales del escenario 5G preparado para RAN de *gnbsim* I

A la vista de los resultados arrojados por esta captura, destacamos los siguientes eventos:

- El SMF envía una petición inicial hacia el NRF para suscribirse a cualquier evento de UPF (Paquete nº 26).

[smf] [smf_app] [debug] Send ITTI msg to N11 task to subscribe to UPF status notification from NRF

Tabla 5.1: Fichero *smf.log*: Solicitud de registro enviada por el SMF al NRF

[nrf] [sbi_srv] [info] Got a request to create a new subscription [nrf] [nrf_app] [info] Handle Create a new subscription (HTTP version 1) [nrf] [sbi_srv] [info] Got a request to register an NF instance/Update an NF instance, Instance ID: 2a919e21-9530-45e6-9f3c-5dc4baf99764

Tabla 5.2: Fichero *nrf.log*: Solicitud de registro del SMF

- El SMF solicita el registro al NRF (Paquete nº: 36).

[smf] [sbi_srv] [info] HTTP2 server started
[smf] [smf_sbi] [debug] Send NFSubscribeNotify to NRF to be notified when a new UPF becomes available (HTTP version 1)
[smf] [smf_sbi] [debug] Send NFStatusNotify to NRF, NRF URL 192.168.70.130:80/nnrf-nfm/v1/subscriptions
[smf] [smf_sbi] [debug] Send NFStatusNotify to NRF, msg body:
“nfStatusNotificationUri”:“192.168.70.133:80/nsmf-nfstatus-notify/v1/subscriptions”, “reqNotifEvents”:[“NF_REGISTERED”, “NF_DEREGISTERED”], “subscrCond”:“NfTypeCond”:“nfType”:“UPF”, “validityTime”:“20390531T235959”

Tabla 5.3: Fichero *smf.log*: Solicitud de registro del SMF contra el NRF

[nrf] [sbi_srv] [info] Got a request to create a new subscription
[nrf] [nrf_app] [info] Handle Create a new subscription (HTTP version 1)
[nrf] [sbi_srv] [info] Got a request to register an NF instance/Update an NF instance, Instance ID: 2a919e21-9530-45e6-9f3c-5dc4baf99764
[nrf] [nrf_app] [info] Handle Register NF Instance/Update NF Instance (HTTP version 1)
[nrf] [nrf_app] [info] Check if a profile with this ID 2a919e21-9530-45e6-9f3c-5dc4baf99764 exist
[nrf] [nrf_app] [info] NF profile (ID 2a919e21-9530-45e6-9f3c-5dc4baf99764) not found
[nrf] [nrf_app] [info] Added/Updated NF Profile (ID 2a919e21-9530-45e6-9f3c-5dc4baf99764) to the DB
[nrf] [nrf_app] [info] Handle NF status registered event, profile id 2a919e21-9530-45e6-9f3c-5dc4baf99764
[nrf] [nrf_app] [info] Find a NF profile with ID 2a919e21-9530-45e6-9f3c-5dc4baf99764
[nrf] [nrf_app] [info] Get the list of subscriptions related to this profile, profile id 2a919e21-9530-45e6-9f3c-5dc4baf99764
[nrf] [nrf_app] [info] Verifying subscription, subscription id 1
[nrf] [nrf_app] [info] Added/Updated NF Instance, NF info: “capacity”:100, “heartBeatTimer”:10, “ipv4Addresses”:[“192.168.70.133”], “json_data”:null, “nfInstanceId”:“2a919e21-9530-45e6-9f3c-5dc4baf99764”, “nfInstanceName”:“OAI-SMF”, “nfServices”:[“ipEndPoints”:[“ipv4Address”:“192.168.70.133”, “port”:80, “transport”:“”], “nfServiceStatus”:“REGISTERED”, “scheme”:“http”, “serviceInstanceId”:“nsmf-pdusession”, “serviceName”:“nsmf-pdusession”, “versions”:[“apiFullVersion”:“1.0.0”, “apiVersionInUri”:“v1”]], “nfStatus”:“REGISTERED”, “nfType”:“SMF”, “priority”:1, “sNssais”:[“sd”:“123”, “sst”:222, “sd”:“1”, “sst”:1],

“smfInfo”:“sNssaiSmfInfoList”:[“dnnSmfInfoList”:[“dnn”:“default”], “sNssai”:“sd”:“123”,
“sst”:222, “dnnSmfInfoList”:[“dnn”:“oai”], “sNssai”:“sd”:“1”, “sst”:1,
“dnnSmfInfoList”:[“dnn”:“oai.ipv4”], “sNssai”:“sd”:“1”, “sst”:1]

Tabla 5.4: Fichero *nrf.log*: Registro del SMF

- El UPF se registra en el NRF (Paquetes nº: 50 y 62).

```
[spgwu] [spgwu_app] [start] Starting...
[spgwu] [spgwu_app] [info ] Send NF Instance Registration to NRF
[spgwu] [spgwu_sx ] [info ] handle_receive(38 bytes)
[spgwu] [spgwu_app] [info ] Response from NRF, HTTP Code: 201
[spgwu] [spgwu_sx ] [info ] Handle SX ASSOCIATION SETUP REQUEST
[spgwu] [spgwu_app] [info ] Response from NRF, JSON data:
“capacity”:100, “fqdn”:“oai-spgwu”, “heartBeatTimer”:10, “ipv4Addresses”:[“192.168.70.134”],
“json_data”:null, “nfInstanceId”:“4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244”,
“nfInstanceName”:“OAI-UPF”, “nfServices”:[], “nfStatus”:“REGISTERED”, “nfType”:“UPF”,
“priority”:1, “sNssais”:[“sd”:“123”, “sst”:222],
“upfInfo”:“sNssaiUpfInfoList”:[“dnnUpfInfoList”:[“dnn”:“default”], “sNssai”:“sd”:“123”,
“sst”:222] [spgwu] [spgwu_app] [start] Started
[spgwu] [spgwu_app] [info ] TIME-OUT event timer id 2
[spgwu] [spgwu_app] [info ] Send NF Update to NRF
[spgwu] [spgwu_sx ] [info ] handle_receive(16 bytes)
[spgwu] [spgwu_sx ] [info ] Received SX HEARTBEAT REQUEST
[spgwu] [spgwu_app] [info ] Response from NRF, HTTP Code: 204
[spgwu] [spgwu_app] [info ] Got successful response from NRF
```

Tabla 5.5: Fichero *spgwu.log*: Solicitud de registro enviada por el UPF al NRF

```
[nrf] [sbi_srv] [info ] Got a request to register an NF instance/Update an NF instance,
Instance ID: 4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244
[nrf] [nrf_app] [info ] Handle Register NF Instance/Update NF Instance (HTTP version 1)
[nrf] [nrf_app] [info ] Check if a profile with this ID
4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244 exist
[nrf] [nrf_app] [info ] NF profile (ID 4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244) not found
[nrf] [nrf_app] [info ] Added/Updated NF Profile (ID
4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244) to the DB
[nrf] [nrf_app] [info ] Handle NF status registered event, profile id
```

4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244
[nrf] [nrf_app] [info] Find a NF profile with ID 4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244
[nrf] [nrf_app] [info] Get the list of subscriptions related to this profile, profile id 4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244
[nrf] [nrf_app] [info] Verifying subscription, subscription id 1
[nrf] [nrf_app] [info] Subscription id 1, uri 192.168.70.133:80/nsmf-nfstatus-notify/v1/subscriptions
[nrf] [nrf_app] [info] Added/Updated NF Instance, NF info:
“capacity”:100, “fqdn”:“oai-spgwu”, “heartBeatTimer”:10, “ipv4Addresses”:[“192.168.70.134”], “json_data”:null, “nfInstanceId”:“4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244”, “nfInstanceName”:“OAI-UPF”, “nfServices”:[], “nfStatus”:“REGISTERED”, “nfType”:“UPF”, “priority”:1, “sNssais”:[“sd”:“123”, “sst”:222], “upfInfo”:“sNssaiUpfInfoList”:[“dnnUpfInfoList”:[“dnn”:“default”], “sNssai”:“sd”:“123”, “sst”:222] [nrf] [sbi_srv] [info]

Tabla 5.6: Fichero *nrf.log*: Solicitud de registro del UPF

- El NRF notifica al SMF del registro del UPF (Paquete nº: 55).

[smf] [sbi_srv] [info] NFStatusNotifyApiImpl, received a NF status notification...
[smf] [smf_app] [debug] Convert NotificationData (OpenAPI) to Data Notification Msg
[smf] [smf_app] [debug] NF instance info
[smf] [smf_app] [debug] Instance ID: 4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244
[smf] [smf_app] [debug] Instance name: OAI-UPF
[smf] [smf_app] [debug] Instance type: UPF
[smf] [smf_app] [debug] Status: REGISTERED
[smf] [smf_app] [debug] HeartBeat timer: 10
[smf] [smf_app] [debug] Priority: 1
[smf] [smf_app] [debug] Capacity: 100
[smf] [smf_app] [debug] SNSSAI:
[smf] [smf_app] [debug] SST 222, SD 123
[smf] [smf_app] [debug] FQDN: oai-spgwu
[smf] [smf_app] [debug] IPv4 Addr:
[smf] [smf_app] [debug] 192.168.70.134
[smf] [smf_app] [debug] UPF Info:
[smf] [smf_app] [debug] Parameters supported by the UPF:
[smf] [smf_app] [debug] SNSSAI (SST 222, SD 123)
[smf] [smf_app] [debug] DNN default
[smf] [smf_app] [info] Handle a NF status notification from NRF (HTTP version 1)

[smf] [smf_app] [debug] Resolve a DNS (name oai-spgwu, protocol http): Ip Addr 192.168.70.134, port 80
[smf] [smf_app] [debug] Add a new UPF node, FQDN oai-spgwu
[smf] [smf_app] [debug] Add a new UPF node, Ipv4 Addr 192.168.70.134
[smf] [smf_n4] [info] handle_receive(42 bytes)
[smf] [smf_n4] [debug] handle_receive_pfcp_msg msg type 6 length 38
[smf] [smf_n4] [info] Received N4 ASSOCIATION SETUP RESPONSE from an UPF
[smf] [smf_n4] [info] Received N4 ASSOCIATION SETUP RESPONSE

Tabla 5.7: Fichero *smf.log*: Notificación del NRF al SMF del registro de un nuevo UPF

- El SMF y el UPF establecen una sesión PFCP (Paquetes nº: 67 y 68).

[smf] [smf_app] [debug] Resolve a DNS (name oai-spgwu, protocol http): Ip Addr 192.168.70.134, port 80
[smf] [smf_app] [debug] Add a new UPF node, FQDN oai-spgwu
[smf] [smf_app] [debug] Add a new UPF node, Ipv4 Addr 192.168.70.134
[smf] [smf_n4] [info] handle_receive(42 bytes)
[smf] [smf_n4] [debug] handle_receive_pfcp_msg msg type 6 length 38
[smf] [smf_n4] [info] Received N4 ASSOCIATION SETUP RESPONSE from an UPF
[smf] [smf_n4] [info] Received N4 ASSOCIATION SETUP RESPONSE

Tabla 5.8: Fichero *smf.log*: Solicitud de registro del UPF II

Si cambiamos el filtro de *Wireshark* para analizar la captura en mayor profundidad obtenemos un resultado como el observado en la figura 5.2.

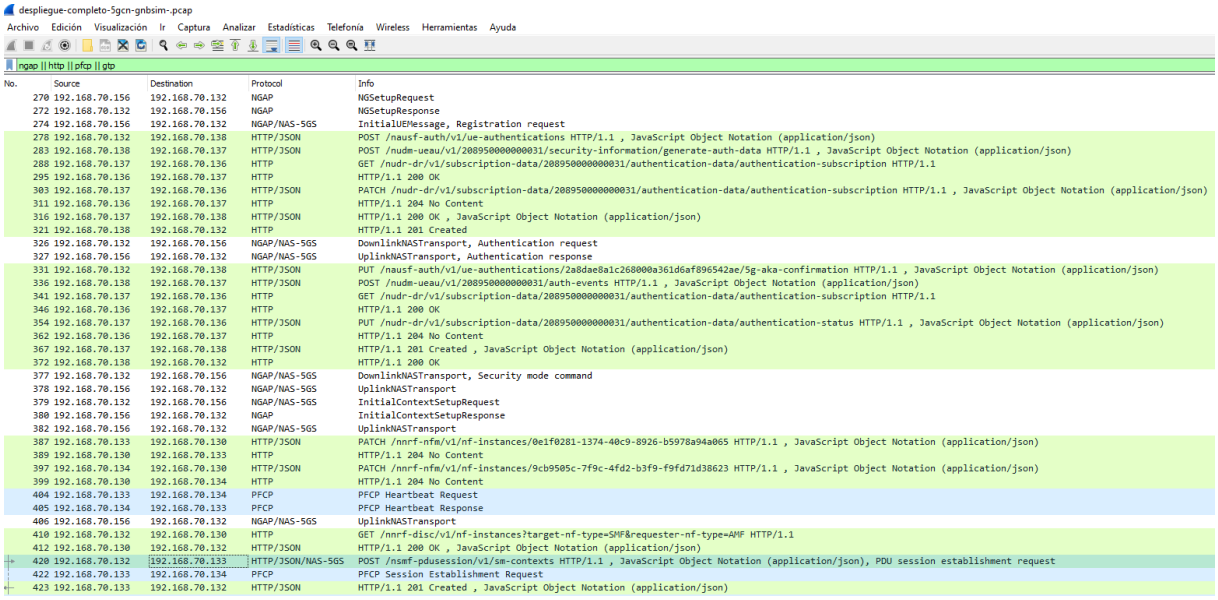


Figura 5.2. Tráfico intercambiado entre los elementos funcionales del escenario 5G preparado para RAN de *gnbsim* II

De esta captura, podemos observar:

- Establecimiento de la interfaz NG (Paquetes nº: 270 y 272).

```
Running gnbsim
[gnbsim]2022/06/15 16:52:32.818815 example.go:46: Dail LocalAddr:
127.0.0.1/192.168.70.156:34416; RemoteAddr: 192.168.70.132:38412
[gnbsim]2022/06/15 16:52:32.819006 example.go:66: write: len 44, info: &Stream:0 SSN:0
Flags:0 _:0 PPID:1006632960 Context:0 TTL:0 TSN:0 CumTSN:0 AssocID:0
[gnbsim]2022/06/15 16:52:32.888978 example.go:87: read: len 512, info: &Stream:0 SSN:0
Flags:0 _:0 PPID:1006632960 Context:0 TTL:0 TSN:1033202730 CumTSN:0 AssocID:13
[gnbsim]2022/06/15 16:52:32.890067 example.go:66: write: len 72, info: &Stream:0 SSN:0
Flags:0 _:0 PPID:1006632960 Context:0 TTL:0 TSN:1033202730 CumTSN:0 AssocID:13
Procedure Code: id-NGSetup (21)
PDU Length: 60
Protocol IEs: 4 items
  Item 0
    Protocol IE: id-AMFName (1)
      IE length: 9
        decoding id(1) not supported yet.
        dump: 03004f41492d414d46
  Item 1
```

```
      Protocol IE: id-ServedGUAMIList (96)
        IE length: 15
          decoding id(96) not supported yet.
          dump: 010002f8598000400064f0110a0041
    Item 2
      Protocol IE: id-RelativeAMFCapacity (86)
        IE length: 1
          decoding id(86) not supported yet.
          dump: 1e
    Item 3
      Protocol IE: id-PLMNSupportList (80)
        IE length: 16
          decoding id(80) not supported yet.
          dump: 0002f859000116f000007b100800000c
[gnbsim]2022/06/15 16:52:33.216122 example.go:87: read: len 528, info: &Stream:0 SSN:1
Flags:0 _:0 PPID:1006632960 Context:0 TTL:0 TSN:1033202731 CumTSN:0 AssocID:13
```

Tabla 5.9: Fichero *gnbsim.log*: Procedimiento *id-NGSetup*

```
[AMF] [amf_n2 ] [info ] Received NGSetupRequest message, handling
[AMF] [amf_n2 ] [debug] Handle NG Setup Request...
[AMF] [amf_n2 ] [debug] Parameters: assoc_id 14, stream 0
[AMF] [amf_n2 ] [debug] Update gNB context with assoc id (14)
[AMF] [amf_n2 ] [debug] RAN Node Info, Global RAN Node ID: 0x400, MCC 208, MNC
95
[AMF] [amf_n2 ] [warn ] Missing IE RanNodeName
[AMF] [amf_n2 ] [debug] IE DefaultPagingDRX: 0
[AMF] [amf_n2 ] [debug] TAC configured 40960, TAC received 40960
[AMF] [amf_n2 ] [debug] Encoding NG_SETUP_RESPONSE ...
ServedGUAMIList::numberOfservedGUAMIItem (2)
mcc = 208 mnc = 95
PLMNSupportList::numberOfplmnsupportItemItem (1)
PLMNSupportItem::numOfSnssai 2
SuccessfulOutcome ::=
  procedureCode: 21
  criticality: 0 (reject)
  value: NGSetupResponse ::=
    protocolIEs: ProtocolIE-Container ::=
      NGSetupResponseIEs ::=
```

```
        id: 1
        criticality: 0 (reject)
        value: OAI-AMF
      NGSetupResponseIEs ::=
        id: 96
        criticality: 0 (reject)
        value: ServedGUAMIList ::=
          ServedGUAMIItem ::=
            gUAMI: GUAMI ::=
              pLMNIdentity: 02 F8 59
              aMFRegionID: 80
              aMFSetID: 00 40 (6 bits unused)
              aMFPointer: 00 (2 bits unused)
          ServedGUAMIItem ::=
            gUAMI: GUAMI ::=
              pLMNIdentity: 64 F0 11
              aMFRegionID: 0A
              aMFSetID: 00 40 (6 bits unused)
              aMFPointer: 04 (2 bits unused)
      NGSetupResponseIEs ::=
        id: 86
        criticality: 1 (ignore)
        value: 30
      NGSetupResponseIEs ::=
        id: 80
        criticality: 0 (reject)
        value: PLMNSupportList ::=
          PLMNSupportItem ::=
            pLMNIdentity: 02 F8 59
            sliceSupportList: SliceSupportList ::=
              SliceSupportItem ::=
                s-NSSAI: S-NSSAI ::=
                  sST: DE
                  sD: 00 00 7B
              SliceSupportItem ::=
                s-NSSAI: S-NSSAI ::=
                  sST: 01
                  sD: 00 00 0C
[AMF] [ngap ] [debug] er.encoded (512)
```

[AMF] [sctp] [debug] [Socket 23, Assoc ID 14] Sending buffer 0x619000020380 of 512 bytes on stream 0 with ppid 60
[AMF] [sctp] [debug] Successfully sent 512 bytes on stream 0
[AMF] [amf_n2] [debug] Sending NG_SETUP_RESPONSE Ok
[AMF] [amf_n2] [debug] gNB with gNB_id 0x400, assoc_id 14 has been attached to AMF

Tabla 5.10: Fichero *amf.log*: Procedimiento *id-NGSetup*

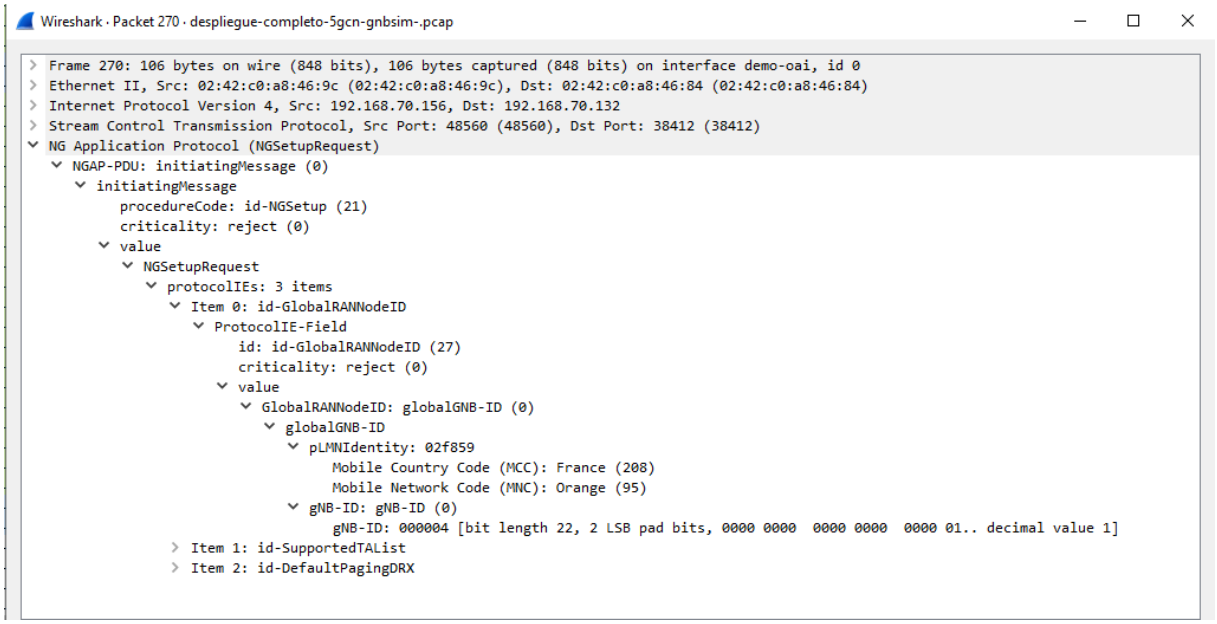


Figura 5.3. Captura *Wireshark*, paquete 270: Petición *NGSetup* del gNB al NRF

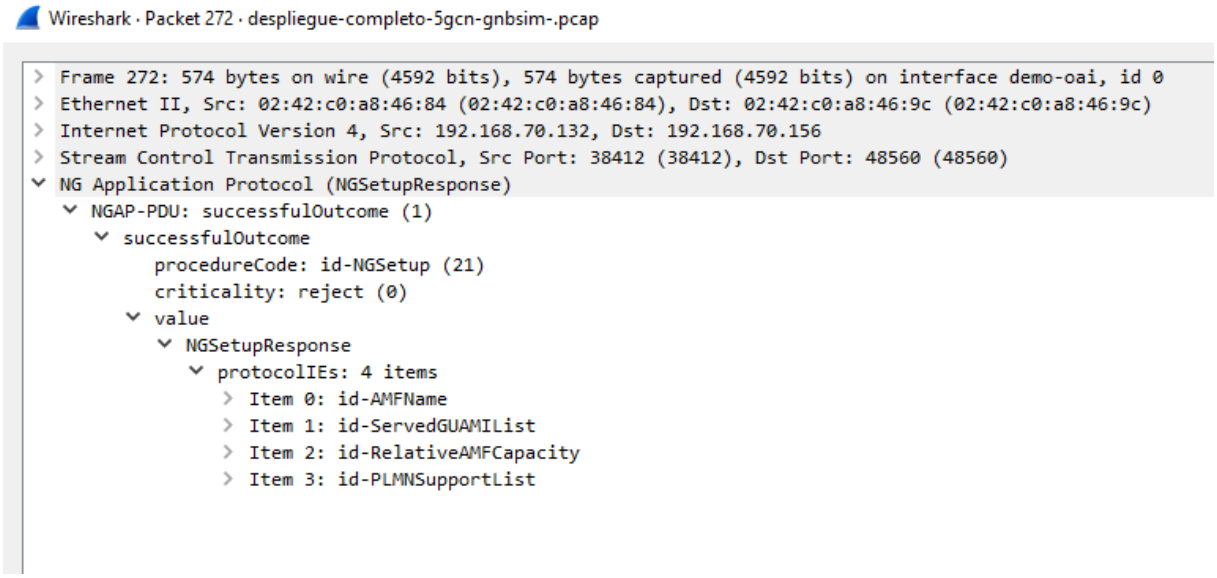


Figura 5.4. Captura *Wireshark*, paquete 272: Respuesta *NGSetup* del NRF al gNB

- Mensaje de registro del terminal UE (Paquetes nº: 274).

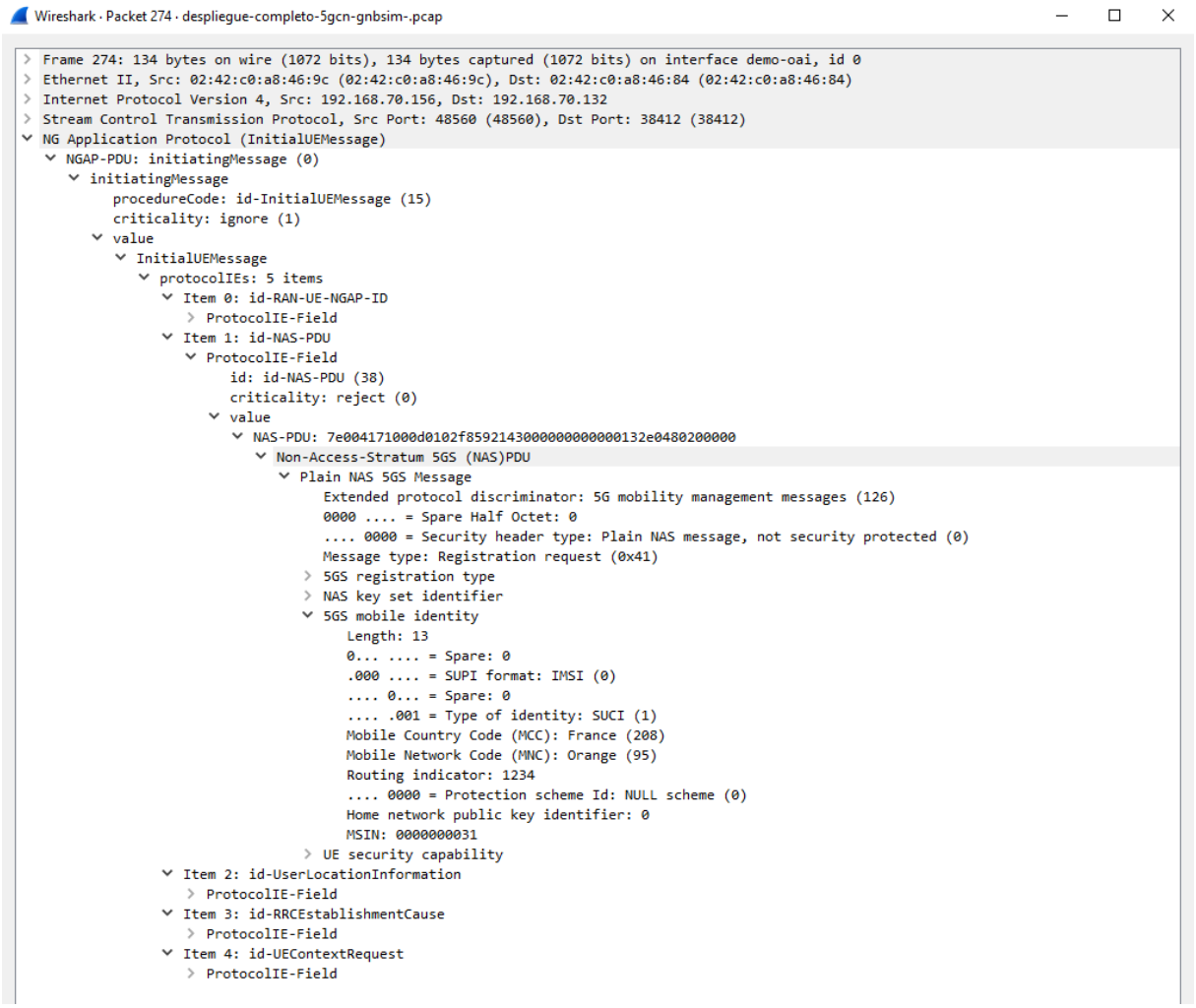


Figura 5.5. Captura *Wireshark*, paquete 274: Mensaje inicial del UE para registro

```
[AMF] [ngap ] [debug] Sending ITTI Initial UE Message to TASK_AMF_N2
[LibNGAP]Received RanUeNgapId 0
[AMF] [ngap ] [debug] Received RanUeNgapId 0
Received TAC: 40960
[AMF] [amf_n2 ] [info ] Received INITIAL_UE_MESSAGE message, handling
[AMF] [amf_n2 ] [debug] Handle Initial UE Message...
```

Tabla 5.11: Fichero *amf.log*: Mensaje inicial recibido por el AMF

- Autenticación y procesos de seguridad (Paquetes nº: 274, 326, 327 y 377).

[AMF] [amf_n2] [debug] Create a new UE NGAP context with ran_ue_ngap_id 0x0 [AMF] [amf_n2] [debug] 5g_s_tmsi not present [AMF] [amf_app] [debug] Received NAS_SIG_ESTAB_REQ [AMF] [amf_app] [debug] No existing UE Context, Create a new one with ran_amf_id app_ue_ranid_0:amfid_1

Tabla 5.12: Fichero *amf.log*: Extracto proceso de autenticación y seguridad

[AMF] [nas_mm] [debug] Decoded RegistrationRequest message (len 25) [AMF] [amf_n1] [debug] Received IMSI 208950000000031 [AMF] [amf_n1] [info] Associating IMSI (imsi-208950000000031) with nas_context (0x61600000ff80) [AMF] [amf_app] [debug] Add UE Info (IMSI 208950000000031) success [AMF] [amf_n1] [debug] Set 5GMM state to _5GMM_COMMON_PROCEDURE_INITIATED [AMF] [amf_n1] [debug] Signal the UE Registration State Event notification for SUPI imsi-208950000000031 [AMF] [amf_n1] [debug] Send request to SBI to triger UE Location Report (SUPI imsi-208950000000031) [AMF] [amf_n1] [debug] Start to run registration procedure [AMF] [amf_n1] [debug] SUCI SUPI format IMSI is available [AMF] [amf_n1] [debug] Authentication vector in nas_context is not available [AMF] [amf_n1] [debug] Start to generate authentication vectors [AMF] [amf_n1] [debug] Get Authentication Vectors from AUSF [AMF] [amf_n11] [debug] Send UE Authentication Request to AUSF (HTTP version 1) [AMF] [amf_n11] [debug] Send UE Authentication Request to AUSF, URL 192.168.70.138:80/nausf-auth/v1/ue-authentications [AMF] [amf_n11] [debug] Send UE Authentication Request to AUSF, msg body: “servingNetworkName”:“5G:mnc095.mcc208.3gppnetwork.org” , “supiOrSuci”:“208950000000031” [AMF] [amf_n11] [debug] UE Authentication, response from AUSF, HTTP Code: 201 [AMF] [amf_n11] [debug] UE Authentication, response from AUSF , “5gAuthData”:“autn”:“a800d507b4968000f44a60123ac9b080” , “hxresStar”:“aaae294d2e9c7984f98b9b6ad63571ff” , “rand”:“eb35c164d7501ad9c5d2197cec5086b8”,“_links”:“5G_AKA”:“href” : “http://192.168.70.138:80/nausf-auth/v1/ue-

authentications/a800d507b4968000f44a60123ac9b080/5g-aka-
confirmation”,“authType”:“5G_AKA”
Data: eb35c164d7501ad9c5d2197cec5086b8 (32 bytes)

Tabla 5.13: Fichero *amf.log*: Extracto proceso de autenticación y seguridad I

[AMF] [amf_n1] [debug] Received authentication response message, handling...
[AMF] [amf_n1] [info] Found nas_context (0x61600000ff80) with amf_ue_ngap_id (0x1)
[AMF] [nas_mm] [debug] Decoding AuthenticationResponse message
[AMF] [nas_mm] [debug] Decoded_size (3)
[AMF] [nas_mm] [debug] First option IEI (0x2d)
[AMF] [nas_mm] [debug] Decoding IEI (0x2D)
[AMF] [nas_mm] [debug] Decoding Authentication_Response_Parameter IEI 0x2d
[AMF] [nas_mm] [debug] Decoded Authentication_Response_Parameter (len 18)
[AMF] [nas_mm] [debug] Next IEI (0x0)
[AMF] [nas_mm] [debug] Decoded AuthenticationResponse message len (21)
[AMF] [amf_n1] [debug] _5g_aka_confirmation_from_ausf
[AMF] [amf_n1] [debug] resStar
7c a8 e6 ca d0 d7 65 8 a5 d4 e4 b0 2f ba dd a2
[AMF] [amf_n1] [info] resStar_s (7CA8E6CAD0D76508A5D4E4B02FBADDA2)
[AMF] [amf_n11] [info] Send HTTP message to http://192.168.70.138:80/nausf-
auth/v1/ue-authentications/a800d507b4968000f44a60123ac9b080/5g-aka-confirmation
[AMF] [amf_n11] [info] HTTP message Body:
“resStar”:“7CA8E6CAD0D76508A5D4E4B02FBADDA2”
[AMF] [amf_n11] [info] Get response with HTTP code (200)
Data: 74ca61489cbb9112bc5a813504c10d727969317e1e4eda4343f35b15da0ec4a6 (64 bytes)
Data (formatted):
74 ca 61 48 9c bb 91 12 bc 5a 81 35 4 c1 d 72 79 69 31 7e 1e 4e da 43 43 f3 5b 15 da e c4 a6
[AMF] [amf_n1] [debug] 5G AV: kseaf
74 ca 61 48 9c bb 91 12 bc 5a 81 35 4 c1 d 72 79 69 31 7e 1e 4e da 43 43 f3 5b 15 da e c4 a6
[AMF] [amf_n1] [debug] Deriving kamf
[AMF] [amf_n1] [debug] derive_kamf ...
[AMF] [amf_n1] [debug] kamf
12 cd 10 98 df 3f a4 d5 b4 49 8a 49 95 69 36 6d a 35 91 b2 ba 56 6e d5 f3 ca f5 fe b5 8b 8c
7a
[AMF] [amf_n1] [debug] Authentication successful by network!
[AMF] [amf_n1] [debug] Start Security Mode Control procedure
[AMF] [amf_n1] [debug] Using
INTEGRITY_PROTECTED_WITH_NEW_SECU_CTX for SecurityModeControl

message

Tabla 5.14: Fichero *amf.log*: Extracto proceso de autenticación y seguridad II

- Registro del terminal UE completado (Paquete nº: 378).

[AMF] [amf_n1] [debug] Registration-Accept message buffer 7e 0 42 1 1 77 0 b f2 2 f8 59 80 0 41 0 0 0 1 54 7 0 2 f8 59 0 a0 0 15 12 8 de 0 0 7b 0 0 0 0 8 1 0 0 c 0 0 0 0 21 2 1 0 5e 1 b6 [AMF] [amf_n1] [info] UE (IMSI 208950000000031, GUTI 20895128111, current RAN ID 0, current AMF ID 1) has been registered to the network [AMF] [amf_app] [debug] Update UE State (IMSI 208950000000031, State 5GMM-REGISTERED) success [AMF] [amf_n1] [debug] Set 5GMM state to _5GMM_REGISTERED [AMF] [amf_n1] [debug] Signal the UE Registration State Event notification for SUPI imsi-208950000000031 [AMF] [amf_n1] [debug] Send request to SBI to triger UE Registration State Report (SUPI imsi-208950000000031)

Tabla 5.15: Fichero *amf.log*: Registro del UE aceptado

- Petición de establecimiento de sesión *Protocol Data Unit* (PDU) (Paquetes nº: 404 y 405).

- Petición del AMF hacia el NRF de la dirección del SMF (Paquetes nº: 410 y 412).

[AMF] [amf_n11] [debug] Send NFDiscovery to NRF to discover the available SMFs [AMF] [amf_n11] [debug] NFDiscovery, response from NRF, HTTP Code: 200 [AMF] [amf_n11] [debug] NFDiscovery, got successful response from NRF [AMF] [amf_n11] [debug] NFDiscovery, response from NRF, json data: “nfInstances”:[“capacity”:100 , “heartBeatTimer”:10 , “ipv4Addresses”:[“192.168.70.133”] , “json_data”:null , “nfInstanceId”:“2a919e21-9530-45e6-9f3c-5dc4baf99764” , “nfInstanceName”:“OAI-SMF” , “nfServices”:[“ipEndPoints”:[“ipv4Address”:“192.168.70.133” , “port”:80 , “transport”:“”] , “nfServiceStatus”:“REGISTERED” , “scheme”:“http” , “serviceInstanceId”:“nsmf-pdusession” , “serviceName”:“nsmf-pdusession” , “versions”:[“apiFullVersion”:“1.0.0” , “apiVersionInUri”:“v1”]] , “nfStatus”:“REGISTERED” , “nfType”:“SMF” , “priority”:1 , “sNssais”:[“sd”:“123” , “sst”:222 , “sd”:“1” , “sst”:1] , “smfInfo”:“sNssaiSmfInfoList”:[“dnnSmfInfoList”:[“dnn”:“default”] , “sNssai”:“sd”:“123” , “sst”:222 , “dnnSmfInfoList”:[“dnn”:“oai”] , “sNssai”:“sd”:“1” , “sst”:1 ,

“dnnSmfInfoList”:[“dnn”:“oai.ipv4”] , “sNssai”:“sd”:“1” , “sst”:1]] , “searchId”:“1” , “validityPeriod”:100000 [AMF] [amf_n11] [debug] S-NSSAI [SST- 222, SD -123] is matched for SMF profile
[AMF] [amf_n11] [debug] NFDiscovery, SMF Addr: 192.168.70.133, SMF Api Version: v1, SMF Port: 80
[AMF] [amf_n11] [debug] Decoded PTI for PDUSessionEstablishmentRequest(0x1)
[AMF] [amf_n11] [debug] Handle PDU Session Establishment Request (SUPI imsi-208950000000031, PDU Session ID 1)
[AMF] [amf_n11] [debug] Message body
“anType”:“3GPP_ACCESS” , “dnn”:“default” , “gpsi”:“msisdn-200000000001” , “n1MessageContainer”:“n1MessageClass”:“SM” , “n1MessageContent”:“contentId”:“n1SmMsg” , “pduSessionId”:1 , “pei”:“imei-200000000000001” , “requestType”:“INITIAL_REQUEST” , “sNssai”:“sd”:“123” , “sst”:222 , “servingNetwork”:“mcc”:“208” , “mnc”:“95” , “servingNfId”:“servingNfId” , “smContextStatusUri”:“http://192.168.70.132:80/nsmf-pdusession/callback/imsi-208950000000031/1” , “supi”:“imsi-208950000000031”

n1sm buffer: 2e0101c1ffff91
[AMF] [amf_n11] [debug] Call SMF service: 192.168.70.133:80/nsmf-pdusession/v1/sm-contexts

Tabla 5.16: Fichero *amf.log*: Descubrimiento del SMF

- Petición de establecimiento de sesión PDU entre AMF y SMF (Paquetes nº: 420 y 423).

[AMF] [ngap] [debug] Handle PDU Session Resource Setup Response
[LibNGAP]Received RanUeNgapId 0
[AMF] [amf_n11] [info] Receive Nsmf_PDUSessionUpdateSMContext, handling ...
[AMF] [amf_n11] [debug] Send PDU Session Update SM Context Request to SMF (SUPI imsi-208950000000031, PDU Session ID 1)
[AMF] [amf_n11] [debug] SMF URI:
192.168.70.133:80/nsmf-pdusession/v1/sm-contexts/1/modify
n1sm buffer: 0003e0c0a8469c9acb04420006
[AMF] [amf_n11] [debug] Call SMF service:
192.168.70.133:80/nsmf-pdusession/v1/sm-contexts/1/modify
[AMF] [amf_app] [debug] [Format string as Hex] Input string (26 bytes):
0003e0c0a8469c9acb04420006
[AMF] [amf_app] [debug] Data (formatted):
00 03 e0 c0 a8 46 9c 9a cb 04 42 00 06
[AMF] [amf_n11] [debug] Send HTTP message to SMF with body ——Boundary

Content-Type: application/json
“n2SmInfo”:“contentId”:“n2msg” , “n2SmInfoType”:“PDU_RES_SETUP_RSP”
——Boundary
Content-Type: application/vnd.3gpp.ngap
Content-Id: n2msg
[AMF] [amf_n11] [info] Get response with HTTP code (200)

Tabla 5.17: Fichero *amf.log*: Establecimiento de sesión PDU con el SMF

[smf] [sbi_srv] [info] Received a SM context create request from AMF.
[smf] [smf_app] [debug] Parsing the message with Simple Parser
[smf] [smf_app] [debug] Boundary: —-Boundary
[smf] [smf_app] [debug] Content Type: application/json
[smf] [smf_app] [debug] Body: “anType”:“3GPP_ACCESS” , “dnn”:“default” ,
“gpsi”:“msisdn-200000000001” , “n1MessageContainer”:“n1MessageClass”:“SM” ,
“n1MessageContent”:“contentId”:“n1SmMsg” , “pduSessionId”:1 ,
“pei”:“imei-200000000000001” , “requestType”:“INITIAL_REQUEST” , “sNssai”:“sd”:“123” ,
“sst”:222 , “servingNetwork”:“mcc”:“208” , “mnc”:“95” , “servingNfId”:“servingNfId” ,
“smContextStatusUri”:“http://192.168.70.132:80/nsmf-pdusession/callback/imsi-
208950000000031/1” , “supi”:“imsi-208950000000031”
[smf] [smf_app] [debug] Content Type: application/vnd.3gpp.5gnas
[smf] [smf_app] [debug] Body:
[smf] [sbi_srv] [debug] Number of MIME parts 2
[smf] [sbi_srv] [info] PDU Session Create SM Context Request.
[smf] [sbi_srv] [debug] Create a pdu_session_create_sm_context_request message and
store the necessary information
[smf] [smf_app] [debug] Convert SmContextMessage (OpenAPI) to
PDUSession_CreateSMContext
[smf] [smf_app] [debug] N1 SM message: .
[smf] [smf_app] [debug] SUPI imsi-208950000000031, SUPI Prefix imsi, IMSI
208950000000031
[smf] [smf_app] [debug] DNN default
[smf] [smf_app] [debug] S-NSSAI SST 222, SD 123
[smf] [smf_app] [debug] PDU Session ID 1
[smf] [smf_app] [debug] ServingNfId servingNfId
[smf] [smf_app] [debug] RequestType INITIAL_REQUEST
[smf] [smf_app] [debug] SMContextStatusUri
http://192.168.70.132:80/nsmf-pdusession/callback/imsi-208950000000031/1
[smf] [smf_app] [warn] No SelMode available

[smf] [smf_app] [debug] Serving Network (MCC 208, MNC 95)
[smf] [smf_app] [debug] AN Type 3GPP_ACCESS
[smf] [sbi_srv] [debug] Promise ID generated 10
[smf] [smf_app] [info] Handle a PDU Session Create SM Context Request from an AMF (HTTP version 1)
[smf] [smf_app] [debug] Create a new SMF context with SUPI 208950000000031
[smf] [smf_app] [debug] The Session Management Subscription data is not available
[smf] [smf_app] [debug] Retrieve Session Management Subscription data from local configuration
[smf] [smf_app] [debug] Get Session Management Subscription from the configuration file
[smf] [smf_app] [debug] Default session type IPV4
[smf] [smf_app] [debug] Session AMBR Uplink 20Mbps, Downlink 22Mbps
[smf] [smf_app] [info] Inserted DNN Subscription, key: 222, dnn default
[smf] [smf_app] [debug] Generated a SMF Context ID 0x1
[smf] [smf_app] [info] Handle a PDU Session Create SM Context Request message from AMF (HTTP version 1)
[smf] [smf_app] [info] Find PDU Session with ID 1
[smf] [smf_app] [debug] Create a new PDU session
[smf] [smf_app] [info] Add PDU Session with Id 1
[smf] [smf_app] [debug] PDU Session Id (1) has been added successfully

Tabla 5.18: Fichero *smf.log*: Establecimiento de sesión PDU con el AMF

- Establecimiento de sesión PFCP entre SMF y UPF (Paquetes nº: 422 y 428).

[smf] [smf_app] [debug] Resolve a DNS (name oai-spgwu, protocol http): Ip Addr 192.168.70.134, port 80
[smf] [smf_app] [debug] Add a new UPF node, FQDN oai-spgwu
[smf] [smf_app] [debug] Add a new UPF node, Ipv4 Addr 192.168.70.134
[smf] [smf_n4] [info] handle_receive(42 bytes)
[smf] [smf_n4] [debug] handle_receive_pfcp_msg msg type 6 length 38
[smf] [smf_n4] [info] Received N4 ASSOCIATION SETUP RESPONSE from an UPF
[smf] [smf_n4] [info] Received N4 ASSOCIATION SETUP RESPONSE
[smf] [smf_app] [info] Node ID Type FQDN: oai-spgwu
[smf] [smf_app] [info] Node ID Type FQDN: oai-spgwu, IPv4 Addr: 192.168.70.134
[smf] [smf_app] [info] Associate with UPF profile
[smf] [smf_app] [debug] NF instance info
[smf] [smf_app] [debug] Instance ID: 4bc4c9b4-9d36-47c2-93fb-d1d08d8e2244
[smf] [smf_app] [debug] Instance name: OAI-UPF

```
[smf] [smf_app] [debug] Instance type: UPF
[smf] [smf_app] [debug] Status: REGISTERED
```

Tabla 5.19: Fichero *amf.log*: Extracto del establecimiento de la sesión PFCP con el UPF

- Dirección IP asignada al UE (Paquete nº: 422).

```
[smf] [smf_app] [debug] UE Address Allocation
[smf] [smf_app] [info ] Find a DNN Subscription with key: 222, map size 1
[smf] [smf_app] [info ] Find DNN configuration with DNN default
[smf] [smf_app] [debug] PDU Session Type IPv4
[smf] [smf_app] [info ] PAA, Ipv4 Address: 12.1.1.2
[smf] [sbi_srv] [debug] AMF IP Addr 192.168.70.132
```

Tabla 5.20: Fichero *smf.log*: IP del UE registrada desde el SMF

UEs' information				
Index	Status	Global ID	gNB Name	PLMN
1	Connected	0x400		208.95

Tabla 5.21: Fichero *amf.log*: Registro del AMF I

UEs' information								
Index	5GMM state	IMSI	GUTI	RAN UE NGAP ID	AMF UE ID	PLMN	Cell ID
1	5GMM-REGISTERED	208950000000031		0	1	208,95	262160

Tabla 5.22: Fichero *amf.log*: Registro del AMF II

Después de haber analizado el tráfico inicial, se realizarán las pruebas para el estudio del comportamiento de la solución propuesta. Es importante que cuando se concluya la parte de despliegue y las pruebas de este escenario se siga el procedimiento para el cierre ordenado de los distintos componentes de la red. En ANEXO C sección C.1.1.3 se describen los pasos a seguir.

5.1.1.2. Pruebas de ping e iperf

Como se ha mencionado en la sección anterior, una vez se ha realizado la captura del tráfico intercambiado entre los distintos elementos de la red, se procederá con la evaluación del desempeño de estos. En este caso las pruebas realizadas sobre la red son las siguientes:

- Pruebas de conectividad ping:
  - Desde una red externa simulada por "oai-ext-dn terminal" UE.
  - Desde terminal UE hacia "google.com".
  - Entre terminales UE.
- Pruebas de tráfico con *iperf* entre UE y nodo externo simulado.

Tras la realización de las pruebas de conectividad, queda demostrado que los elementos del núcleo de red proporcionados por OAI cumplen satisfactoriamente con las expectativas funcionales básicas para este proyecto.

Todo el detalle de las pruebas de tráfico y funcionamiento se encuentran con mayor detalle en el ANEXO C sección C.1.1.

5.1.1.3. Análisis de ficheros *log*

OAI nos permite analizar la información de registro de los diferentes elementos del núcleo de la red 5G, concretamente podemos extraer información de los siguientes componentes:

- NRF
- AMF
- SMF
- UPF
- *GNBSIM*

Esta capacidad resulta especialmente útil para realizar labores de depuración, así como para estudiar los distintos procedimientos y mensajes intercambiados y su conformidad con las normas establecidas por 3GPP.

5.1.2. Escenario basado en *UERANSIM*

A continuación, se llevarán a cabo pruebas de funcionamiento sobre el escenario basado en OAI y $UERANSIM$. Estas pruebas consisten en la captura y el análisis del tráfico que se intercambia entre los elementos del escenario.

Lo que se refleja en los siguientes apartados es una síntesis de todas las pruebas realizadas. Para un mayor detalle de los comandos empleados, así como la secuencia de acciones seguidas para obtener los resultados descritos en este apartado, se puede consultar el ANEXO C sección C.1.2.1.

5.1.2.1. Despliegue de los elementos de red

Llegados a esta sección, resulta vital finalizar los pasos previos de instalación y configuración (ver ANEXO B apartados B.2 y B.3) para la implementación de la solución que integra OAI con $UERANSIM$. Una vez completados, se puede proceder con la puesta en marcha de los distintos componentes del núcleo de la red, así como del propio $UERANSIM$.

En primer lugar, se debe iniciar la captura del tráfico con $Wireshark$ en la interfaz "demo-oai". A continuación, se debe abrir una consola de comandos en el directorio "./oai-cn5g-fed/docker-compose" para lanzar los componentes del núcleo de red. Cuando todos los elementos del núcleo se encuentren operativos, entonces se podrán ejecutar el software proporcionado por $UERANSIM$ (ver ANEXO C sección C.1.2.1).

Una vez se hayan generado con éxito todos los componentes del escenario, se puede detener la captura de $Wireshark$ para observar el intercambio de paquetes inicial, es decir, la comunicación entre los distintos elementos.

Después de haber analizado la comunicación inicial, se deben realizar las pruebas pertinentes para el estudio del comportamiento de esta solución y su adecuación para uso en entornos didácticos. De nuevo, antes de comenzar con cualquiera de las pruebas descritas a continuación, se debe de iniciar una captura de tráfico con el $software\ Wireshark$ en la interfaz "demo-oai".

Es importante que cuando se concluya la parte de despliegue y las pruebas de este escenario se siga el procedimiento para el cierre ordenado de los distintos componentes de la red. En ANEXO C sección C.1.2.3 se describen los pasos a seguir.

5.1.2.2. Pruebas de ping e iperf

En esta sección se describirán las pruebas realizadas para evaluar la conectividad entre los distintos elementos que componen la red desplegada, así como su desempeño. Estas pruebas son las siguientes:

- Pruebas de conectividad ping:
  - Desde una red externa simulada por "$oai - ext - dn$" terminal UE.
  - Desde terminal UE hacia "google.com".
  - Entre terminales UE.
- Pruebas de tráfico con $iperf$ entre UE y nodo externo simulado.

Tras la realización de las pruebas de conectividad, queda demostrado que los elementos del núcleo de red proporcionados por OAI cumplen satisfactoriamente con las expectativas de funcionamiento básicas para este proyecto y son completamente integrables con el $software$ $UERANSIM$. Todo el detalle de las pruebas de tráfico y funcionamiento se encuentran en mayor detalle en el ANEXO C sección C.1.2.

5.1.2.3. Análisis de ficheros log

OAI nos permite analizar la información de registro de los diferentes elementos del núcleo de la red 5G, concretamente podemos extraer información de los siguientes componentes:

- NRF
- AMF
- SMF
- UPF
- $UERANSIM$

La capacidad de analizar los ficheros de registro resulta especialmente útil para realizar labores de depuración, así como para estudiar los distintos procedimientos, sus mensajes intercambiados y su conformidad con las normas 3GPP.

5.1.3. Herramientas integradas *UERANSIM*

Adicionalmente, $UERANSIM$ proporciona un paquete de funcionalidades clasificadas como "experimentales" [57], a pesar de este hecho, se ha decidido abordar su estudio e implementación para comprobar el nivel de compatibilidad con el núcleo de red proporcionado por OAI. Esta serie de funcionalidades pueden resultar especialmente útiles para interactuar con el escenario de red a través de los terminales de usuario (UEs). Concretamente, podemos realizar las siguientes operaciones:

- A través de un nodo gNB (consultar ANEXO C sección C.1.5.1).
- A través de cualquier terminal UE (consultar ANEXO C sección C.1.5.2).

Todas las pruebas mencionadas en este apartado se encuentran desarrolladas a un mayor nivel de detalle en el ANEXO C apartado C.1.5.

5.1.4. Escenario final: *gnbsim* y *UERANSIM*

5.1.4.1. Despliegue de los elementos de red

A continuación, se llevarán a cabo pruebas de funcionamiento sobre el escenario basado en OAI, $UERANSIM$ y *gnbsim*. Estas pruebas consisten en la captura y el análisis del tráfico que se intercambia entre los elementos del despliegue. Este escenario final, tiene como objetivo integrar las dos soluciones $software$ para la parte de acceso radio ($UERANSIM$ y *gnbsim*. El objetivo, consiste en evaluar el comportamiento de ambas tecnologías, trabajando de forma conjunta en una misma red 5G SA.

5.1.4.2. Despliegue de los elementos de red

Para poder desplegar todos los elementos de este escenario de forma correcta, es muy importante seguir el orden indicado en las instrucciones que se muestran a continuación.

En primer lugar, se debe iniciar la captura del tráfico con $Wireshark$ en la interfaz "demo-oai". A continuación, se debe abrir una consola de comandos en el directorio "./oai-cn5g-fed/docker-compose" para lanzar los componentes del núcleo de red. Cuando todos los elementos del núcleo se encuentren operativos, entonces se podrán ejecutar los contenedores de *gnbsim* y $UERANSIM$ **en este orden** (ver ANEXO C sección C.1.3.1).

Una vez se hayan generado con éxito todos los componentes del escenario, se puede detener la captura de $Wireshark$ para observar el intercambio de paquetes inicial, es decir, la comunicación entre los distintos elementos.

Después de haber analizado la comunicación inicial, se deben realizar las pruebas pertinentes para el estudio del comportamiento de esta solución y su adecuación para uso en entornos didácticos. De nuevo, antes de comenzar con cualquiera de las pruebas descritas a continuación, se debe de iniciar una captura de tráfico con el $software$ $Wireshark$ en la interfaz "demo-oai".

Es importante que cuando se concluya la parte de despliegue y las pruebas de este escenario se siga el procedimiento para el cierre ordenado de los distintos componentes de la red. En ANEXO C sección C.1.3.3 se describen los pasos a seguir.

5.1.4.3. Pruebas de ping e iperf

En esta sección se describirán las pruebas realizadas para evaluar la conectividad entre los distintos elementos que componen la red desplegada, así como su desempeño. Estas pruebas son las siguientes:

- Pruebas de conectividad ping:
  - Desde una red externa simulada por "$oai - ext - dn$" terminal UE.
  - Desde terminal UE hacia "google.com".
  - Entre terminales UE.
- Pruebas de tráfico con $iperf$ entre UE y nodo externo simulado.

Tras la realización de las pruebas de conectividad, queda demostrado que los elementos del núcleo de red proporcionados por OAI cumplen satisfactoriamente con las expectativas de funcionamiento esperadas. Todo el detalle de las pruebas de tráfico y funcionamiento se encuentran en mayor detalle en el ANEXO C sección C.1.3.

5.1.4.4. Análisis de ficheros log

OAI nos permite analizar la información de registro de los diferentes elementos del núcleo de la red 5G, concretamente podemos extraer información de los siguientes componentes:

- NRF
- AMF
- SMF
- UPF
- $UERANSIM$ y $gnbsim$

La capacidad de analizar los ficheros de registro resulta especialmente útil para realizar labores de depuración, así como para estudiar los distintos procedimientos, sus mensajes intercambiados y su conformidad con las normas 3GPP.

Capítulo 6

Presupuesto

En este capítulo se especifica el presupuesto necesario para llevar a cabo este proyecto. A continuación, se detallarán los recursos *hardware*, *software* y humanos, que se han empleado para el despliegue de los distintos escenarios 5G SA para entornos didácticos.

6.1. Recursos hardware

Al disponer el estudiante de los recursos *hardware* necesarios para el desarrollo del proyecto y al tratarse de una solución basada completamente en virtualización, no ha sido necesaria la adquisición de ningún tipo de equipo o componente. Sin embargo, si se replica el proyecto sin equipamiento previo, el gasto en una máquina con los recursos *hardware* mínimos recomendados en el capítulo 4 sección 4.4 se estima en 984,34€. [58]

Por lo tanto, el **coste total de los recursos *hardware* podría ser de 984,34 €** si no se dispone previamente de los recursos mínimos.

6.2. Recursos software

A continuación, se listan los recursos *software* empleados en el proyecto:

- Imagen ISO del sistema operativo *Ubuntu* en su versión 20.04 LTS de 64 bits.
- *Software* de virtualización *Oracle VM VirtualBox*.
- Módulos *software* proporcionados por *OpenAirInterface* (OAI).
- Componentes *software* proporcionados por los desarrolladores de *UERANSIM*.
- Elementos *software* proporcionados por los desarrolladores de *gnbsim*.

Puesto que los desarrollos propuestos para llevar a cabo este proyecto están apoyados en su totalidad por *software* de código abierto y libre, no ha sido necesaria la adquisición de ningún tipo de solución *software*.

En conclusión, los **costes totales a razón de recursos *software* es de 0 €.**

6.3. Recursos humanos

A continuación, se detallan los recursos humanos que han contribuido en el proyecto:

- Un estudiante de Grado en Ingeniería Telemática, autor del proyecto, con un coste de 13,79 € por hora, durante 318 horas. [59]
    - Precio: 4.385,22 €.

Así pues, el **coste total de los recursos humanos asciende a: 4.385,22 €.**

6.4. Coste final

- Coste total de los recursos *hardware*: 984,34 €.
- Coste total de los recursos *software*: 0 €.
- Coste total de los recursos humanos: 4.385,22 €.

Coste total del proyecto: 5.369,56 €.

Capítulo 7

Conclusión y trabajos futuros

En este capítulo se detallan las conclusiones obtenidas después de haber realizado el proyecto, así como las posibles líneas de trabajo futuro que pueden llevarse a cabo.

7.1. Conclusiones

En este proyecto se ha llevado a cabo el despliegue de tres prototipos de red 5G virtualizados con el fin de explorar distintas soluciones para el desarrollo de una arquitectura de red 5G SA que soporte o integre servicios de VoNR. Todos los prototipos están formados por un núcleo de red común, formado por diversos contenedores *Docker* que permiten el despliegue e implementación de todos los elementos que componen el núcleo de una red 5G. Estas imágenes son desarrolladas y mantenidas por *OpenAirInterface* (OAI). Por otra parte, los componentes de la red de acceso radio (gNB y UEs) son proporcionados por *gnbsim* y $UERANSIM$.

Así pues, el diseño y despliegue de los prototipos de red se ha realizado de forma modular, mediante dos bloques perfectamente diferenciados, representando fielmente la arquitectura de una red 5G real. Su funcionamiento e integración con el núcleo de red proporcionado por OAI han sido probados de forma individual en escenarios dedicados de forma exclusiva a cada propuesta. Así mismo, se ha implementado un escenario final que aúna todas las soluciones en una única arquitectura de red 5G SA. Además, se han definido diversas pruebas para evaluar el nivel de validez de los diseños de red con respecto a los objetivos del proyecto. [36]

Para que los prototipos de red sean válidos para un entorno didáctico, es necesario que sea posible experimentar con ellos, es decir, modificar sus parámetros de configuración y observar su comportamiento. También se hace necesario que pueda profundizarse lo más posible en el funcionamiento interno del prototipo, esto es, como interactúan sus elementos, como se comunican entre ellos, como se transmite la información, etc. [60]

Tras la realización del proyecto se han obtenido las siguientes conclusiones:

- Las diferentes soluciones *software* se han desplegado en los distintos escenarios previstos para ello de forma satisfactoria y mostrando un nivel de integración muy elevado.
- De forma general, *OpenAirInterface* (OAI) ofrece una solución completa, expandible, ampliamente documentada y muy flexible para la implementación de núcleos de red 5G.
- *UERANSIM* proporciona una solución muy interesante para la implementación de terminales de usuario (UEs) y gNBs, en definitiva para implementación de redes de acceso radio virtuales. Esto es debido a que cuenta con una gran variedad de herramientas y comandos integrados que posibilitan la realización de una gran variedad de pruebas y experimentos que se pueden realizar con una red 5G SA y que cumplen con las especificaciones definidas para 5G.
- Todos los elementos que componen la red, tanto la parte del núcleo como la parte de red de acceso radio, son fácilmente configurables a través de diversos ficheros.
- Las pruebas realizadas mediante el análisis de tráfico corroboran que las soluciones *software* elegidas para el despliegue de los prototipos de red, cumplen con las especificaciones 3GPP. [15]
- La investigación realizada revela que el grado de madurez de los desarrollos de soluciones IMS para la implementación de VoNR es muy bajo aún. No se han encontrado soluciones *software* libre, que posibiliten la integración de los escenarios propuestos, con servicios VoNR. Se ha catalogado una posible solución *software*, que actualmente no es compatible con el núcleo de red proporcionado por OAI

7.2. Trabajos futuros

Después de la realización del proyecto se han definido las siguientes líneas para trabajos futuros:

- Despliegue, configuración e integración de un sistema IMS que pueda soportar VoNR. De entre todas las soluciones observadas durante la realización de este proyecto *Kamailio* promete grandes avances en este sentido, además de posibilitar la integración con el núcleo de red proporcionado por OAI. Actualmente el grado de madurez de estos desarrollos es muy bajo, sin embargo, en un futuro próximo cuando el foco de los desarrollos se centre en el despliegue de servicios sobre las redes 5G en lugar de solo centrarse en el desarrollo de los elementos del núcleo de red 5G, será posible el despliegue de nuevos servicios, entre ellos VoNR.

- Ampliación de la arquitectura de red, permitiendo realizar un mayor número de pruebas sobre los elementos básicos que componen la red 5G. Esta ampliación puede darse a través de desarrollos software propios que permitan explotar aún más las capacidades de estos escenarios, o a través de un seguimiento regular de los avances logrados por los desarrolladores de las soluciones *software* propuestas en este proyecto ($UERANSIM$, *gnbsim*, OAI).

Bibliografía

[1] Ministerio de asuntos económicos y transformación digital, *Estrategia de impulso de la tecnología 5G*, Madrid: ES, 2022. [en línea]. Disponible en: https://portal.mineco.gob.es/RecursosNoticia/mineco/prensa/noticias/2020/201201_np_impulso5G.pdf.

[2] *La última edición del Ericsson Mobility Report 2021.* [en línea]. Disponible en: https://www.ericsson.com/es/reports-and-papers/mobility-report/reports/november-2021 [consulta: 20 jun. 2022].

[3] *5G: The Revolution Begins - Drawing Capital Research.* [en línea]. Disponible en: https://drawingcapital.substack.com/p/5g-the-revolution-begins [consulta: 20 jun. 2022].

[4] Z. Imane, T. Mazri y E. Amine, «5G: ARCHITECTURE OVERVIEW AND DEPLOYMENTS SCENARIOS», *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci*, vol. XLIV-4/W3-2020, págs. 435 -440, 2020. DOI: https://doi.org/10.5194/isprs-archives-XLIV-4-W3-2020-435-2020. [consulta: 20 abr. 2022].

[5] M. Nohrborg, *"Long Term Evolution LTE - 3GPP"*, 2018. [En línea] Disponible en: https://www.3gpp.org/technologies/keywords-acronyms/98-lte. [Último acceso: 2022/05/07].

[6] S. Köksal, *Evolution of Core Network (3G vs. 4G vs. 5G)*, 2019. [En línea] Disponible en: https://medium.com/@sarpkoksal/core-network-evolution-3g-vs-4g-vs-5g-7738267503c7. [Último acceso: 2022/05/07].

[7] H. Remmert, *¿Qué es la arquitectura de red 5G?*, 2021. [En línea] Disponible en: https://es.digi.com/blog/post/5g-network-architecture#:~:text=El%20n%C3%BAcleo%205G%20utiliza%20una,se%20muestra%20en%20el%20diagrama. [Último acceso: 2022/05/07].

[8] A. Osseiran, et al., *5G Wireless access: an overview*, Disponible en https://www.ericsson.com/498a10/assets/local/reports-papers/white-papers/whitepaper-5g-wireless-access.pdf. [consulta: 16 abr. 2021].

[9] R. S. Shetty, *5G Mobile Core Network Desing, Deployment, Automation and Testing Strategies*, 1.ª ed. Apress, 2021.

[10] R. D. R. Álvarez, «Implementación de un Prototipo de Estación Base 5G mediante la Transmisión con USRPs», Tesis de mtría., Valencia ES, 2020.

[11] E. Dahlman y J. S. Stefan Parkvall, *5G NR: The next generaton Wireless Access Technology.* [en línea]. Disponible en: `https://learning.oreilly.com/library/view/5g-nr-the/9780128143247/xhtml/Cover.xhtml`.

[12] Samsung, *4G-5G Interworking, RAN-level and CN-level Interworking*, 2017. [en línea]. Disponible en: `https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-paper/4g-5g-interworking/global-networks-insight-4g-5g-interworking-0.pdf`.

[13] Techplayon, *Deployments Scenarios for 5G NR*, 2017. [en línea]. Disponible en: `https://www.techplayon.com/deployments-scenarios-for-5g-nr/`.

[14] J. Aranda, E. J. Sacoto-Cabrera, D. Haro-Mendoza y F. Astudillo-Salinas, «5G ntworks: A review from the perspectives of architecture, business models, cybersecurity, and research developments», *Revista Digital Novasinergia*, vol. 4, n.º 1, págs. 6-41, mayo de 2021. [en línea]. Disponible en: `https://doi.org/10.37135/ns.01.07.01`.

[15] «5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 17.4.0 Release 17)», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2022.

[16] *The challenge of policy and charging control in a 5G network.* [en línea]. Disponible en: `https://disruptive.asia/challenge-of-policy-charging-control-5g-network/` [consulta: 19 jun. 2022].

[17] Techplayon, *5G Core (5GC) Characteristics*, 2020. [en línea]. Disponible en: `https://www.techplayon.com/5g-core-5gc-characteristics/`.

[18] «5G; NG-RAN; Architecture description (3GPP TS 38.401 version 17.0.0 Release 17)», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2022.

[19] «5G; NR; NR and NG-RAN Overal description; Stage-2 (3GPP TS 38.300 version 17.0.0 Release 17)», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2022.

[20] P. C. Parrilla y C. R. Nespereira, *Sistemas móviles celulares 5G*, Redes de Comunicaciones Móviles.UD4.

[21] «5G; NG-RAN; NG Application Protocol (NGAP) (3GPP TS 38.413 version 17.0.0 Release 17)», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2022.

[22] «5G; NG-RAN; NG general aspects and principles (3GPP TS 38.410 version 16.2.0 Release 16)», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2020.

[23] *SCTP: ¿qué es el protocolo SCTP? - IONOS*. [en línea]. Disponible en: https://www.ionos.es/digitalguide/servidores/know-how/sctp/ [consulta: 26 jun. 2022].

[24] *GTP, vulnerabilidades en redes 4G y 5G. | by Marvin G. Soto | Medium*. [en línea]. Disponible en: https://marvin-soto.medium.com/gtp-vulnerabilidades-en-redes-4g-y-5g-a3af94048315 [consulta: 26 jun. 2022].

[25] RedHat, *¿What is NFV?*, Disponible en https://www.redhat.com/es/topics/virtualization/what-is-nfv. [consulta: 20 mayo 2022].

[26] E. Tittel, *¿SDN vs. NFV: What's the difference?*, Disponible en https://www.cisco.com/c/en/us/solutions/software-defined-networking/sdn-vs-nfv.html (2022/05/20).

[27] *European Telecommunications Standards Institute* (ETSI), «*Network Function Virtualisation* (NFV); Architectural Framework», 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2014.

[28] A. Z. Sánchez, «Estudio de soluciones para la orquestación de recursos Cloud/NFV», Tesis de mtría., Madrid ES, 2019.

[29] J. Lämmel, *Voice over New Radio (VoNR) | ng-voice*, Accessed Jun. 13, 2022. [en línea]. Disponible en: https://www.ng-voice.com/voice-over-new-radio-vonr/.

[30] S. Kinney, *What is 5G VoNR*, Accessed Jun. 13, 2022. [en línea]. Disponible en: https://rcrwireless.com/20220221/5g/what-is-5g-vonr.

[31] GSA, «Global Progress to *Voice Over New Radio* (VoNR) October 2021», GSA, Sheffield, UK, inf. téc., 2021. [en línea]. Disponible en: https://gsacom.com/paper/global-progress-to-voice-over-new-radio-vonr-october-2021/.

[32] GSA, «5G Standalone January 2022 – Member Report with Annex», GSA, Sheffield, UK, inf. téc., 2022. [en línea]. Disponible en: https://gsacom.com/paper/5g-standalone-january-2022-member-report-with-annex/.

[33] L. Bonati, M. Polese, S. D'Oro, S. Basagni y T. Melodia, «Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead», *Computer Networks*, vol. 182, pág. 107 516, 2020, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107516. [en línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1389128620311786.

[34] A. Gabilondo, Z. Fernandez, A. Martin y col., «5G SA Multi-vendor Network Interoperability Assessment», en *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2021, págs. 1-6. DOI: 10.1109/BMSB53066.2021.9547167.

[35] Open5GS, *Open Source Project of 5GC and EPC (Release-16)*, Último acceso May. 21, 2022. [en línea]. Disponible en: https://open5gs.org/.

[36] A. C. Jaen, «Desarrollo de un entorno virtualizado para emulación de una red 5G SA», Madrid ES, 2022.

[37] *free5GC*. [en línea]. Disponible en: https://www.free5gc.org/ [consulta: 30 jun. 2022].

[38] *5G Core Network*, Disponible en https://openairinterface.org/. [consulta: 13 mayo 2022].

[39] *All the needed repositories for the OpenAirInterface 5G core network development*, Disponible en https://gitlab.eurecom.fr/oai/cn5g. [consulta: 13 mayo 2022].

[40] *Openairinterface 5G Wireless Implementation*, Disponible en https://gitlab.eurecom.fr/oai/openairinterface5g. [consulta: 13 mayo 2022].

[41] A. Güngör, *UERANSIM*, Accessed Jun. 18, 2022. [en línea]. Disponible en: https://github.com/aligungr/UERANSIM.

[42] A. Güngör, *Feature Set*, Accessed Jun. 18, 2022. [en línea]. Disponible en: https://github.com/aligungr/UERANSIM/wiki/Feature-Set.

[43] *OpenAirInterface 5G Core Network Deployment and Testing with UERANSIM*, Disponible en https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/69af986694e9c983f86918de9e6f69bbe2436c40/docs/DEPLOY_SA5G_WITH_UERANSIM.md (2022/05/15).

[44] R. Kharade, *Gnbsim*, Disponible en https://gitlab.eurecom.fr/kharade/gnbsim. [consulta: 23 mayo 2022].

[45] Kamailio, *Features*, Accessed May. 30, 2022. [en línea]. Disponible en: https://www.kamailio.org/w/features/.

[46] V. info, *Kamailio*, Accessed May. 30, 2022. [en línea]. Disponible en: https://www.kamailio.org/w/features/.

[47] RedHat, *¿Qué es DOCKER?*, Disponible en https://www.redhat.com/es/topics/containers/what-is-docker?sc_cid=7013a000002wLvzAAE&gclid=CjwKCAjw7IeUBhBbEiwADhiEMXYDaDKRu9htQQQqgTRx_JYznKRcumI-2RcjdawOSpWqXQgqJRmPJRoCEdAQAvD_BwE&gclsrc=aw.ds. [consulta: 16 mayo 2022].

[48] Microsoft, *¿Qué es Docker?*, Disponible en https://docs.microsoft.com/es-es/dotnet/architecture/microservices/container-docker-introduction/docker-defined. [consulta: 16 mayo 2022].

[49] OSM VNF Onboarding Task Force, *OSM VNF ONBOARDING GUIDELINES*, 06921 Sophia Antipolis CEDEX, France, 2019. [en línea]. Disponible en: https://osm.etsi.org/wikipub/index.php/Release_notes_and_whitepapers.

[50] Open Source MANO, *OSM Quickstart*, 2022. [en línea]. Disponible en: https://osm.etsi.org/docs/user-guide/latest/01-quickstart.html.

[51] «Network Functions Virtualisation (NFV) Release 4; Architectural Framework; Report on further NFV support for 5G», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2021.

[52] Lenovo, *Lenovo IdeaPad Y700-15ISK - 80NV009BSP*, Accessed May. 24, 2022. [en línea]. Disponible en: https://www.tiendalenovo.es/lenovo-ideapad-y700-15isk-80nv009bsp.

[53] R. Defosseux, *Deploy-pre-requisites*, Accessed May. 28, 2022. [en línea]. Disponible en: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_PRE_REQUESITES.md.

[54] Docker, *User bridge networks*, Accessed May. 28, 2022. [en línea]. Disponible en: https://docs.docker.com/network/bridge/#enable-forwarding-from-docker-containers-to-the-outside-world.

[55] R. Defosseux, *Retrieve Official Images*, Accessed May. 28, 2022. [en línea]. Disponible en: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/RETRIEVE_OFFICIAL_IMAGES.md.

[56] G. Zone, *Voice Over 5G*, Accessed Jun. 12, 2022. [en línea]. Disponible en: https://the5gzone.com/index.php/voice-over-5g/.

[57] A. Güngör, *UERANSIM Usage*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://github.com/aligungr/UERANSIM/wiki/Usage.

[58] PCComponentes, *PcCom StartUp Pro Intel Core i7-11700/16GB/1TB SSD + Windows 11 Home*, Accessed Jun. 19, 2022. [en línea]. Disponible en: https://www.pccomponentes.com/pccom-startup-pro-intel-core-i7-11700-16gb-1tb-ssd-windows-11-home.

[59] Indeed, *¿Cuánto se gana en España, como Ingeniero/a en telecomunicaciones?*, Accessed Jun. 13, 2022. [en línea]. Disponible en: https://es.indeed.com/career/ingeniero-en-telecomunicaciones/salaries.

[60] S. P. Díaz, «Despliegue de prototipo de red LTE con OpenAirInterface y NexEPC: Extensión de red y ampliación de funcionalidades», Madrid ES, 2019.

[61] *Install Docker Engine on Ubuntu | Docker Documentation*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://docs.docker.com/engine/install/ubuntu/.

[62] *Use bridge networks*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://docs.docker.com/network/bridge/#enable-forwarding-from-docker-containers-to-the-outside-world.

[63] *Create a Docker ID*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://hub.docker.com/.

[64] S. Arora, *DEPLOY SA5G BASIC DS TESTER DEPLOYMENT*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_SA5G_BASIC_DS_TESTER_DEPLOYMENT.md.

[65] R. Defosseux, *DEPLOY SA5G WITH GNBSIM*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_SA5G_WITH_GNBSIM.md.

[66] R. Kharade, *DEPLOY SA5G WITH GNBSIM*, Accessed Jun. 08, 2022. [en línea]. Disponible en: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_SA5G_WITH_UERANSIM.md.

[67] S. Arora, *DEPLOY SA5G MINI DS TESTER DEPLOYMENT.* Accessed Jun. 08, 2022. [en línea]. Disponible en: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_SA5G_MINI_DS_TESTER_DEPLOYMENT.md#52-user-subscprition-profile.

[68] «5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.2.0 Release 15)», *European Telecommunications Standards Institute* (ETSI), 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - France, inf. téc., 2018.

Anexos

Anexo A

Preparación para la instalación

A.1. Instalación de *Docker*, *Docker Compose* y *Wireshark*

En esta sección se detallarán los pasos seguidos para la instalación del gestor de contenedores *Docker* y la herramienta *Docker Compose*, así como de *Wireshark*, proporcionando una descripción breve de cada comando usado y capturas de estos.

A.1.0.1. Instalación de *Docker* [61]

El primer paso, previo a todo el proceso de instalación tiene como objetivo que la instalación realizada sea una "instalación limpia", por lo que es recomendable la eliminación de posibles versiones anteriores de *Docker*. Para ello debemos ejecutar los comandos de la Tabla A.1.

```
root@pfg-VirtualBox:/home/pfg# apt-get remove docker docker-engine docker.io
containerd runc
```

Tabla A.1: Eliminación de posibles versiones anteriores de *Docker*

El resultado obtenido de la ejecución de este comando no tiene por qué ser en todos los casos exitoso, es decir, puede no existir ninguna versión anterior de *Docker*. Dado ese caso, la respuesta a su ejecución, sera una advertencia y un proceso de eliminación fallido. En cualquier caso, después de la ejecución del comando mostrado en la Tabla A.1, podemos comenzar con la instalación de *Docker*.

En primer lugar, antes de instalar *Docker Engine* por primera vez, se deben configurar sus repositorios. Una vez configurados, se procederá con la instalación y actualización de *Docker* desde el repositorio oficial.

En la Tabla A.2, se muestran los comandos necesarios para instalar los paquetes que permiten usar repositorios a través del protocolo HTTP, necesario para obtener de los repositorios oficiales de *Docker* su *software*.

```
root@pfg-VirtualBox:/home/pfg# apt-get update
root@pfg-VirtualBox:/home/pfg# apt-get install \
ca-certificates \
curl \
gnupg \
lsb-release
```

Tabla A.2: Instalación de *Docker* a través de repositorio oficial

Una vez ha finalizado el proceso, es necesario añadir las claves *GNU Privacy Guard* (GPG) oficiales de *Docker* ya que de lo contrario resultará imposible realizar la instalación desde su repositorio. Para ello debe ejecutarse el siguiente comando (ver Tabla A.3).

```
root@pfg-VirtualBox:/home/pfg# sudo mkdir -p /etc/apt/keyrings
root@pfg-VirtualBox:/home/pfg# curl -fsSL
https://download.docker.com/linux/ubuntu/gpg | sudo gpg –dearmor -o
/etc/apt/keyrings/docker.gpg
```

Tabla A.3: Añadir claves GPG oficiales

A continuación, el siguiente paso consistirá en ejecutar un comando para configura correctamente el repositorio. El comando es el mostrado en la Tabla A.4

```
root@pfg-VirtualBox:/home/pfg# echo \
>"deb [arch=$(dpkg –print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
>$(lsb_release -cs) stable"| sudo tee /etc/apt/sources.list.d/docker.list >/dev/null
```

Tabla A.4: Configuración del repositorio *Docker*

Después de ejecutar la sentencia de configuración, se puede proceder con la instalación del motor *Docker* en su última versión disponible, para ello deben ejecutarse los siguientes mandatos.

```
root@pfg-VirtualBox:/home/pfg# sudo apt-get update
root@pfg-VirtualBox:/home/pfg# sudo apt-get install docker-ce docker-ce-cli
```

Tabla A.5: Instalación del motor *Docker*

Una vez haya finalizado el proceso, *Docker Engine* estará instalado en nuestro sistema.

Adicionalmente, si se desea instalar otra versión de *Docker Engine*, también es posible. Para ello se debe ejecutar el comando presentado en la Tabla A.6. Después de su ejecución, se mostrarán las versiones disponibles en el repositorio de *Docker*.

```
root@pfg-VirtualBox:/home/pfg# apt-cache madison docker-ce
docker-ce | 5:20.10.16~3-0~ubuntu-focal
| https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
docker-ce | 5:20.10.15~3-0~ubuntu-focal
| https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
docker-ce | 5:20.10.14~3-0~ubuntu-focal
| https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
docker-ce | 5:20.10.13~3-0~ubuntu-focal
| https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
[ . . . ]
```

Tabla A.6: Enumeración de las versiones disponibles en el repositorio de *Docker*

En este caso, seleccionaremos la instalación de la versión 5:20.10.16 de docker-ce, es decir, la última disponible.

```
root@pfg-VirtualBox:/home/pfg# sudo apt-get install
docker-ce=5:20.10.16~3-0~ubuntu-focal docker-ce-cli=5:20.10.16~3-0~ubuntu-focal
containerd.io docker-compose-plugin
```

Tabla A.7: Instalación de la versión de *Docker* seleccionada

Final e independientemente del proceso de instalación que se haya seguido, es altamente recomendable verificar que la instalación de *Docker Engine* se ha realizado correctamente, esto podemos lograrlo ejecutando una prueba “hello-world”, tal como muestra la Tabla A.8

```
root@pfg-VirtualBox:/home/pfg# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:80f31da1ac7b312ba29d65080fddf797dd76acfb870e677f390d5acba9741b17
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Tabla A.8: Ejecución de *hello-word* para comprobar la instalación de *Docker Engine*

Llegados a este punto, si después de la ejecución del comando mostrado en la Tabla A.8 se ha obtenido la misma respuesta, es indicativo de que la instalación de *Docker Engine* se ha realizado satisfactoriamente.

A continuación, se procederá con la instalación de *Docker Compose* una herramienta necesaria para la ejecución de varios contenedores de forma simultánea y coordinada.

Al igual que en la instalación de *Docker*, se nos permite instalar la última versión disponible o seleccionar la versión a instalar. En el primer caso, para ejecutar la instalación de *Docker Compose* en su última versión disponible, se deberá ejecutar el comando que se muestra a continuación.

```
root@pfg-VirtualBox:/home/pfg# apt-get install docker-compose-plugin
```

Tabla A.9: Instalación de la última versión de *Docker Compose*

Si, por el contrario, deseamos instalar una versión concreta de *Docker Compose* debemos ejecutar el siguiente comando para que se listen las versiones que se encuentran disponibles en el repositorio oficial de *Docker*.

```
root@pfg-VirtualBox:/home/pfg# apt-cache madison docker-compose-plugin
docker-compose-plugin | 2.5.0~ubuntu-focal |
https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
docker-compose-plugin | 2.3.3~ubuntu-focal |
https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

Tabla A.10: Lista de versiones disponibles en el repositorio oficial para *Docker Compose*

Para este ejemplo y en nuestro caso, se optará por la instalación de la última versión disponible para *Ubuntu* 20.04. Así pues, debemos ejecutar el siguiente comando.

```
root@pfg-VirtualBox:/home/pfg# apt-get install
docker-compose-plugin=2.5.0~ubuntu-focal
```

Tabla A.11: Instalación de la versión de *Docker Compose* seleccionada

Finalmente, comprobamos que la instalación de *Docker Compose* se ha llevado a cabo de forma satisfactoria y para ello debemos emplear el comando mostrado en la Tabla A.12, que nos devolverá la versión de *Docker Compse* instalada, si el proceso ha sido satisfactorio.

```
root@pfg-VirtualBox:/home/pfg# docker compose version
Docker Compose version v2.5.0
```

Tabla A.12: Comprobación de la instalación de *Docker Compose*

A.1.0.2. Habilitar comunicación desde contenedores *Docker* hacia redes externas [62]

De forma predeterminada, el tráfico entre contenedores conectados al adaptador puente por defecto no se reenvía hacia redes externas. Para habilitar este reenvío, se deben realizar dos pasos. A continuación, se muestran los comandos que deben de ejecutarse. [54]

```
root@pfg-VirtualBox:/home/pfg# sysctl net.ipv4.conf.all.forwarding=1.
```

Tabla A.13: Configuración para permitir el reenvío de IP

```
root@pfg-VirtualBox:/home/pfg# sudo iptables -P FORWARD ACCEPT
```

Tabla A.14: Cambio en las políticas de las tablas IP

Debe tenerse en cuenta que estas configuraciones no son persistentes, esto es, al reiniciar la máquina anfitriona sera necesario volver a realizar este paso de configuración, de lo contrario es posible experimentar problemas durante el desarrollo de los escenarios.

Para cerrar esta sección, el último paso consistirá en registrarse en la plataforma de *Docker Hub*, a fin de tener acceso a los repositorios oficiales y, obtener las imágenes para la realización del proyecto. Para formalizar nuestro registro debemos darnos de alta como usuarios en su portal: https://hub.docker.com/signup, tal como puede observarse en la figura A.1

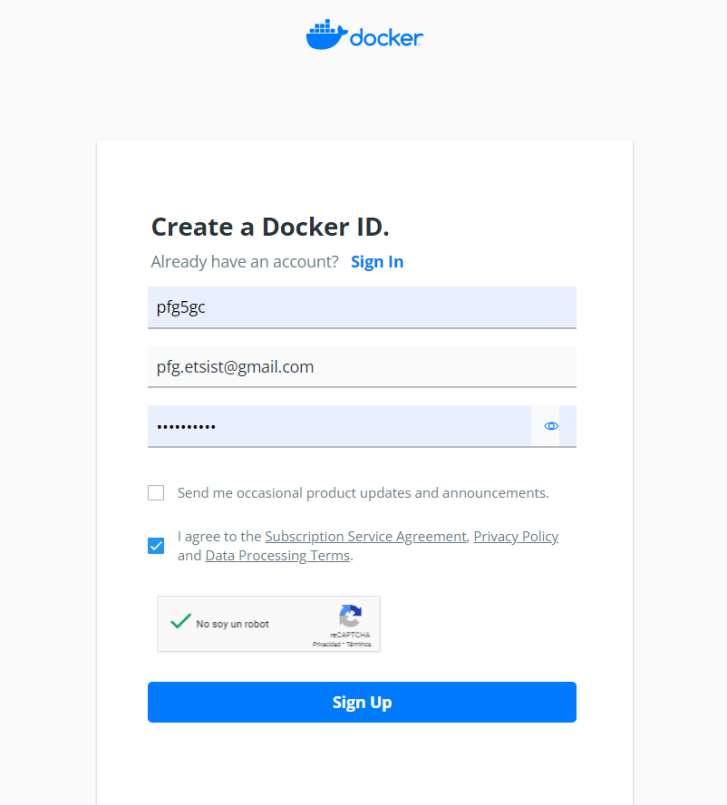


Figura A.1. Registro en *Docker* [63] [53]

Habiendo completado estos pasos, estaremos en disposición de continuar con la instalación del resto de componentes del *software* proporcionado por OAI a través de sus imágenes en *Docker*.

A.1.1. Instalación de *Wireshark* [64]

En primer lugar, es necesario añadir el repositorio oficial de *Wireshark* para su posterior instalación. En la Tabla A.15 puede verse el comando correspondiente.

root@pfg-VirtualBox:/home/pfg# add-apt-repository ppa:wireshark-dev/stable Latest stable Wireshark releases back-ported from Debian package versions.

Tabla A.15: Comando para añadir el repositorio de *Wireshark*

A continuación, una vez añadido el repositorio de *Wireshark*, se procederá con la actualización de los repositorios del sistema, tal como se muestra a continuación.

```
root@pfg-VirtualBox:/home/pfg# apt update
```

Tabla A.16: Actualización de los repositorios

Una vez finalizado el paso anterior, se puede proceder a la instalación de *Wireshark* a través del comando que se muestra en la Tabla A.17

```
root@pfg-VirtualBox:/home/pfg# apt install wireshark
```

Tabla A.17: Instalación de la última versión de *Wireshark*

Para terminar, es conveniente comprobar que la instalación se ha llevado a cabo de forma correcta, para ello ha de emplearse el comando que se muestra en la Tabla A.18.

```
root@pfg-VirtualBox:/home/pfg# wireshark –version
Wireshark 3.6.5 (Git v3.6.5 packaged as 3.6.5-1∼ubuntu20.04.0+wiresharkdevstable)
```

Tabla A.18: Comprobación de la instalación de *Wireshark*

A.1.2. Obtención de imágenes para el desarrollo del proyecto [53]

Las primeras imágenes que debemos de obtener, de acuerdo con la información proporcionada por OAI en su Wiki, son: *Ubuntu:bionic* y *MySQL:5.7*.
Sin embargo, adaptando estas recomendaciones a la versión actual de *Ubuntu*, las imágenes que debemos obtener son las siguientes:

- *Ubuntu:focal*
- *MySQL:latest*

En primer lugar, debemos de introducir las credenciales del usuario que se ha creado anteriormente, tal como se muestra en la tabla A.19.

```
root@pfg-VirtualBox:/home/pfg# docker login Login with your Docker ID to push and
pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username: pfg5gc
Password:
Login Succeeded
```

Tabla A.19: Operación login, en *Docker Hub*

A continuación, procedemos a obtener las imágenes de los repositorios oficiales a través de los siguientes comandos.

```
root@pfg-VirtualBox:/home/pfg# docker pull ubuntu:focal
```

Tabla A.20: Obtención de *Ubuntu:focal*

```
root@pfg-VirtualBox:/home/pfg# docker pull mysql:latest
```

Tabla A.21: Obtención de la imagen MySQL en su última versión estable

A continuación, se debe proceder con la obtención del resto de imágenes que contendrán el software necesario para la implementación de los escenarios que deseamos desarrollar en este proyecto.

A.1.2.1. Obteniendo las imágenes *Docker* oficiales para los componentes del núcleo de red 5G [55]

En esta sección se describen los pasos a seguir para obtener las imágenes *Docker* oficiales, proporcionadas por OAI para la implementación de los diferentes componentes del núcleo de red 5G. Los comandos adecuados, son los que se muestran en las tablas siguientes.

```
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-amf:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-nrf:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-spgwu-tiny:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-smf:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-udr:latest
```

```
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-udm:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-ausf:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-upf-vpp:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/oai-nssf:latest
root@pfg-VirtualBox:/home/pfg# docker pull oaisoftwarealliance/trf-gen-cn5g:latest
```

Tabla A.22: Obtención de las imágenes de los elementos del 5G CN proporcionadas por OAI

Después de haber obtenido las imágenes de los diferentes elementos que componen la red, procederemos a renombrarlos, para que sea más fácil reconocerlos. Para ello se deben ejecutar los comandos que se muestran a continuación:

```
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-amf:latest
oai-amf:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-nrf:latest
oai-nrf:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-smf:latest
oai-smf:latest
root@pfg-VirtualBox:/home/pfg# docker image tag
oaisoftwarealliance/oai-spgwu-tiny:latest oai-spgwu-tiny:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-udr:latest
oai-udr:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-udm:latest
oai-udm:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-ausf:latest
oai-ausf:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-upf-vpp:latest
oai-upf-vpp:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/oai-nssf:latest
oai-nssf:latest
root@pfg-VirtualBox:/home/pfg# docker image tag oaisoftwarealliance/trf-gen-cn5g:latest
trf-gen-cn5g:latest
```

Tabla A.23: Comandos para renombrar las imágenes proporcionadas por OAI

Una vez se han completado satisfactoriamente las operaciones, podemos hacer *logout* de nuestra sesión *Docker* ya que no será necesario obtener más imágenes por el momento.

A continuación, se procederá a la sincronización de las demostraciones proporcionados por OAI, ya que los escenarios desarrollados a lo largo de este proyecto estarán basados en ellos.

Para obtener estas configuraciones de prueba, o demostraciones, es necesario ejecutar los siguientes comandos:

```
root@pfg-VirtualBox:/home/pfg# git clone –branch v1.3.0
https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.git
root@pfg-VirtualBox:/home/pfg# cd oai-cn5g-fed
```

Tabla A.24: Clonado de los repositorios de OAI

```
root@pfg-VirtualBox:/home/pfg# ./scripts/syncComponents.sh
```

Tabla A.25: Sincronización de todos los módulos proporcionados por OAI

Llegados a este punto, se puede dar por finalizada la preparación del entorno de desarrollo para el despliegue de los distintos escenarios.

Para resumir, en esta sección se ha llevado a cabo la instalación de *Docker Engine*, así como la herramienta *Docker Compose*. A continuación, se han modificado los parámetros de red apropiados para habilitar el envío de paquetes desde los contenedores *Docker* hacia redes externas y viceversa, ejecutando los comandos correspondientes. Después de todo ello, se ha procedido con la instalación de *Wireshark* y con el registro en la plataforma *Docker Hub*, de donde se han obtenido las imágenes oficiales de los componentes del núcleo de red 5G, proporcionadas por OAI.

A.2. Instalación de Gnbsim y *UERANSIM*

En esta sección se describen los pasos a seguir para la obtención de las imágenes de *UERANSIM* y *gnbsim* así como su configuración para el correcto desempeño en el desarrollo de los distintos escenarios.

A.2.1. Gnbsim [65]

Para la instalación de *gnbsim* será necesario construir nuestra propia imagen *Docker* de este componente. Para ello es **MUY IMPORTANTE** abrir una terminal **FUERA** del directorio de trabajo **oai-cn5g-fed**, de lo contrario el componente no llegará a funcionar nunca. Una vez nos hemos asegurado de cumplir con el requisito anterior, debemos de ejecutar los siguientes comandos:

```
root@pfg-VirtualBox:/home/pfg# git clone https://gitlab.eurecom.fr/kharade/gnbsim.git
root@pfg-VirtualBox:/home/pfg# cd gnbsim
root@pfg-VirtualBox:/home/pfg# docker build –tag gnbsim:latest –target gnbsim –file
docker/Dockerfile.ubuntu.18.04 .
```

Tabla A.26: Construcción de la imagen *Docker* del componente *gnbsim*

A.2.2. *UERANSIM* [66]

Al igual que se ha que en el apartado anterior, para la instalación de *UERANSIM* debemos de construir nuestra propia imagen *Docker* de este elemento. Para ello debemos seguir la siguiente secuencia de comandos:

```
root@pfg-VirtualBox:/home/pfg# git clone -b docker_support
https://github.com/orion-belt/UERANSIM.git
root@pfg-VirtualBox:/home/pfg# cd UERANSIM
root@pfg-VirtualBox:/home/pfg# docker build –target ueransim –tag ueransim:latest -f
docker/Dockerfile.ubuntu.18.04 .
```

Tabla A.27: Construcción de la imagen *Docker* del elemento *UERANSIM*

Anexo B

Configuración para desarrollo de escenarios didácticos

B.1. Configuración de los escenarios

Todos los escenarios parten de un archivo con extensión *.yaml*.

Estos ficheros son especialmente importantes, debido a que son los que empleará *Docker* a través de *Docker-compose* para el despliegue de manera ordenada de cada uno de los elementos del núcleo de red 5G, junto con sus parámetros de configuraciones apropiados.

B.1.1. Configuración gnbsim

Para comenzar a trabajar con *gnbsim*, se debe crear una conexión puente con la red *demo-oai-public-net*. Este es un paso que resulta indispensable, debido a que sin la conexión puente resultaría imposible el funcionamiento de los distintos elementos del núcleo de red. Por lo tanto, debemos crear la conexión puente. Para ello existen dos métodos:

B.2. Creación de una conexión puente

B.2.1. Crear conexión puente manualmente

Este método es recomendable si tenemos interés en capturar tráfico entre los distintos elementos de la red, durante el arranque de estos. Por ejemplo, este modo de configuración puede emplearse cuando deseamos verificar que todas las funciones de red se han registrado de forma satisfactoria en el NRF. [67]

Para ello debemos cambiar la configuración del fichero *docker-compose* que se empleará para el despliegue de los componentes del núcleo de red. Este fichero, se encuentra en el siguiente directorio: *./oai-cn5g-fed/docker-compose/docker-compose-basic-nrf.yaml*. Si nos dirigimos a la parte final del fichero, encontraremos una configuración similar a la que se muestra a continuación:

```
networks:
    # public_net:
        # external:
            # name: demo-oai-public-net
    public_net:
        driver: bridge
        name: demo-oai-public-net
        ipam:
            config:
                - subnet: 192.168.70.128/26
        driver_opts:
            com.docker.network.bridge.name: "demo-oai"
```

Tabla B.1: Estado inicial del fichero docker-compose-basic-nrf.yaml

Los cambios que se deben efectuar sobre este fichero deben de ser tales que el resultado final sea exactamente el mismo que el mostrado en la siguiente tabla.

```
networks:
    public_net:
        external:
            name: demo-oai-public-net
    # public_net:
        # driver: bridge
        # name: demo-oai-public-net
        # ipam:
            # config:
                # - subnet: 192.168.70.128/26
        # driver_opts:
            # com.docker.network.bridge.name: "demo-oai"
```

Tabla B.2: Edición del fichero docker-compose-basic-nrf.yaml

A continuación, una vez se ha configurado el fichero .yaml para el despliegue del escenario, es necesario configurar el puente *demo-oai* en la máquina virtual. Para configurar el puente se debe introducir el comando que se muestra en la tabla B.3

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker network create \
    --driver=bridge \
    --subnet=192.168.70.128/26 \
    -o "com.docker.network.bridge.name"="demo-oai" \
    demo-oai-public-net
    ac22619cea9151e8145b7e992ee43b96e94edee01b27ddaab4d0a68169a2e961
```

Tabla B.3: Configuración del puente *demo-oai* en la máquina virtual

Para comprobar que los cambios se han realizado con éxito, es aconsejable introducir los siguientes comandos, cuyos resultados deben de ser similares a los mostrados a continuación.

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# ifconfig demo-oai
demo-oai: flags=4099<UP,BROADCAST,MULTICAST>mtu 1500
    inet 192.168.70.129 netmask 255.255.255.192 broadcast 192.168.70.191
    inet6 fe80::42:cbff:feea:eb15 prefixlen 64 scopeid 0x20<link>
    ether 02:42:cb:ea:eb:15 txqueuelen 0 (Ethernet)
    RX packets 8468 bytes 346515 (346.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10587 bytes 27345543 (27.3 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Tabla B.4: Comprobación configuración del puente *demo-oai* en la máquina virtual I

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker network ls
 NETWORK ID     NAME                DRIVER    SCOPE
 e324e1663318   bridge              bridge    local
 ac22619cea91   demo-oai-public-net bridge    local
```

Tabla B.5: Comprobación configuración del puente *demo-oai* en la máquina virtual II

Realizados estos cambios, la creación y configuración del puente estaría completada y lista para el despliegue de los diferentes escenarios de trabajo.

B.2.2. Crear conexión puente de forma automática

Este método es recomendable si la capturar tráfico entre los distintos elementos de la red durante el arranque no es nuestro principal interés.

Para ello debemos mantener la configuración del fichero *docker-compose* que se empleará para el despliegue de los componentes del núcleo de red. En este caso, se encuentra en el directorio *./oai-cn5g-fed/docker-compose/docker-compose-basic-nrf.yaml* y su aspecto es el siguiente:

```
networks:
    # public_net:
        # external:
            # name: demo-oai-public-net
    public_net:
        driver: bridge
        name: demo-oai-public-net
        ipam:
            config:
                - subnet: 192.168.70.128/26
        driver_opts:
            com.docker.network.bridge.name: "demo-oai"
```

Tabla B.6: Configuración del fichero docker-compose-basic-nrf.yaml

Una vez se han completado los pasos de cualquiera de las dos opciones, los ficheros de configuración estarán listos para funcionar con *gnbsim*.

Sin embargo, para extender la compatibilidad con *UERANSIM* es necesario realizar cambios en esta configuración. Para obtener todos los detalles y, antes de avanzar hacia la sección de configuración de *UERANSIM* (Resulta extremadamente importante consultar el Anexo C apartado C.2.1).

B.3. Configuración *UERANSIM*

Para el correcto funcionamiento y compatibilidad entre los distintos elementos de *UERANSIM* y el núcleo de red de OAI es necesario realizar una serie de configuraciones previas.

Antes de empezar, es necesario haber completado satisfactoriamente los pasos previos que encontramos en la sección B.1.1, donde se describen los pasos para crear una interfaz puente para el correcto flujo de datagramas. Una vez completados se puede proceder con los pasos específicos para preparar el escenario desarrollado para *UERANSIM*.

B.3.0.1. Configuración AMF

En el momento de realizar este proyecto *UERANSIM* no es compatible con los algoritmos de cifrado *NIA0* y *NIA1*. [68]

Por lo tanto, es necesario actualizar la configuración de todos los elementos afectados, en este caso el AMF, así pues debemos dirigirnos al fichero de configuración del despliegue basado en *UERANSIM*, este se puede encontrar en la ruta: *./oai-cn5g-fed/docker-compose/docker-compose-basic-vpp-nrf.yaml*.

Una vez en él, se debe localizar la sección con los parámetros de configuración del AMF y modificarlos, añadiendo las líneas que se muestran a continuación.

```
- INT_ALGO_LIST=[ “NIA1”, “NIA2”]
- CIPH_ALGO_LIST=[“NEA1” , “NEA2”]
```

Tabla B.7: Configuración algoritmos de cifrado para el escenario basado en *UERANSIM*

El resultado final de esa sección del fichero de configuración deberá de ser similar a la que se puede observar en la tabla B.8.

```
oai-amf:
    container_name: “oai-amf”
    image: oai-amf:latest
    environment:
        - TZ=Europe/paris
        - INSTANCE=0
        - PID_DIRECTORY=/var/run
        - MCC=208
        - MNC=95
        - REGION_ID=128
        - AMF_SET_ID=1
        - SERVED_GUAMI_MCC_0=208
        - SERVED_GUAMI_MNC_0=95
        - SERVED_GUAMI_REGION_ID_0=128
        - SERVED_GUAMI_AMF_SET_ID_0=1
        - SERVED_GUAMI_MCC_1=460
        - SERVED_GUAMI_MNC_1=11
        - SERVED_GUAMI_REGION_ID_1=10
        - SERVED_GUAMI_AMF_SET_ID_1=1
```

```
    - PLMN_SUPPORT_MCC=208
    - PLMN_SUPPORT_MNC=95
    - PLMN_SUPPORT_TAC=0xa000
    - SST_0=222
    - SD_0=123
    - SST_1=1
    - SD_1=12
    - AMF_INTERFACE_NAME_FOR_NGAP=eth0
    - AMF_INTERFACE_NAME_FOR_N11=eth0
    - SMF_INSTANCE_ID_0=1          - SMF_FQDN_0=oai-smf
    - SMF_IPV4_ADDR_0=192.168.70.133
    - SMF_HTTP_VERSION_0=v1
    - SELECTED_0=true
    - SMF_INSTANCE_ID_1=2
    - SMF_FQDN_1=oai-smf
    - SMF_IPV4_ADDR_1=0.0.0.0
    - SMF_HTTP_VERSION_1=v1
    - SELECTED_1=false
    - MYSQL_SERVER=192.168.70.131
    - MYSQL_USER=root
    - MYSQL_PASS=linux
    - MYSQL_DB=oai_db
    - OPERATOR_KEY=63bfa50ee6523365ff14c1f45f88737d
    - NRF_IPV4_ADDRESS=192.168.70.130
    - NRF_PORT=80
    - NF_REGISTRATION=yes
    - SMF_SELECTION=yes
    - USE_FQDN_DNS=yes
    - EXTERNAL_AUSF=yes
    - NRF_API_VERSION=v1
    - NRF_FQDN=oai-nrf
    - AUSF_IPV4_ADDRESS=192.168.70.138
    - AUSF_PORT=80
    - AUSF_API_VERSION=v1
    - AUSF_FQDN=oai-ausf
    - INT_ALGO_LIST=[ “NIA1” , “NIA2”]
    - CIPH_ALGO_LIST=[ “NEA1” , “NEA2”]
  depends_on:
    - mysql
```

```
    - vpp-upf
    - oai-ext-dn
    - oai-ausf

volumes:
    - ./amf-healthcheck.sh:/openair-amf/bin/amf-healthcheck.sh
healthcheck:
    test: /bin/bash -c "/openair-amf/bin/amf-healthcheck.sh"
    interval: 10s
    timeout: 15s
    retries: 5
networks:
    public_net:
        ipv4_address: 192.168.70.132
```

Tabla B.8: Configuración algoritmos de cifrado para el escenario basado en *UERANSIM*, resultado final

En primera instancia, según las indicaciones del propio *EURECOM* estas modificaciones son suficientes para comenzar con el desarrollo de un prototipo de red basado en sus componentes de núcleo de red y los elementos proporcionados por *UERANSIM* [66].

Sin embargo, durante la realización de este proyecto se encontraron errores de configuración en los parámetros de red de OAI. Dichos errores son discutidos y solventados en el Anexo C, tal y como se mencionó en la sección previa de este mismo anexo. Por lo tanto, antes de continuar con las pruebas, es imprescindible que se dirija al Anexo C apartado C.2.1.

Anexo C

Obtención de resultados y corrección de errores

C.1. Obtención de resultados

C.1.1. Ejecución de pruebas para escenario basado en *gnbsim*

C.1.1.1. Despliegue de los elementos de red

En primer lugar, para la puesta en marcha de los componentes del núcleo de red, tal como se muestra en la figura C.1

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# python3
./core-network.py --type start-basic
```

Tabla C.1: Comando para lanzar los elementos del núcleo de red proporcionados por OAI

Cuando todos los componentes se hayan lanzado de forma satisfactoria, obtendremos un mensaje como el de la Tabla C.2

[2022-06-15 18:44:36,637[root:DEBUG: All components are healthy, please see below for more details....

NAME	COMMAND	SERVICE	STATUS	PORTS
mysql	"docker-entrypoint.s..."	mysql	vrunning (healthy)	33060/tcp
oai-amf	"/bin/bash /openair-..."	oai-amf	running (healthy)	38412/sctp
oai-ausf	"/bin/bash /openair-..."	oai-ausf	vrunning (healthy)	80/tcp
oai-ext-dn	"/bin/bash -c ' apt ..."	oai-ext-dn	running	
oai-nrf	"/bin/bash /openair-..."	oai-nrf	vrunning (healthy)	v9090/tcp
oai-smf	"/bin/bash /openair-..."	oai-smf	running (healthy)	9090/tcp
oai-spgwu	"/openair-spgwu-tiny..."	oai-spgwu	running (healthy)	8805/udp
oai-udm	"/bin/bash /openair-..."	oai-udm	running (healthy)	80/tcp
oai-udr	v "/bin/bash /openair-..."	oai-udr	running (healthy)	80/tcp

Tabla C.2: Comando para comprobar el estado de los contenedores con los elementos del núcleo de red de OAI

A continuación, estamos listos para lanzar *gnbsim*, para ello se debe ejecutar el comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml up -d gnbsim
```

Tabla C.3: Comando para lanzar "gnbsim"

Cuando se haya completado el proceso, es recomendable comprobar que todos los elementos están debidamente instanciados y funcionales, esto puede hacerse a través del siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# ocker-compose
-f docker-compose-gnbsim.yaml up -d gnbsim
```

Tabla C.4: Comando para comprobar el estado de los contenedores con los elementos de la red

Este comando, nos devolverá una salida, que debe de ser similar a la mostrada a continuación.

CONTAINER ID	IMAGE	STATUS	NAMES
2ad428f94fb0	gnbsim:latest	Up 32 seconds (healthy)	gnbsim
c25db05aa023	ubuntu:bionic	Up 4 minutes	oai-ext-dn
31b6391a3a41	oai-amf:latest	Up 4 minutes (healthy)	oai-amf
753ae61f715f	oai-spgwu-tiny:latest	Up 4 minutes (healthy)	oai-spgwu
84c164ab8136	oai-smf:latest	Up 4 minutes (healthy)	oai-smf
6f0ce91e4efb	oai-nrf:latest	Up 4 minutes (healthy)	oai-nrf
565617169b42	mysql:5.7	Up 4 minutes (healthy)	mysql

Tabla C.5: Resultado de comprobar el estado de los contenedores con los elementos de la red

Si todos los elementos se encuentran en estado "healthy", podemos continuar. El siguiente paso consiste en comprobar que nuestro UE ha obtenido una dirección IP, para ello basta con comprobarlo de la siguiente manera:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim 2>&1 | grep "UE address:"
[gnbsim]2022/06/16 09:27:38.030547 example.go:332: UE address: 12.1.1.2
```

Tabla C.6: Comando para obtener la dirección IP de un UE

Finalmente, añadiremos más terminales de usuario (UEs) para realizar las pruebas de tráfico de red. Para añadir un terminal, debe de introducirse el comando que se muestra en la Tabla C.7

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml up -d gnbsim2
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml up -d gnbsim2
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml up -d gnbsim2
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml up -d gnbsim2
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# sleep 40
```

Tabla C.7: Sentencia para añadir terminales UE al escenario de red

Ahora comprobamos que los nuevos componentes se han añadido satisfactoriamente, para ello debemos lanzar el siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml ps -a
```

Tabla C.8: Comando para comprobar el estado de los contenedores de *gnbsim*

La respuesta del comando anterior, si se han seguido todos los pasos previos, debe de ser similar a la mostrada en la siguiente figura.

NAME	SERVICE	STATUS
gnbsim	gnbsim	running (healthy)
gnbsim2	gnbsim2	running (healthy)
gnbsim3	gnbsim3	running (healthy)
gnbsim4	gnbsim4	running (healthy)
gnbsim5	gnbsim5	running (healthy)
mysql	mysql	running (healthy)
oai-amf	oai-amf	running (healthy)
oai-ausf	oai-ausf	running (healthy)
oai-ext-dn	oai-ext-dn	running
oai-nrf	oai-nrf	running (healthy)
oai-smf	oai-smf	running (healthy)
oai-spgwu	oai-spgwu	running (healthy)
oai-udm	oai-udm	running (healthy)
oai-udr	oai-udr	running (healthy)

Tabla C.9: Respuesta para el comando para comprobar el estado de los contenedores de *gnbsim*

Habiendo obtenido el resultado anterior, se puede proceder a la obtención de las direcciones IP que se han asignado a los UE que hemos añadido para realizar las pruebas de tráfico. El comando adecuado es el que se ha visto anteriormente en la Tabla C.6 y el resultado debe de ser como el que se muestra a continuación.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim2 2>&1 | grep “UE address:”
[gnbsim]2022/06/16 09:33:52.556079 example.go:332: UE address: 12.1.1.3
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
```

```
gnbsim3 2>&1 | grep "UE address:"
[gnbsim]2022/06/16 09:33:57.869161 example.go:332: UE address: 12.1.1.4
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim4 2>&1 | grep "UE address:"
[gnbsim]2022/06/16 09:34:04.558267 example.go:332: UE address: 12.1.1.5
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim5 2>&1 | grep "UE address:"
[gnbsim]2022/06/16 09:34:11.220741 example.go:332: UE address: 12.1.1.6
```

Tabla C.10: Sentencia para obtener las direcciones IP de los UEs conectados a la red

Anotadas las direcciones IP y habiendo comprobado que todos los elementos se están ejecutando de forma correcta, se puede proceder a la realización de las pruebas de tráfico.

C.1.1.2. Pruebas de ping e iperf

Para realizar las pruebas de concepto y la obtención de resultados sobre este despliegue, es necesario seguir los pasos que a continuación se muestran para cada tipo de prueba.

- Pruebas de conectividad ping
  - Desde una red externa simulada por "*oai-ext-dn*" terminal UE 1

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
oai-ext-dn ping -c 3 12.1.1.2
PING 12.1.1.2 (12.1.1.2) 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=63 time=0.742 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=63 time=1.39 ms
64 bytes from 12.1.1.2: icmp_seq=3 ttl=63 time=1.27 ms

---12.1.1.2 ping statistics ---
3 packets transmitted, 3 received, 0 % packet loss, time 2021ms
rtt min/avg/max/mdev = 0.742/1.136/1.391/0.284 ms
```

Tabla C.11: Comando para lanzar un *ping* desde una red externa hacia un terminal UE de la red desplegada

- Desde terminal UE 1 hacia "google.com"

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
gnbsim ping -c 4 -I 12.1.1.2 google.com
PING google.com (142.250.184.14) from 12.1.1.2 : 56(84) bytes of data.
64 bytes from mad41s10-in-f14.1e100.net (142.250.184.14): icmp_seq=1 ttl=114 time=5.70
ms
64 bytes from mad41s10-in-f14.1e100.net (142.250.184.14): icmp_seq=2 ttl=114 time=7.97
ms
64 bytes from mad41s10-in-f14.1e100.net (142.250.184.14): icmp_seq=3 ttl=114 time=7.14
ms
64 bytes from mad41s10-in-f14.1e100.net (142.250.184.14): icmp_seq=4 ttl=114 time=7.56
ms

---google.com ping statistics ---
4 packets transmitted, 4 received, 0 % packet loss, time 3008ms
rtt min/avg/max/mdev = 5.706/7.096/7.971/0.856 ms
```

Tabla C.12: *Ping* desde un terminal UE de la red desplegada hacia "google.com"

- Pruebas de conectividad desde red externa "*oai-ext-dn*" al resto de UEs

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
oai-ext-dn ping -c 2 12.1.1.3
PING 12.1.1.3 (12.1.1.3) 56(84) bytes of data.
64 bytes from 12.1.1.3: icmp_seq=1 ttl=63 time=0.372 ms
64 bytes from 12.1.1.3: icmp_seq=2 ttl=63 time=1.13 ms

---12.1.1.3 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1017ms
rtt min/avg/max/mdev = 0.372/0.754/1.137/0.383 ms


root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
oai-ext-dn ping -c 2 12.1.1.4
PING 12.1.1.4 (12.1.1.4) 56(84) bytes of data.
64 bytes from 12.1.1.4: icmp_seq=1 ttl=63 time=0.661 ms
64 bytes from 12.1.1.4: icmp_seq=2 ttl=63 time=0.456 ms
```

```
---12.1.1.4 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1015ms
rtt min/avg/max/mdev = 0.456/0.558/0.661/0.105 ms


root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
oai-ext-dn ping -c 2 12.1.1.5
PING 12.1.1.5 (12.1.1.5) 56(84) bytes of data.
64 bytes from 12.1.1.5: icmp_seq=1 ttl=63 time=0.541 ms
64 bytes from 12.1.1.5: icmp_seq=2 ttl=63 time=1.30 ms

---12.1.1.5 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1011ms
rtt min/avg/max/mdev = 0.541/0.921/1.302/0.381 ms


root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
oai-ext-dn ping -c 2 12.1.1.6
PING 12.1.1.6 (12.1.1.6) 56(84) bytes of data.
64 bytes from 12.1.1.6: icmp_seq=1 ttl=63 time=0.396 ms
64 bytes from 12.1.1.6: icmp_seq=2 ttl=63 time=1.77 ms

---12.1.1.6 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1021ms
rtt min/avg/max/mdev = 0.396/1.087/1.778/0.691 ms
```

Tabla C.13: Comando para lanzar un *ping* desde red externa hacia el resto de los terminales UE

- Pruebas de conectividad entre terminales UE (Ejemplos)

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
gnbsim ping -c 2 -I 12.1.1.2 12.1.1.3
PING 12.1.1.3 (12.1.1.3) from 12.1.1.2 : 56(84) bytes of data.
64 bytes from 12.1.1.3: icmp_seq=1 ttl=64 time=0.621 ms
64 bytes from 12.1.1.3: icmp_seq=2 ttl=64 time=1.76 ms

---12.1.1.3 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1031ms
rtt min/avg/max/mdev = 0.621/1.192/1.764/0.572 ms

root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
```

```
gnbsim2 ping -c 2 -I 12.1.1.3 12.1.1.4
PING 12.1.1.4 (12.1.1.4) from 12.1.1.3 : 56(84) bytes of data.
64 bytes from 12.1.1.4: icmp_seq=1 ttl=64 time=0.626 ms
64 bytes from 12.1.1.4: icmp_seq=2 ttl=64 time=1.38 ms

---12.1.1.4 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1001ms
rtt min/avg/max/mdev = 0.626/1.005/1.385/0.380 ms

root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
gnbsim3 ping -c 2 -I 12.1.1.4 12.1.1.6
PING 12.1.1.6 (12.1.1.6) from 12.1.1.4 : 56(84) bytes of data.
64 bytes from 12.1.1.6: icmp_seq=1 ttl=64 time=0.611 ms
64 bytes from 12.1.1.6: icmp_seq=2 ttl=64 time=1.89 ms

---12.1.1.6 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1002ms
rtt min/avg/max/mdev = 0.611/1.252/1.893/0.641 ms

root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
gnbsim4 ping -c 2 -I 12.1.1.5 12.1.1.3
PING 12.1.1.3 (12.1.1.3) from 12.1.1.5 : 56(84) bytes of data.
64 bytes from 12.1.1.3: icmp_seq=1 ttl=64 time=0.614 ms
64 bytes from 12.1.1.3: icmp_seq=2 ttl=64 time=1.27 ms

---12.1.1.3 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1014ms
rtt min/avg/max/mdev = 0.614/0.943/1.272/0.329 ms

root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
gnbsim5 ping -c 2 -I 12.1.1.6 12.1.1.2
PING 12.1.1.2 (12.1.1.2) from 12.1.1.6 : 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=64 time=0.628 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=64 time=1.84 ms

---12.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1069ms
rtt min/avg/max/mdev = 0.628/1.236/1.845/0.609 ms
```

Tabla C.14: Mandato para lanzar un *ping* desde un terminal UE hacia otro

- Pruebas de tráfico con *iperf* entre UE y nodo externo simulado

  En primer lugar, establecemos un servidor *iperf* dentro de un contenedor externo, para ello debemos de ejecutar el siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
```

Tabla C.15: Comando para lanzar servidor *ipref* en un contenedor externo

A continuación, abrimos un nuevo terminal en el directorio actual, en este caso será para el cliente (UE 1). Desde el lanzamos la prueba a través del comando mostrado a continuación.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
gnbsim iperf3 -c 192.168.70.135 -B 12.1.1.2
Connecting to host 192.168.70.135, port 5201
[4] local 12.1.1.2 port 48827 connected to 192.168.70.135 port 5201
[ID] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 44.6 MBytes 374 Mbits/sec 285 341 KBytes
[4] 1.00-2.00 sec 47.1 MBytes 395 Mbits/sec 7 339 KBytes
[4] 2.00-3.00 sec 52.9 MBytes 444 Mbits/sec 0 444 KBytes
[4] 3.00-4.00 sec 45.9 MBytes 385 Mbits/sec 0 519 KBytes
[4] 4.00-5.00 sec 35.6 MBytes 299 Mbits/sec 0 568 KBytes
[4] 5.00-6.00 sec 46.1 MBytes 387 Mbits/sec 28 264 KBytes
[4] 6.00-7.00 sec 48.2 MBytes 404 Mbits/sec 45 219 KBytes
[4] 7.00-8.00 sec 42.6 MBytes 358 Mbits/sec 0 337 KBytes
[4] 8.00-9.00 sec 47.2 MBytes 396 Mbits/sec 21 328 KBytes
[4] 9.00-10.00 sec 29.8 MBytes 250 Mbits/sec 148 204 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-10.00 sec 440 MBytes 369 Mbits/sec 534 sender
[4] 0.00-10.00 sec 437 MBytes 367 Mbits/sec receiver

iperf Done.
```

Tabla C.16: Comando para lanzar la prueba *ipref* desde un terminal UE cualquiera

El resultado obtenido del lado del servidor debe de ser similar al que se muestra en la Tabla C.17.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.70.134, port 51035
[5] local 192.168.70.135 port 5201 connected to 192.168.70.134 port 48827
[ID] Interval Transfer Bandwidth
[5] 0.00-1.00 sec 40.9 MBytes 343 Mbits/sec
[5] 1.00-2.00 sec 47.0 MBytes 394 Mbits/sec
[5] 2.00-3.00 sec 53.2 MBytes 446 Mbits/sec
[5] 3.00-4.00 sec 46.2 MBytes 388 Mbits/sec
[5] 4.00-5.00 sec 34.9 MBytes 293 Mbits/sec
[5] 5.00-6.00 sec 46.5 MBytes 390 Mbits/sec
[5] 6.00-7.00 sec 47.8 MBytes 401 Mbits/sec
[5] 7.00-8.00 sec 42.9 MBytes 360 Mbits/sec
[5] 8.00-9.00 sec 47.4 MBytes 397 Mbits/sec
[5] 9.00-10.00 sec 29.7 MBytes 249 Mbits/sec
[5] 10.00-10.02 sec 440 KBytes 201 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ID] Interval Transfer Bandwidth
[5] 0.00-10.02 sec 0.00 Bytes 0.00 bits/sec sender
[5] 0.00-10.02 sec 437 MBytes 366 Mbits/sec receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Tabla C.17: Resultado obtenido de la prueba *ipref*

C.1.1.3. Finalización de los elementos de red

Para poner fin de manera ordenada al despliegue de los elementos de este escenario, es necesario seguir un orden. En primer lugar, deben de cerrarse los contenedores *gnbsim*, para ello se ejecutará el siguiente comando.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-gnbsim.yaml down
```

Tabla C.18: Sentencia para cerrar todos los contenedores *Docker* de *gnbsim* activos

A continuación, se puede proceder con la finalización de todos los contenedores de elementos de red 5G, para ello haremos uso de la siguiente orden.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# python3
./core-network.py –type stop-basic –fqdn no –scenario 1
```

Tabla C.19: Sentencia para cerrar todos los contenedores *Docker* de los elementos del núcleo de red 5G activos

Finalizada la ejecución de estos dos comandos, todos los elementos que componían el escenario de red habrán sido cerrados correctamente.

C.1.1.4. Análisis de ficheros log

C.1.2. Ejecución de pruebas para escenario basado en *UERANSIM*

C.1.2.1. Despliegue de los elementos de red

En primer lugar, para la puesta en marcha de los componentes del núcleo de red, tal como se muestra en la figura C.20

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-basic-vpp-nrf.yaml up -d
```

Tabla C.20: Sentencia para lanzar los contenedores con los elementos del núcleo de red proporcionados por OAI

Comprobamos que todos los componentes se hayan lanzado de forma satisfactoria, a través del comando mostrado en la Tabla C.21

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker ps -a
```

Tabla C.21: Comando para comprobar los contenedores con los elementos del núcleo de red proporcionados por OAI

Si todos los elementos del núcleo de red están instanciados correctamente, obtendremos una salida similar a la mostrada a continuación.

```
[2022-06-15 18:44:36,637[ root:DEBUG: All components are healthy, please see below for
more details....
CONTAINER ID   IMAGE                STATUS                    NAMES
1b7e99850773   oai-smf:latest       Up 13 minutes (healthy)   oai-smf
3a8fba5320f3   oai-amf:latest       Up 13 minutes(healthy)    oai-amf
9ec8487525ae   oai-ausf:latest      Up 13 minutes (healthy)   oai-ausf
6e8e6e7481e8   ubuntu:bionic        Up 13 minutes             oai-ext-dn
1a4668f03e06   oai-udm:latest       Up 13 minutes (healthy)   oai-udm
733fdcb14a73   oai-udr:latest       Up 13 minutes (healthy)   oai-udr
e1e07a50656c   oai-upf-vpp:latest   Up 13 minutes (healthy)   vpp-upf
29dd316c2d8b   mysql:5.7            Up 13 minutes (healthy)   mysql
06b5162e7413   oai-nrf:latest       Up 13 minutes (healthy)   oai-nrf
```

Tabla C.22: Respuesta del comando para lanzar los contenedores con los elementos del núcleo de red proporcionados por OAI

A continuación, estamos listos para lanzar $UERANSIM$, para ello se debe ejecutar el comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-ueransim-vpp.yaml up -d
```

Tabla C.23: Comando para lanzar el contenedor proporcionado por $UERANSIM$

Cuando se haya completado el proceso, es recomendable comprobar que todos los elementos están debidamente instanciados y funcionando. Por lo tanto, se debe de ejecutar de nuevo el comando visto en la Tabla C.21. De la ejecución de este comando, debemos de obtener una respuesta similar a la mostrada a continuación.

root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker ps -a

CONTAINER ID	IMAGE	STATUS	NAMES
fb5b41add0dc	ueransim:latest	Up About a minute (healthy)	ueransim
fe56be6ace8e	oai-smf:latest	Up About a minute (healthy)	oai-smf
f929416df7fd	oai-amf:latest	Up About a minute (healthy)	oai-amf
32b06f7f312c	oai-ausf:latest	Up About a minute (healthy)	oai-ausf
704301d9f947	ubuntu:bionic	Up About a minute	oai-ext-dn
59d6ba01250c	oai-udm:latest	Up About a minute (healthy)	oai-udm
1f9ffe2232a1	oai-udr:latest	Up About a minute (healthy)	oai-udr
4d0273027aed	oai-upf-vpp:latest	Up About a minute (healthy)	vpp-upf
1ff98ba18575	mysql:5.7	Up About a minute (healthy)	mysql
f9076972996a	oai-nrf:latest	Up About a minute (healthy)	oai-nrf

Tabla C.24: Comando para comprobar el estado de los contenedores con los elementos de red

Si todos los elementos se encuentran en estado "healthy", podemos continuar. El siguiente paso consiste en comprobar que nuestro UE ha obtenido una dirección IP, es decir, ha establecido una sesion PDU. Podemos comprobarlo accediendo a los registros de $UERANSIM$ y filtrando los resultados a través del comando "grep", tal como se ve en la siguiente manera:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
ueransim | grep "TUN"
[app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun0, 12.1.1.22] is up.
```

Tabla C.25: Comando para obtener la dirección de un terminal UE de $UERANSIM$

Debido a que por defecto el escenario solo está configurado para instanciar un único UE, debemos de realizar una modificación sobre el fichero de configuración "*docker-compose-ueransim-vpp.yaml*". En primer lugar, debemos detener el contenedor $UERANSIM$, para ello ejecutamos el comando mostrado en la siguiente tabla.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-ueransim-vpp.yaml down
```

Tabla C.26: Comando para detener el contenedor $UERANSIM$

Después de haber detenido el contenedor, y una vez nos encontramos situados en el fichero, se debe localizar el parámetro "-NUMBER_OF_UE", que por defecto tendrá el valor "1". En este caso y para la realización de las pruebas de funcionamiento añadiremos 4 UEs más, por lo tanto, el valor final del parámetro debe ser: "-NUMBER_OF_UE=5". Finalizada la edición del fichero, se guardarán cambios y se volverá a lanzar $UERANSIM$ empleando el mismo comando mostrado en la Tabla C.23

A continuación, una vez se ha lanzado de nuevo $UERANSIM$, comprobamos que todos los UE han establecido una sesión PDU y se les ha asignado una dirección IP válida. Si todo está correcto, se obtendrá un resultado similar al mostrado en la Tabla C.27.

```
root@pfg5g-VirtualBox:/etc/docker/oai-cn5g-fed/docker-compose# docker logs ueransim |
grep -i "tun"
[208950000000035|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun0, 12.1.1.2] is up.
[208950000000034|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun1, 12.1.1.3] is up.
[208950000000031|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun2, 12.1.1.5] is up.
[208950000000033|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun3, 12.1.1.4] is up.
[208950000000032|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun4, 12.1.1.6] is up.
```

Tabla C.27: Comando para obtener las direcciones de los terminales UEs de $UERANSIM$

Tal como se ve en los resultados obtenidos, todos los terminales están operativos con su propia sesión PDU establecida y con una dirección IP única. Adicionalmente se puede consultar el registro del AMF para consultar el estado de los terminales de usuario, para ello se debe lanzar el comando mostrado en la Tabla C.29

```
root@pfg5g-VirtualBox:/etc/docker/oai-cn5g-fed/docker-compose# docker logs oai-amf
```

Tabla C.28: Comando para consultar el registro del AMF

La respuesta del comando anterior deberá arrojar un resultado similar al mostrado en las Tablas C.29 y C.30.

gNBs' information

Index	Status	Global ID	gNB Name	PLMN
1	Connected	0x1	UERANSIM-gnb-208-95-1	208, 95

Tabla C.29: Resultado del registro del AMF I

Tal como se ve en los resultados obtenidos, todos los terminales están operativos con su propia sesión PDU establecida y con una dirección IP única. Adicionalmente se puede consultar el registro del AMF para ver el estado de los terminales de usuario. Para ello se debe lanzar el comando mostrado en la Tabla C.28

UEs' information

Index	5GMM state	IMSI	GUTI	RAN UE NGAP ID	AMF UE ID	PLMN	Cell ID
1	5GMM-REGISTERED	208950000000031		4	4	208, 95	256
2	5GMM-REGISTERED	208950000000032		5	5	208, 95	256
3	5GMM-REGISTERED	208950000000033		3	3	208, 95	256
4	5GMM-REGISTERED	208950000000034		2	2	208, 95	256
5	5GMM-REGISTERED	208950000000035		1	1	208, 95	256

Tabla C.30: Resultado del registro del AMF II

Habiendo realizado todos los pasos hasta este punto y anotadas las direcciones IP junto con el *International Mobile Subscriber Identity* (IMSI) de cada terminal de usuario, nos encontramos en disposición de proceder a la realización de las pruebas de tráfico.

C.1.2.2. Pruebas de ping e iperf

Para realizar las pruebas de concepto y la obtención de resultados sobre este despliegue, es necesario seguir los pasos que a continuación se muestran para cada tipo de prueba.

- Pruebas de conectividad ping
  - Desde una red externa simulada por "*oai-ext-dn*" terminal UE 1

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 3 12.1.1.2
PING 12.1.1.2 (12.1.1.2) 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=63 time=3.13 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=63 time=1.61 ms
64 bytes from 12.1.1.2: icmp_seq=3 ttl=63 time=1.77 ms

— 12.1.1.2 ping statistics —
3 packets transmitted, 3 received, 0 % packet loss, time 2004ms
```

Tabla C.31: *Ping* desde una red externa simulada, hacia un terminal UE de $UERANSIM$

  - Desde terminal UE 1 hacia "google.com"

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec
ueransim ping -c 3 -I uesimtun0 google.com
PING google.com (142.250.184.174) from 12.1.1.2 uesimtun0: 56(84) bytes of data.
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=1 ttl=112
time=5.43 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=2 ttl=112
time=5.56 ms
64 bytes from mad07s23-in-f14.1e100.net (142.250.184.174): icmp_seq=3 ttl=112
time=4.97 ms

— google.com ping statistics —
3 packets transmitted, 3 received, 0 % packet loss, time 2002ms
```

Tabla C.32: Comando para lanzar *ping* desde un terminal UE de $UERANSIM$, hacia "google.com"

- Pruebas de conectividad desde red externa "*oai-ext-dn*" al resto de UE

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 2 12.1.1.3
PING 12.1.1.3 (12.1.1.3) 56(84) bytes of data.
64 bytes from 12.1.1.3: icmp_seq=1 ttl=63 time=4.64 ms
64 bytes from 12.1.1.3: icmp_seq=2 ttl=63 time=1.53 ms

---12.1.1.3 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1001ms
rtt min/avg/max/mdev = 1.531/3.087/4.643/1.556 ms


root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 2 12.1.1.4
PING 12.1.1.4 (12.1.1.4) 56(84) bytes of data.
64 bytes from 12.1.1.4: icmp_seq=1 ttl=63 time=1.42 ms
64 bytes from 12.1.1.4: icmp_seq=2 ttl=63 time=1.41 ms

---12.1.1.4 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1001ms
rtt min/avg/max/mdev = 1.418/1.419/1.421/0.037 ms


root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 2 12.1.1.5
PING 12.1.1.5 (12.1.1.5) 56(84) bytes of data.
64 bytes from 12.1.1.5: icmp_seq=1 ttl=63 time=4.11 ms
64 bytes from 12.1.1.5: icmp_seq=2 ttl=63 time=1.44 ms

---12.1.1.5 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1001ms
rtt min/avg/max/mdev = 1.447/2.780/4.114/1.334 ms


root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 2 12.1.1.6
PING 12.1.1.6 (12.1.1.6) 56(84) bytes of data.
64 bytes from 12.1.1.6: icmp_seq=1 ttl=63 time=2.23 ms
64 bytes from 12.1.1.6: icmp_seq=2 ttl=63 time=1.49 ms
```

```
---12.1.1.6 ping statistics ---
2 packets transmitted, 2 received, 0 % packet loss, time 1001ms
rtt min/avg/max/mdev = 1.499/1.866/2.233/0.367 ms
```

Tabla C.33: Comando para lanzar *ping* desde una red externa simulada, hacia terminales UE de *UERANSIM*

- Pruebas de tráfico con *iperf* entre UE y nodo externo simulado

  En primer lugar, establecemos un servidor *iperf* dentro de un contenedor externo, para ello debemos ejecutar el siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
```

Tabla C.34: Comando para lanzar servidor *iperf* en un contenedor externo

A continuación, abrimos un nuevo terminal en el directorio actual, en este caso será para el cliente (UE 1). Desde el lanzamos la prueba a través del comando mostrado a continuación.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
ueransim iperf3 -c 192.168.73.135 -B 12.1.1.2

Connecting to host 192.168.73.135, port 5201
[ 4] local 12.1.1.2 port 51887 connected to 192.168.73.135 port 5201

[ ID] Interval Transfer Bandwidth Retr Cwnd
[ 4] 0.00-1.00 sec 5.61 MBytes 47.0 Mbits/sec 308 5.27 KBytes
[ 4] 1.00-2.00 sec 1.91 MBytes 16.0 Mbits/sec 245 5.27 KBytes
[ 4] 2.00-3.00 sec 2.41 MBytes 20.2 Mbits/sec 218 6.58 KBytes
[ 4] 3.00-4.00 sec 2.34 MBytes 19.7 Mbits/sec 196 7.90 KBytes
[ 4] 4.00-5.00 sec 2.34 MBytes 19.7 Mbits/sec 223 5.27 KBytes
[ 4] 5.00-6.00 sec 2.16 MBytes 18.1 Mbits/sec 242 51.3 KBytes
[ 4] 6.00-7.00 sec 2.22 MBytes 18.6 Mbits/sec 247 5.27 KBytes
[ 4] 7.00-8.00 sec 2.47 MBytes 20.7 Mbits/sec 230 10.5 KBytes
[ 4] 8.00-9.00 sec 2.41 MBytes 20.2 Mbits/sec 214 14.5 KBytes
[ 4] 9.00-10.00 sec 1.97 MBytes 16.6 Mbits/sec 289 21.1 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
```

```
[ ID] Interval Transfer Bandwidth Retr
[ 4] 0.00-10.00 sec 25.9 MBytes 21.7 Mbits/sec 2412 sender
[ 4] 0.00-10.00 sec 24.8 MBytes 20.8 Mbits/sec receiver
iperf Done.
```

Tabla C.35: Comando para iniciar prueba *iperf* en un terminal UE

El resultado obtenido del lado del servidor debe de ser similar al que se muestra en la Tabla C.36.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 12.1.1.2, port 34629 [ 5] local 192.168.73.135 port 5201
connected to 12.1.1.2 port 51887
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-1.00 sec 4.51 MBytes 37.8 Mbits/sec
[ 5] 1.00-2.00 sec 1.95 MBytes 16.4 Mbits/sec
[ 5] 2.00-3.00 sec 2.42 MBytes 20.3 Mbits/sec
[ 5] 3.00-4.00 sec 2.35 MBytes 19.7 Mbits/sec
[ 5] 4.00-5.00 sec 2.32 MBytes 19.4 Mbits/sec
[ 5] 5.00-6.00 sec 2.10 MBytes 17.6 Mbits/sec
[ 5] 6.00-7.00 sec 2.33 MBytes 19.5 Mbits/sec
[ 5] 7.00-8.00 sec 2.44 MBytes 20.5 Mbits/sec
[ 5] 8.00-9.00 sec 2.39 MBytes 20.1 Mbits/sec
[ 5] 9.00-10.00 sec 1.95 MBytes 16.4 Mbits/sec
[ 5] 10.00-10.01 sec 48.7 KBytes 30.0 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-10.01 sec 0.00 Bytes 0.00 bits/sec sender
[ 5] 0.00-10.01 sec 24.8 MBytes 20.8 Mbits/sec receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Tabla C.36: Resultado prueba *iperf* para terminal de usuario de $UERANSIM$

C.1.2.3. Finalización de los elementos de red

Para poner fin de manera ordenada al despliegue de los elementos de este escenario, es necesario seguir un orden. En primer lugar, debe cerrarse el contenedor *UERANSIM*, para ello se ejecutará la siguiente sentencia.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-ueransim-vpp.yaml down
```

Tabla C.37: Sentencia para cerrar el contenedor *Docker* de *UERANSIM* activo

A continuación, se puede proceder con la finalización de todos los contenedores de elementos de red 5G, para ello haremos uso de la siguiente orden.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# python3
./core-network.py –type stop-basic –fqdn no –scenario 1
```

Tabla C.38: Sentencia para cerrar todos los contenedores *Docker* de los elementos del núcleo de red 5G activos

Finalizada la ejecución de estos dos comandos, todos los elementos que componían el escenario de red habrán sido cerrados correctamente.

C.1.2.4. Análisis de ficheros log

C.1.3. Ejecución de pruebas sobre escenario final: *gnbsim* y *UERANSIM*

C.1.3.1. Despliegue de elementos de red

En primer lugar, para la puesta en marcha de los componentes del núcleo de red se debe de ejecutar el comando que se muestra en la Tabla C.39

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose#
docker-compose -f docker-compose-basic-vpp-nrf.yaml up -d
```

Tabla C.39: Sentencia para lanzar los contenedores con los elementos del núcleo de red proporcionados por OAI

A continuación, comprobamos que todos los componentes se hayan lanzado de forma satisfactoria, a través del comando mostrado en la Tabla C.40

root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker ps -a

Tabla C.40: Comando para comprobar los contenedores con los elementos del núcleo de red proporcionados por OAI

Si todos los elementos del núcleo de red, están instanciados correctamente, obtendremos una salida similar a la mostrada a continuación.

CONTAINER ID	IMAGE	STATUS	NAMES
b0cfb5ee8714	oai-smf:latest	About a minute ago (healthy)	oai-smf
6b3e5abf32f1	oai-amf:latest	About a minute ago(healthy)	oai-amf
69f5fc82887d	oai-ausf:latest	About a minute ago (healthy)	oai-ausf
b67f4ad8dbf1	ubuntu:bionic	About a minute ago	oai-ext-dn
c4d1b392c995	oai-udm:latest	About a minute ago (healthy)	oai-udm
82169edcd6e1	oai-udr:latest	About a minute ago (healthy)	oai-udr
d69fe387a9a5	oai-upf-vpp:latest	About a minute ago (healthy)	vpp-upf
2e5247ad9c99	mysql:5.7	About a minute ago (healthy)	mysql
91656d32a548	oai-nrf:latest	About a minute ago (healthy)	oai-nrf

Tabla C.41: Estado de los componentes del núcleo de red para el escenario final

En este punto, estamos listos para lanzar los componentes de la parte radio (*gnbsim* y $UERANSIM$), para ello se deben ejecutar los comandos vistos en la sección C.1.1.1 Tabla C.3 para los componentes de *gnbsim* y, los comandos vistos en la sección C.1.2.1 Tabla C.23.

Resulta **de vital importancia** lanzar los componentes radio en el orden especificado, ya que, de lo contrario no se garantiza el correcto funcionamiento de estos. Además, se debe tener en consideración que por defecto los contenedores solo están configurados para instanciar un único UE, tanto en $UERANSIM$ como en *gnbsim*.

Si se desea añadir más terminales para la realización de pruebas, es necesario seguir los procedimientos realizados en las secciones dedicadas a cada solución $software$:

- Para $UERANSIM$, seguir el procedimiento visto en el apartado C.1.2.1.
- Para *gnbsim*, seguir el procedimiento realizado en el apartado C.1.1.1 Tabla C.7.

Cuando se haya completado el proceso, es recomendable comprobar que todos los elementos están debidamente instanciados y funcionando. Por lo tanto, se debe de ejecutar de nuevo el comando visto en la Tabla C.40. De la ejecución de este comando, debemos de obtener una respuesta similar a la mostrada a continuación.

CONTAINER ID	IMAGE	STATUS	NAMES
606cef9436f6	ueransim:latest	About a minute ago (healthy)	ueransim
4f784115b220	gnbsim:latest	About a minute ago (healthy)	gnbsim
9256f7e4889e	gnbsim2:latest	About a minute ago (healthy)	gnbsim2
b9eea266d1dd	gnbsim3:latest	About a minute ago (healthy)	gnbsim3
b0cfb5ee8714	oai-smf:latest	About a minute ago (healthy)	oai-smf
6b3e5abf32f1	oai-amf:latest	About a minute ago(healthy)	oai-amf
69f5fc82887d	oai-ausf:latest	About a minute ago (healthy)	oai-ausf
b67f4ad8dbf1	ubuntu:bionic	About a minute ago	oai-ext-dn
c4d1b392c995	oai-udm:latest	About a minute ago (healthy)	oai-udm
82169edcd6e1	oai-udr:latest	About a minute ago (healthy)	oai-udr
d69fe387a9a5	oai-upf-vpp:latest	About a minute ago (healthy)	vpp-upf
2e5247ad9c99	mysql:5.7	About a minute ago (healthy)	mysql
91656d32a548	oai-nrf:latest	About a minute ago (healthy)	oai-nrf

Tabla C.42: Estado de los componentes del escenario final

Si todos los elementos se encuentran en estado “healthy”, podemos continuar. El siguiente paso consiste en comprobar que los UE han obtenido una dirección IP, es decir, ha establecido una sesión PDU. Podemos comprobarlo:

1. Para el caso de los UE de $UERANSIM$, accediendo a los registros y filtrando los resultados a través del comando “grep”, tal como se hizo en el apartado C.1.2.1, Tabla C.25 y,
2. para el caso de los UE de *gnbsim*, accediendo a los registros del contenedor y filtrando con la ayuda del comando “grep”, de la misma forma que se realizó en el apartado C.1.1.1 Tabla C.10.

A continuación, si todo está correcto, anotadas las direcciones IP junto con el IMSI de cada terminal de usuario y, habiendo realizado todos las comprobaciones hasta este punto, nos encontramos en disposición de proceder a la realización de las pruebas de tráfico. Para este ejemplo, las direcciones IPs e IMSIs empleadas han sido las siguientes:

- *UERANSIM*

```
root@pfg5g-VirtualBox:/etc/docker/oai-cn5g-fed/docker-compose# docker logs ueransim |
grep -i "tun"
[208950000000034|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun0, 12.1.1.4] is up.
[208950000000033|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun1, 12.1.1.5] is up.
[208950000000031|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun2, 12.1.1.7] is up.
[208950000000035|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun3, 12.1.1.6] is up.
[208950000000032|app] [info] Connection setup for PDU session[1] is successful, TUN
interface[uesimtun4, 12.1.1.8] is up.
```

Tabla C.43: Direcciones de UEs *UERANSIM*, escenario final

- *gnbsim*

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim 2>&1 | grep "UE address:"
[gnbsim]2022/06/20 10:49:06.240395 example.go:332: UE address: 12.1.1.2
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim2 2>&1 | grep "UE address:"
[gnbsim]2022/06/20 10:51:15.244227 example.go:332: UE address: 12.1.1.3
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker logs
gnbsim3 2>&1 | grep "UE address:"
[gnbsim]2022/06/20 10:51:15.558267 example.go:332: UE address: 12.1.1.9
```

Tabla C.44: Direcciones de UEs *gnbsim*, escenario final

C.1.3.2. Pruebas de ping e iperf

Para realizar las pruebas de concepto y la obtención de resultados sobre este despliegue, es necesario seguir los pasos que a continuación se muestran para cada tipo de prueba.

- Pruebas de conectividad ping
    - Desde una red externa simulada por "*oai-ext-dn*" terminal hacia un UE de $UERANSIM$

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 3 12.1.1.5
PING 12.1.1.5 (12.1.1.5) 56(84) bytes of data.
64 bytes from 12.1.1.5: icmp_seq=1 ttl=63 time=3.13 ms
64 bytes from 12.1.1.5: icmp_seq=2 ttl=63 time=1.61 ms
64 bytes from 12.1.1.5: icmp_seq=3 ttl=63 time=1.77 ms

— 12.1.1.5 ping statistics —
3 packets transmitted, 3 received, 0 % packet loss, time 2004ms
rtt min/avg/max/mdev = 1.616/2.177/3.136/0.681 ms
```

Tabla C.45: *Ping* desde una red externa simulada, hacia un terminal UE de $UERANSIM$

- Pruebas de conectividad ping
    - Desde una red externa simulada por "*oai-ext-dn*" terminal hacia un UE de *gnbsim*

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 3 12.1.1.2
PING 12.1.1.2 (12.1.1.2) 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=63 time=3.13 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=63 time=1.61 ms
64 bytes from 12.1.1.2: icmp_seq=3 ttl=63 time=1.77 ms

— 12.1.1.2 ping statistics —
3 packets transmitted, 3 received, 0 % packet loss, time 2004ms
rtt min/avg/max/mdev = 1.616/2.177/3.136/0.681 ms
```

Tabla C.46: *Ping* desde una red externa simulada, hacia un terminal UE de *gnbsim*

- Desde terminal UE $UERANSIM$ hacia "google.com"

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 3 12.1.1.6
PING 12.1.1.6 ( 12.1.1.6) 56(84) bytes of data.
64 bytes from 12.1.1.6: icmp_seq=1 ttl=63 time=3.13 ms
64 bytes from 12.1.1.6: icmp_seq=2 ttl=63 time=1.61 ms
64 bytes from 12.1.1.6: icmp_seq=3 ttl=63 time=1.77 ms

— 12.1.1.6 ping statistics —
3 packets transmitted, 3 received, 0 % packet loss, time 2004ms
rtt min/avg/max/mdev = 1.616/2.177/3.136/0.681 ms
```

Tabla C.47: *Ping* desde UE $UERANSIM$, hacia "google.com"

- Desde terminal UE *gnbsim* hacia "google.com"

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn ping -c 3 12.1.1.2
PING 12.1.1.2 (12.1.1.2) 56(84) bytes of data.
64 bytes from 12.1.1.2: icmp_seq=1 ttl=63 time=3.13 ms
64 bytes from 12.1.1.2: icmp_seq=2 ttl=63 time=1.61 ms
64 bytes from 12.1.1.2: icmp_seq=3 ttl=63 time=1.77 ms

— 12.1.1.2 ping statistics —
3 packets transmitted, 3 received, 0 % packet loss, time 2004ms
rtt min/avg/max/mdev = 1.616/2.177/3.136/0.681 ms
```

Tabla C.48: *Ping* desde UE *gnbsim*, hacia "google.com"

- Pruebas de tráfico con *iperf* entre UE $UERANSIM$ y nodo externo simulado

  En primer lugar, establecemos un servidor iperf dentro de un contenedor externo, para ello debemos de ejecutar el siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
```

Tabla C.49: Comando para lanzar servidor *iperf* en contenedor externo

A continuación, abrimos un nuevo terminal en el directorio actual, en este caso será para el cliente (UE *UERANSIM*). Desde el lanzamos la prueba a través del comando mostrado a continuación.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
ueransim iperf3 -c 192.168.73.135 -B 12.1.1.5
Connecting to host 192.168.73.135, port 5201
[ 4] local 12.1.1.5 port 51887 connected to 192.168.73.135 port 5201
[ ID] Interval Transfer Bandwidth Retr Cwnd
[ 4] 0.00-1.00 sec 5.61 MBytes 47.0 Mbits/sec 308 5.27 KBytes
[ 4] 1.00-2.00 sec 1.91 MBytes 16.0 Mbits/sec 245 5.27 KBytes
[ 4] 2.00-3.00 sec 2.41 MBytes 20.2 Mbits/sec 218 6.58 KBytes
[ 4] 3.00-4.00 sec 2.34 MBytes 19.7 Mbits/sec 196 7.90 KBytes
[ 4] 4.00-5.00 sec 2.34 MBytes 19.7 Mbits/sec 223 5.27 KBytes
[ 4] 5.00-6.00 sec 2.16 MBytes 18.1 Mbits/sec 242 51.3 KBytes
[ 4] 6.00-7.00 sec 2.22 MBytes 18.6 Mbits/sec 247 5.27 KBytes
[ 4] 7.00-8.00 sec 2.47 MBytes 20.7 Mbits/sec 230 10.5 KBytes
[ 4] 8.00-9.00 sec 2.41 MBytes 20.2 Mbits/sec 214 14.5 KBytes
[ 4] 9.00-10.00 sec 1.97 MBytes 16.6 Mbits/sec 289 21.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval Transfer Bandwidth Retr
[ 4] 0.00-10.00 sec 25.9 MBytes 21.7 Mbits/sec 2412 sender
[ 4] 0.00-10.00 sec 24.8 MBytes 20.8 Mbits/sec receiver
iperf Done.
```

Tabla C.50: Prueba *iperf* desde un terminal UE de *UERANSIM*

El resultado obtenido en el lado del servidor, debe de ser similar al que se muestra en la Tabla C.51.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 12.1.1.5, port 34629 [ 5] local 192.168.73.135 port 5201
connected to 12.1.1.5 port 51887
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-1.00 sec 4.51 MBytes 37.8 Mbits/sec
[ 5] 1.00-2.00 sec 1.95 MBytes 16.4 Mbits/sec
[ 5] 2.00-3.00 sec 2.42 MBytes 20.3 Mbits/sec
[ 5] 3.00-4.00 sec 2.35 MBytes 19.7 Mbits/sec
[ 5] 4.00-5.00 sec 2.32 MBytes 19.4 Mbits/sec
[ 5] 5.00-6.00 sec 2.10 MBytes 17.6 Mbits/sec
[ 5] 6.00-7.00 sec 2.33 MBytes 19.5 Mbits/sec
[ 5] 7.00-8.00 sec 2.44 MBytes 20.5 Mbits/sec
[ 5] 8.00-9.00 sec 2.39 MBytes 20.1 Mbits/sec
[ 5] 9.00-10.00 sec 1.95 MBytes 16.4 Mbits/sec
[ 5] 10.00-10.01 sec 48.7 KBytes 30.0 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-10.01 sec 0.00 Bytes 0.00 bits/sec sender
[ 5] 0.00-10.01 sec 24.8 MBytes 20.8 Mbits/sec receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Tabla C.51: Resultado prueba *iperf* para terminal de usuario de $UERANSIM$

- Pruebas de tráfico con *iperf* entre UE *gnbsim* y nodo externo simulado

  En primer lugar, establecemos un servidor *iperf* dentro de un contenedor externo, para ello debemos de ejecutar el siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
```

Tabla C.52: Comando para lanzar servidor *iperf* en un contenedor externo

A continuación, abrimos un nuevo terminal en el directorio actual, en este caso será para el cliente (UE *gnbsim*). Desde el lanzamos la prueba a través del comando mostrado a continuación.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
ueransim iperf3 -c 192.168.73.135 -B 12.1.1.2
Connecting to host 192.168.73.135, port 5201
[4] local 12.1.1.2 port 48827 connected to 192.168.70.135 port 5201
[ID] Interval Transfer Bandwidth Retr Cwnd
[4] 0.00-1.00 sec 44.6 MBytes 374 Mbits/sec 285 341 KBytes
[4] 1.00-2.00 sec 47.1 MBytes 395 Mbits/sec 7 339 KBytes
[4] 2.00-3.00 sec 52.9 MBytes 444 Mbits/sec 0 444 KBytes
[4] 3.00-4.00 sec 45.9 MBytes 385 Mbits/sec 0 519 KBytes
[4] 4.00-5.00 sec 35.6 MBytes 299 Mbits/sec 0 568 KBytes
[4] 5.00-6.00 sec 46.1 MBytes 387 Mbits/sec 28 264 KBytes
[4] 6.00-7.00 sec 48.2 MBytes 404 Mbits/sec 45 219 KBytes
[4] 7.00-8.00 sec 42.6 MBytes 358 Mbits/sec 0 337 KBytes
[4] 8.00-9.00 sec 47.2 MBytes 396 Mbits/sec 21 328 KBytes
[4] 9.00-10.00 sec 29.8 MBytes 250 Mbits/sec 148 204 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ID] Interval Transfer Bandwidth Retr
[4] 0.00-10.00 sec 440 MBytes 369 Mbits/sec 534 sender
[4] 0.00-10.00 sec 437 MBytes 367 Mbits/sec receiver
iperf Done.
```

Tabla C.53: Prueba *iperf* desde un terminal UE de *gnbsim*

El resultado obtenido del lado del servidor, debe de ser similar al que se muestra en la Tabla C.54.

```
root@pfg-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker exec -it
oai-ext-dn iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 12.1.1.2, port 34629 [5] local 192.168.70.135 port 5201 connected
to 192.168.70.134 port 48827
[ID] Interval Transfer Bandwidth
[5] 0.00-1.00 sec 40.9 MBytes 343 Mbits/sec
[5] 1.00-2.00 sec 47.0 MBytes 394 Mbits/sec
[5] 2.00-3.00 sec 53.2 MBytes 446 Mbits/sec
[5] 3.00-4.00 sec 46.2 MBytes 388 Mbits/sec
[5] 4.00-5.00 sec 34.9 MBytes 293 Mbits/sec
[5] 5.00-6.00 sec 46.5 MBytes 390 Mbits/sec
[5] 6.00-7.00 sec 47.8 MBytes 401 Mbits/sec
[5] 7.00-8.00 sec 42.9 MBytes 360 Mbits/sec
[5] 8.00-9.00 sec 47.4 MBytes 397 Mbits/sec
[5] 9.00-10.00 sec 29.7 MBytes 249 Mbits/sec
[5] 10.00-10.02 sec 440 KBytes 201 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ID] Interval Transfer Bandwidth
[5] 0.00-10.02 sec 0.00 Bytes 0.00 bits/sec sender
[5] 0.00-10.02 sec 437 MBytes 366 Mbits/sec receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Tabla C.54: Resultado prueba *iperf* para terminal de usuario de *gnbsim*

C.1.3.3. Finalización de los elementos de red

Para poner fin de manera ordenada al despliegue de los elementos de este escenario, es necesario seguir un orden específico. En primer lugar, es necesario finalizar los contenedores correspondientes a la parte de la red de acceso radio:

- Para los contenedores de *gnbsim*, se deben seguir las instrucciones dadas en la sección C.1.1.3 tablas C.18 y C.19.
- Para el contenedor de $UERANSIM$, se seguirá la secuencia vista en la sección C.1.2.3 tablas C.37 y C.37.

C.1.4. Análisis de ficheros *log*

C.1.5. Comandos "experimentales" *UERANSIM*

Como se ha mencionado en el capítulo 5, sección 5.1.3, *UERANSIM* pone a nuestra disposición una serie de utilidades accesibles a través de línea de comandos, que nos ayudan a explotar en mayor medida las capacidades del despliegue. Estas funcionalidades están clasificadas como "experimentales", esto quiere decir que aún se encuentran en fase de desarrollo/pruebas, por lo que es posible que, si se intenta replicar en un futuro, se obtengan resultados que difieran de los plasmados en esta memoria. A continuación, se mostrarán las pruebas realizadas durante la evaluación del segundo escenario.

C.1.5.1. Comandos para gNB

En el caso de los gNB, *UERANSIM* cuenta con una funcionalidad denominada *nr-cli*, con la cual podemos abrir una terminal para realizar operaciones específicas de un nodo gNB. Para acceder a ella, el primer paso es localizar el identificador del gNB desde el que queremos actuar, esta información podemos encontrarla en el registro del AMF tal como se muestra en la Tabla C.29. El segundo paso a seguir, una vez obtenido el identificador del gNB, es ubicar en el directorio donde fue clonado *UERANSIM*, en este caso "/home/pfg/Escritorio/UERANSIM". Una vez dentro del directorio, se debe abrir un terminal y ejecutar la sentencia mostrada a continuación.

```
pfgetsist@pfgetsist-VirtualBox:/home/pfg/Escritorio/UERANSIM$ sudo docker exec -it
ueransim ./nr-cli UERANSIM-gnb-208-95-1
```

Tabla C.55: Comando para lanzar un terminal y operar desde un gNB de *UERANSIM*

A continuación, se abrirá un *shell* sobre la que operaremos desde el propio gNB. Para obtener información de los comandos que podemos ejecutar, teclearemos "commands", esto nos arrojará los comandos disponibles.

```
pfgetsist@pfgetsist-VirtualBox:/home/pfg/Escritorio/UERANSIM$ sudo docker exec -it
ueransim ./nr-cli UERANSIM-gnb-208-95-1
[sudo] contraseña para pfg:
$ commands
```

info	Show some information about the gNB
status	Show some status information about the gNB
amf-list	List all AMFs associated with the gNB
amf-info	Show some status information about the given AMF
ue-list	List all UEs associated with the gNB
ue-count	Print the total number of UEs connected the this gNB
ue-release	Request a UE context release for the given UE

Tabla C.56: Comando para operar desde un gNB de *UERANSIM*

Los resultados mostrados a continuación son reflejo de la prueba de cada uno de los comandos y han sido contrastados con la información disponible en los ficheros de configuración que se adjuntan en el ANEXO D para su consulta y verificación.

- Información del gNB

```
---------------------------------------------------------
$ info
    name: UERANSIM-gnb-208-95-1
    nci: 16
    plmn: 208/95
    tac: 40960
    nssai:
        - sst: 0xde
          sd: 0x00007b
    ngap-ip: 192.168.70.141
    gtp-ip: 192.168.72.141
    paging-drx: v128
    ignore-sctp-id: true
---------------------------------------------------------
```

Tabla C.57: Resultado de la ejecución de la sentencia $info$ en el gNB de *UERANSIM*

```
------------------------------------------------------------
$ status
is-ngap-up: true
------------------------------------------------------------
```

Tabla C.58: Resultado de la ejecución de la sentencia *status* en el gNB de *UERANSIM*

- Información del AMF

```
------------------------------------------------------------
$ amf-list
- id: 2
------------------------------------------------------------
```

Tabla C.59: Resultado de la ejecución de la sentencia $amf - list$ en el gNB de *UERANSIM*

```
------------------------------------------------------------
$ amf-info 2
   id: 2
   name: OAI-AMF
   address: 192.168.70.132:38412
   state: CONNECTED
   capacity: 30
   association:
        id: 2
        rx-num: 10
        tx-num: 10
   served-guami:
        - guami:
                plmn: 208/95
                region-id: 128
                set-id: 1
                pointer: 0
            backup-amf:
        - guami:
                plmn: 460/11
                region-id: 10
                set-id: 1
```

```
                pointer: 1
            backup-amf:
    served-plmn:
        - plmn: 208/95
            nssai:
                - sst: 0xde
                  sd: 0x00007b
                - sst: 0x01
                  sd: 0x00000c
------------------------------------------------------------
```

Tabla C.60: Resultado de la ejecución de la sentencia $amf - info$ con id 2, en el gNB de *UERANSIM*

- Información de los UE

```
------------------------------------------------------------
$ ue-list
    - ue-id: 5
      ran-ngap-id: 5
      amf-ngap-id: 5
    - ue-id: 2
      ran-ngap-id: 4
      amf-ngap-id: 4
    - ue-id: 3
      ran-ngap-id: 3
      amf-ngap-id: 3
    - ue-id: 1
      ran-ngap-id: 1
      amf-ngap-id: 1
    - ue-id: 4
      ran-ngap-id: 2
      amf-ngap-id: 2
------------------------------------------------------------
```

Tabla C.61: Resultado de la ejecución de la sentencia $ue - list$ en el gNB de *UERANSIM*

```
------------------------------------------------------------
$ ue-count
5
------------------------------------------------------------
```

Tabla C.62: Resultado de la ejecución de la sentencia *ue − count* en el gNB de *UERANSIM*

```
------------------------------------------------------------
$ ue-release 5
Requesting UE context release
------------------------------------------------------------
$ ue-list
   - ue-id: 2
     ran-ngap-id: 4
     amf-ngap-id: 4
   - ue-id: 3
     ran-ngap-id: 3
     amf-ngap-id: 3
   - ue-id: 1
     ran-ngap-id: 1
     amf-ngap-id: 1
   - ue-id: 4
     ran-ngap-id: 2
     amf-ngap-id: 2
------------------------------------------------------------
```

Tabla C.63: Resultado de la ejecución de la sentencia *ue − release* con id 5, en el gNB de *UERANSIM*

C.1.5.2. Comandos para UE

Para los UE, *UERANSIM* cuenta con dos funcionalidades, denominadas *nr-cli* y *nr-bri*, con las cuales podemos abrir una terminal para realizar operaciones específicas de un nodo UE. Para acceder a ella, el primer paso es localizar el IMSI del UE desde el que queremos actuar, esta información podemos encontrarla en el registro del AMF tal como se muestra en la Tabla C.30. El segundo paso a seguir, una vez obtenido el IMSI, es ubicar en el directorio donde fue clonado *UERANSIM*, en este caso "/home/pfg/Escritorio/UERANSIM". Una vez dentro del directorio, se debe abrir un terminal y ejecutar la sentencia mostrada a continuación.

C.1.5.2.1. Pruebas mediante *nr-cli*

```
pfgetsist@pfgetsist-VirtualBox:/home/pfg/Escritorio/UERANSIM$ sudo docker exec -it
ueransim ./nr-cli imsi-208950000000031
```

Tabla C.64: Comando para lanzar un terminal desde un UE de *UERANSIM*

A continuación, se abrirá un *shell* sobre la que operaremos desde el propio UE seleccionado. Para obtener información de los comandos que podemos ejecutar, teclearemos "commands", esto nos arrojará los comandos disponibles.

```
pfgetsist@pfgetsist-VirtualBox:/home/pfg/Escritorio/UERANSIM$ sudo docker exec -it
ueransim ./nr-cli imsi-208950000000031
imsi-208950000000031
[sudo] contraseña para pfgetsist:
------------------------------------------------------------
$ commands
  info            Show some information about the UE
  status          Show some status information about the UE
  timers          Dump current status of the timers in the UE
  rls-state       Show status information about RLS
  coverage        Dump available cells and PLMNs in the coverage
  ps-establish    Trigger a PDU session establishment procedure
  ps-list         List all PDU sessions
  ps-release      Trigger a PDU session release procedure
  ps-release-all  Trigger PDU session release procedures for all active sessions
  deregister      Perform a de-registration by the UE
------------------------------------------------------------
```

Tabla C.65: Resultado de la ejecución de la sentencia *commands* en un UE de *UERANSIM*

Una vez más, los resultados mostrados son reflejo de la prueba de cada uno de los comandos y han sido contrastados con la información disponible en los ficheros de configuración que se adjuntan en el ANEXO D para su consulta y verificación. De entre todos los comandos, destacamos los que se muestran a continuación.

```
------------------------------------------------------------
$ info
    supi: imsi-208950000000031
    hplmn: 208/95
    imei: 356938035643803
    imeisv: 0035609204079514
    ecall-only: false
    uac-aic:
        mps: false
        mcs: false
    uac-acc:
        normal-class: 0
        class-11: false
        class-12: false
        class-13: false
        class-14: false
        class-15: false
    is-high-priority: false
------------------------------------------------------------
```

Tabla C.66: Comando para obtener la información de un UE

```
------------------------------------------------------------
$ status
    cm-state: CM-CONNECTED
    rm-state: RM-REGISTERED
    mm-state: MM-REGISTERED/NORMAL-SERVICE
    5u-state: 5U1-UPDATED
    sim-inserted: true
    selected-plmn: 208/95
    current-cell: 1
    current-plmn: 208/95
    current-tac: 40960
    last-tai: PLMN[208/95] TAC[40960]
    stored-suci: no-identity
    stored-guti:
        plmn: 208/95
        amf-region-id: 0x80
```

```
        amf-set-id: 1
        amf-pointer: 1
        tmsi: 0x00000008
    has-emergency: false
------------------------------------------------------------
```

Tabla C.67: Comando para obtener la información de estado de un UE

```
------------------------------------------------------------
$ rls-state
sti: 7FD8EA3E9091DE18
gnb-search-space:
- 192.168.70.141
------------------------------------------------------------
```

Tabla C.68: Comando para obtener la información de estado de un UE II

```
------------------------------------------------------------
$ coverage
[1]:
    signal: -1 dBm (Excellent)
    mib:
        barred: false
        intra-freq-reselection: allowed
    sib1:
        nr-cell-id: 0000000000000010
        plmn: 208/95
        tac: 40960
        operator-reserved: false
------------------------------------------------------------
```

Tabla C.69: Comando para mostrar el estado del enlace radio de un UE

```
$ ps-list
PDU Session1:
    state: PS-ACTIVE
    session-type: IPv4
    apn: default
```

```
    s-nssai:
        sst: 0xde
        sd: 0x00007b
    emergency: false
    address: 12.1.1.9
    ambr: up[20Mb/s] down[22Mb/s]
    data-pending: false
-------------------------------------------------------------
```

Tabla C.70: Comando para obtener información de las sesiones PDU de un UE

Uno de los comandos que puede resultar muy interesantes es el comando *deregister* el cual nos permite simular distintas situaciones:

- Desregistro normal
- Desregistro por apagado del terminal
- Desregistro por pérdida de 5G
- Desregistro por extracción de *Subscriber Identity/Identification Module* (SIM)

A continuación, se ejemplifica el caso de un procedimiento de desregistro "normal".

```
pfgetsist@pfgetsist-VirtualBox:/home/pfg/Escritorio/UERANSIM$ sudo docker exec -it
ueransim ./nr-cli imsi-208950000000033
[sudo] contraseña para pfgetsist:
-------------------------------------------------------------
$ commands
  info              Show some information about the UE
  status            Show some status information about the UE
  timers            Dump current status of the timers in the UE
  rls-state         Show status information about RLS
  coverage          Dump available cells and PLMNs in the coverage
  ps-establish      Trigger a PDU session establishment procedure
  ps-list           List all PDU sessions
  ps-release        Trigger a PDU session release procedure
  ps-release-all    Trigger PDU session release procedures for all active sessions
  deregister        Perform a de-registration by the UE
-------------------------------------------------------------
$ deregister
Perform a de-registration by the UE
```

```
Usage:
  deregister <normal|disable-5g|switch-off|remove-sim>
-------------------------------------------------------------
$ deregister normal
De-registration procedure triggered
-------------------------------------------------------------
```

Tabla C.71: Comando para iniciar el proceso de desregistro de un UE

Comprobamos el estado del UE en varias ocasiones ya que el proceso no es instantáneo.

```
-------------------------------------------------------------
$ status
   cm-state: CM-CONNECTED
   rm-state: RM-REGISTERED
   mm-state: MM-DEREGISTER-INITIATED
   5u-state: 5U1-UPDATED
   sim-inserted: true
   selected-plmn: 208/95
   current-cell: 1
   current-plmn: 208/95
   current-tac: 40960
   last-tai: PLMN[208/95] TAC[40960]
   stored-suci: no-identity
   stored-guti:
        plmn: 208/95
        amf-region-id: 0x80
        amf-set-id: 1
        amf-pointer: 1
        tmsi: 0x00000002
   has-emergency: false
-------------------------------------------------------------
```

Tabla C.72: Evolución comando para iniciar el proceso de desregistro de un UE

```
-------------------------------------------------------------
   $ status
   cm-state: CM-IDLE
   rm-state: RM-DEREGISTERED
```

```
    mm-state: MM-DEREGISTERED/ATTEMPTING-REGISTRATION
    5u-state: 5U2-NOT-UPDATED
    sim-inserted: true
    selected-plmn: 208/95
    current-cell: 1
    current-plmn: 208/95
    current-tac: 40960
    last-tai: PLMN[208/95] TAC[40960]
    stored-suci: no-identity
    stored-guti:
      plmn: 208/95
      amf-region-id: 0x80
      amf-set-id: 1
      amf-pointer: 1
      tmsi: 0x00000002
    has-emergency: false
------------------------------------------------------------
```

Tabla C.73: Evolución comando para iniciar el proceso de desregistro de un UE II

```
------------------------------------------------------------
    $ status
    cm-state: CM-IDLE
    rm-state: RM-DEREGISTERED
    mm-state: MM-DEREGISTERED/NO-SUPI
    5u-state: 5U3-ROAMING-NOT-ALLOWED
    sim-inserted: false
    selected-plmn: 208/95
    current-cell: 1
    current-plmn: 208/95
    current-tac: 40960
    last-tai: null
    stored-suci: no-identity
    stored-guti: no-identity
    has-emergency: false
------------------------------------------------------------
```

Tabla C.74: Evolución comando para iniciar el proceso de desregistro de un UE III

Si obtenemos los registros del AMF tal como hicimos con la Tabla C.29 y C.29, observaremos

el siguiente resultado:

|----------gNBs' information----------|

Index	Status	Global ID	gNB Name	PLMN
1	Connected	0x1	UERANSIM-gnb-208-95-1	208, 95

|----------UEs' information ----------|

Index	5GMM state	IMSI	GUTI	RAN UE NGAP ID	AMF UE ID	PLMN	Cell ID
1	5GMM-REGISTERED	208950000000031		5	5	208, 95	256
2	5GMM-REGISTERED	208950000000032		1	1	208, 95	256
3	5GMM-DEREGISTERED	208950000000033		2	2	208, 95	256
4	5GMM-REGISTERED	208950000000034		4	4	208, 95	256
5	5GMM-REGISTERED	208950000000035		3	3	208, 95	256

|----------|

Tabla C.75: Información del AMF acerca del UE desregistrado

Además, si intentamos hacer ping sobre la dirección IP del terminal que hemos desregistrado, comprobaremos que este ya no tiene conectividad.

```
root@pfgetsist-VirtualBox:/home/pfg/Escritorio/oai-cn5g-fed/docker-compose# docker
exec -it oai-ext-dn ping -c 3 12.2.1.3
PING 12.2.1.3 (12.2.1.3) 56(84) bytes of data.

---12.2.1.3 ping statistics ---
3 packets transmitted, 0 received, 100 % packet loss, time 2047ms
```

Tabla C.76: Intento de *ping* sobre terminal desregistrado

C.1.5.2.2. Pruebas mediante *nr-bri* Los desarrolladores de *UERANSIM* también proporcionan herramientas para aprovechar las sesiones PDU establecidas para un determinado UE. Concretamente nos permiten vincular el tráfico de cualquier aplicación de nuestro sistema operativo huésped, a través de la conexión de un terminal UE. Es por ello por lo que para este desarrollo se ha decidido simular la navegación web, mediante dos pruebas:

1. Obtención de página web a través de la conexión del UE
2. Simular navegación en tiempo real a través de la conexión de un UE y el navegador *Firefox*

En el primer caso para obtener una página web empleando la conexión de un terminal, se deben de ejecutar el siguiente comando.

```
pfgetsist@pfgetsist-VirtualBox:/etc/docker/oai-cn5g-fed/docker-
compose/UERANSIM/tools$ ./nr-binder 12.1.1.2 curl https://www.upm.es
```

Tabla C.77: Comando para obtener la página web de la UPM en formato *Hypertext Markup Language* (HTML), a través de la conexión del UE con IP: 12.1.1.2

La respuesta se mostrará en la misma pantalla y consistirá en el código HTML de la página web en cuestión, es este caso la página web de la Univesidad Politécnica de Madrid (UPM).

En el segundo caso de estudio, para vincular todo el tráfico de una aplicación con un terminal UE concreto, debe de lanzarse el siguiente mandato.

```
fgetsist@pfgetsist-VirtualBox:/etc/docker/oai-cn5g-fed/docker-
compose/UERANSIM/tools$ ./nr-binder 12.1.1.2 firefox
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
Missing chrome or resource URL: resource://gre/modules/UpdateListener.jsm
```

```
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
ERROR: ld.so: object './libdevbnd.so' from LD_PRELOAD cannot be preloaded (cannot
open shared object file): ignored.
```

Tabla C.78: Comando para lanzar navegador $Firefox$ sobre la conexión a internet del UE con IP: 12.1.1.2

Acto seguido, se abrirá una instancia del navegador *Firefox* y todo el tráfico generado con él, será canalizado sobre la conexión del UE con dirección IP: 12.1.1.2. A efectos prácticos, esta situación sería similar a estar navegando desde el terminal de usuario.

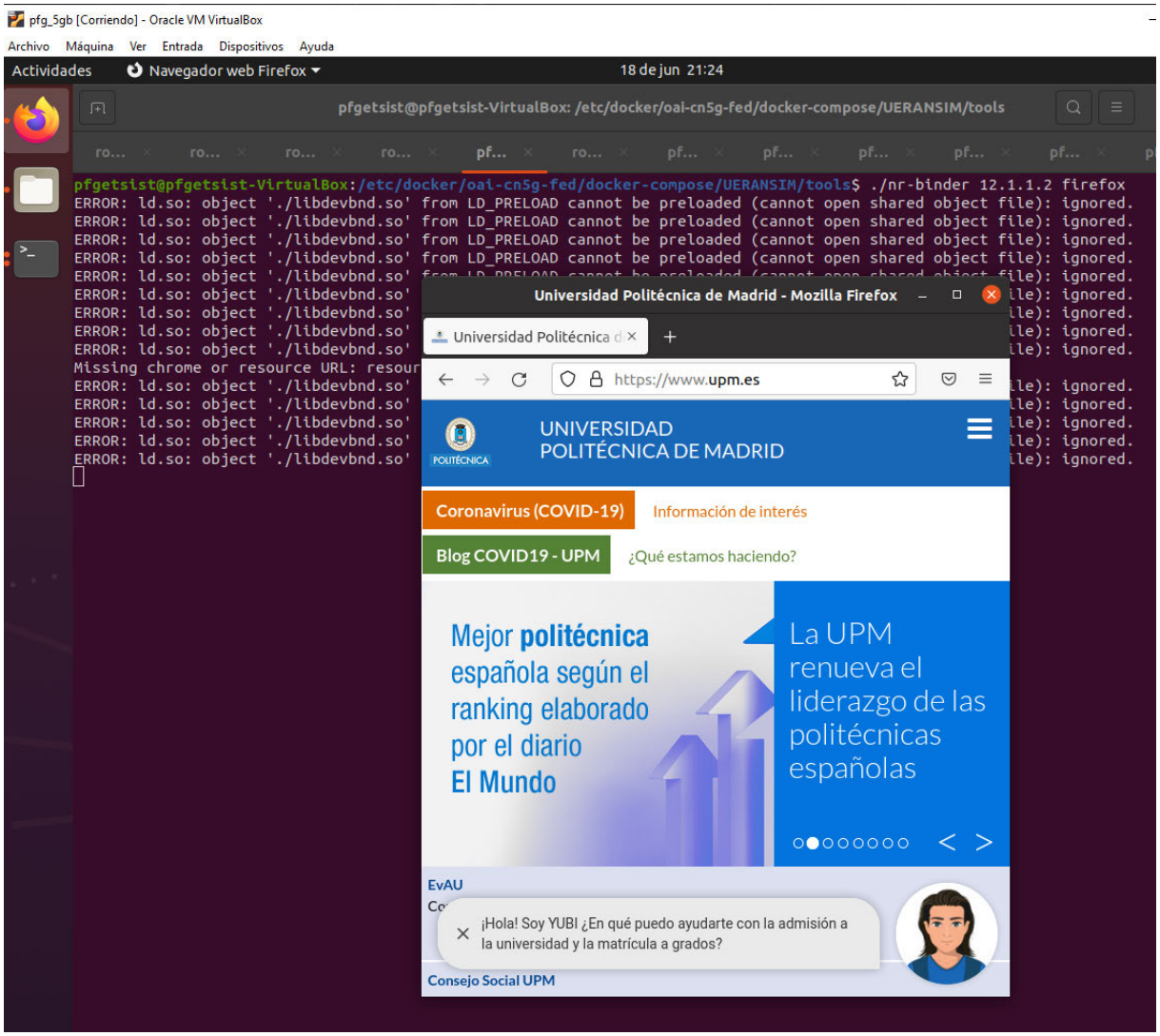


Figura C.1. Experimento para canalizar tráfico del navegador *Firefox* a través de un terminal de usuario

C.1.5.2.3. Análisis de ficheros log

C.2. Corrección de errores

C.2.1. Error configuración de red para *UERANSIM*

Si se intenta ejecutar el escenario basado en $UERANSIM$ puede observarse que se produce un fallo. Este fallo se manifiesta después de lanzar los elementos del escenario, tal como se puede observar en la figura, C.79.

oai-cn5g-fed/docker-compose$ docker ps -a

CONTAINER ID	IMAGE	STATUS	NAMES
4349e8808902	oai-smf:latest	Up 48 seconds (healthy)	oai-smf
62e774768482	oai-amf:latest	Up 48 seconds (healthy)	oai-amf
0302e6a3d2b3	oai-ausf:latest	Up 49 seconds (healthy)	oai-ausf
fb3249a5ade7	ubuntu:bionic	Up 49 seconds	oai-ext-dn
4f114039c218	oai-udm:latest	Up 49 seconds (healthy)	oai-udm
c0838aff8796	**oai-udr:latest**	**Up 50 seconds (unhealthy)**	**oai-udr**
99ab1b23862c	oai-upf-vpp:latest	Up 50 seconds (healthy)	vpp-upf
3469f853e26d	mysql:5.7	Up 51 seconds (healthy)	mysql
ab06bd3104ef	oai-nrf:latest	Up 51 seconds (healthy)	oai-nrf

Tabla C.79: Resultado de la ejecución del escenario basado en *UERANSIM*

Como se puede observar, el UDR nunca llega a arrancar correctamente, por lo que termina en el estado *unhealthy*, si realizamos una captura durante el arranque del escenario para obtener más información acerca de este problema, se puede observar la siguiente secuencia de intercambio de paquetes.

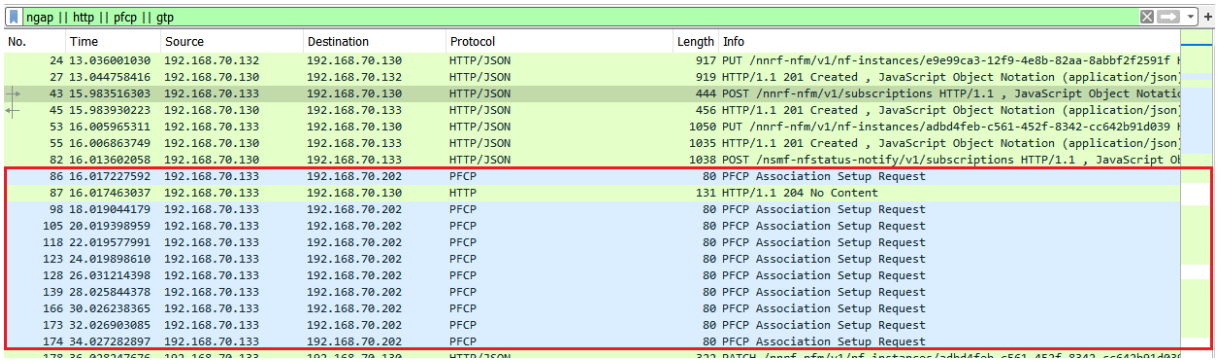


Figura C.2. Captura realizada durante el arranque del núcleo de red, PFCP Association Setup Request: sin respuesta

Como podemos ver, la IP 192.168.70.133, que se corresponde con el SMF, intenta establecer una asociación *Packet Forwarding Control Protocol* (PFCP) con la IP 192.168.70.202, la cual se corresponde con el UPF, sin embargo, vemos que no obtiene éxito en la operación.
Si volvemos atrás y observamos la configuración llevada a cabo en el la sección B.1.1, concretamente en la Tabla B.2, siguiendo las recomendaciones de OAI, debía establecerse la subred "192.168.70.128/26", esta venía configurada por defecto [67]. Ahora bien, si ponemos el foco de atención en la máscara de subred, se trata de una **/26 (255.255.255.192)**. Realizando un cálculo sencillo, obtenemos que, con esta máscara de subred, la última dirección disponible es la "192.168.70.191".

Por lo tanto, durante el proceso de establecimiento de asociación PFCP, es imposible que el SMF pueda alcanzar la dirección "192.168.70.202" (UPF) puesto que se encuentra fuera de la subred "192.168.70.128/26".
Es por ello por lo que, para solucionar este problema con el menor impacto posible, se debe ampliar la máscara de subred. En este caso con una máscara **/24 (255.255.255.0)** es suficiente, ya que así la última dirección IP de la subred sería la "192.168.70.255". Según la decisión que se tomase durante la configuración en la sección B.1.1 del Anexo B, deberemos de corregir el problema de formas diferentes, tal y como se muestra a continuación.

C.2.1.1. Configuración manual

En este caso el fichero de configuración de *docker-compose* que se encuentra en el siguiente directorio: *./oai-cn5g-fed/docker-compose/docker-compose-basic-nrf.yaml* debe mantenerse como se muestra en la Tabla C.80.

```
networks:
    # public_net:
        # external:
        # name: demo-oai-public-net
    public_net:
        driver: bridge
        name: demo-oai-public-net
        ipam:
            config:
                - subnet: 192.168.70.128/26
        driver_opts:
            com.docker.network.bridge.name: "demo-oai"
```

Tabla C.80: Corrección fichero de configuración para desarrollo con *UERANSIM*

En caso de haberse completado la creación del puente a través del comando mostrado en el Anexo B apartado B.2.1, Tabla B.3, sera necesario revertirlo a través del siguiente comando:

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker network remove
demo-oai-public-net
```

Tabla C.81: Corrección de configuración para desarrollo con *UERANSIM*

Posteriormente, se repetirá el proceso del Anexo B, apartado B.2.1, Tabla B.3 pero añadiendo

las siguientes correcciones.

```
root@pfg-VirtualBox:/home/pfg/oai-cn5g-fed/docker-compose# docker network create \
    --driver=bridge \
    --subnet=192.168.70.128/24 \
     -o “com.docker.network.bridge.name”=“demo-oai” \
    demo-oai-public-net
    ac32619cefa951e8145b7e982ee43b97e94edee01b27qdtab4d0a68169a2e161
```

Tabla C.82: Configuración del puente *demo-oai* en la máquina virtual

C.2.1.2. Configuración automática

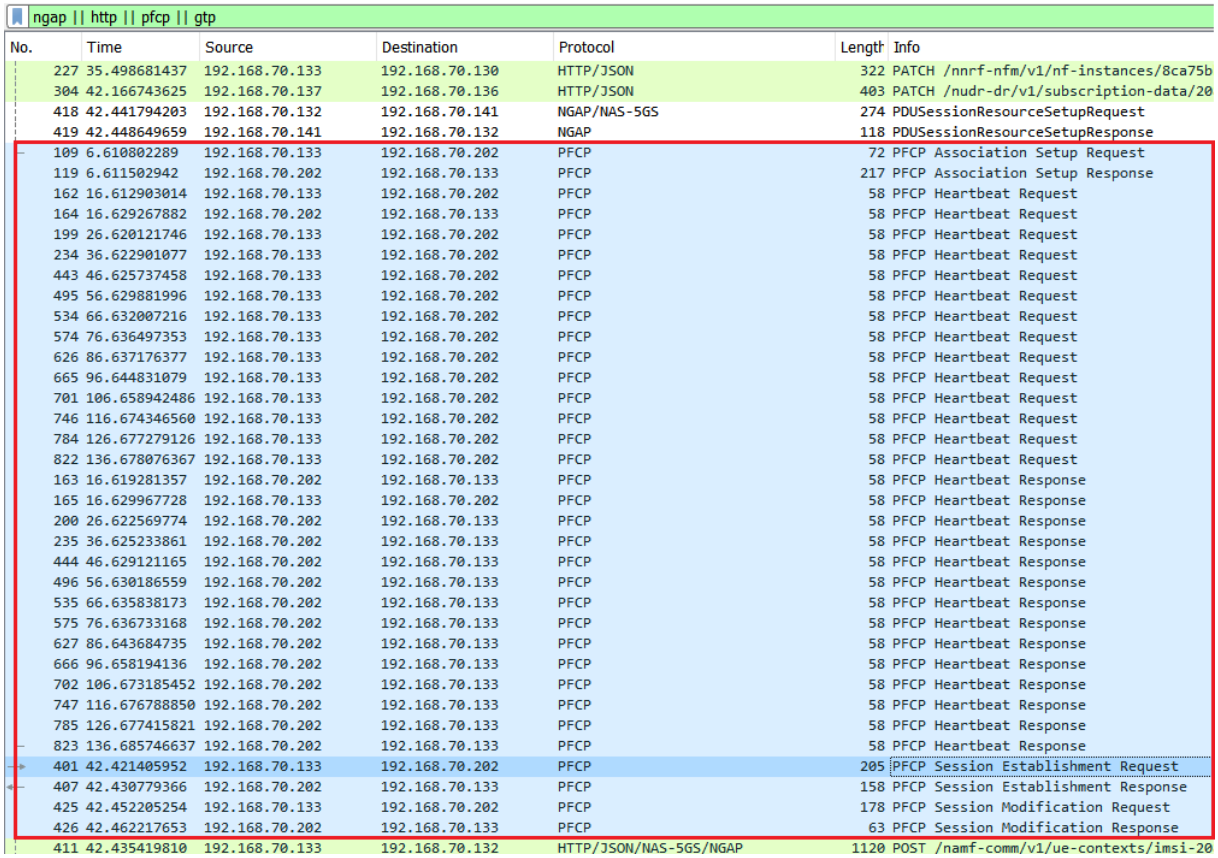
Si se optó por la configuración automática, se debe editar el fichero de configuración de *docker-compose* que se encuentra en el directorio: *./oai-cn5g-fed/docker-compose/docker-compose-basic-nrf.yaml* aplicando los cambios que se muestran en la Tabla C.83.

```
networks:
    # public_net:
        # external:
            # name: demo-oai-public-net
    public_net:
        driver: bridge
        name: demo-oai-public-net
        ipam:
            config:
                - subnet: 192.168.70.128/24
        driver_opts:
            com.docker.network.bridge.name: “demo-oai”
```

Tabla C.83: Configuración del fichero docker-compose-basic-nrf.yaml

C.2.1.3. Resultados de la corrección

Después de efectuar estos cambios, se puede observar que el despliegue funciona correctamente. Si volvemos a capturar el tráfico, se puede ver como se establece la comunicación y la sesión PFCP satisfactoriamente entre el SMF y el UPF.



ngap || http || pfcp || gtp

No.	Time	Source	Destination	Protocol	Length	Info
227	35.498681437	192.168.70.133	192.168.70.130	HTTP/JSON	322	PATCH /nnrf-nfm/v1/nf-instances/8ca75b
304	42.166743625	192.168.70.137	192.168.70.136	HTTP/JSON	403	PATCH /nudr-dr/v1/subscription-data/20
418	42.441794203	192.168.70.132	192.168.70.141	NGAP/NAS-5GS	274	PDUSessionResourceSetupRequest
419	42.448649659	192.168.70.141	192.168.70.132	NGAP	118	PDUSessionResourceSetupResponse
109	6.610802289	192.168.70.133	192.168.70.202	PFCP	72	PFCP Association Setup Request
119	6.611502942	192.168.70.202	192.168.70.133	PFCP	217	PFCP Association Setup Response
162	16.612903014	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
164	16.629267882	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Request
199	26.620121746	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
234	36.622901077	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
443	46.625737458	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
495	56.629881996	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
534	66.632007216	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
574	76.636497353	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
626	86.637176377	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
665	96.644831079	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
701	106.658942486	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
746	116.674346560	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
784	126.677279126	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
822	136.678076367	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Request
163	16.619281357	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
165	16.629967728	192.168.70.133	192.168.70.202	PFCP	58	PFCP Heartbeat Response
200	26.622569774	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
235	36.625233861	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
444	46.629121165	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
496	56.630186559	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
535	66.635838173	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
575	76.636733168	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
627	86.643684735	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
666	96.658194136	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
702	106.673185452	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
747	116.676788850	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
785	126.677415821	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
823	136.685746637	192.168.70.202	192.168.70.133	PFCP	58	PFCP Heartbeat Response
401	42.421405952	192.168.70.133	192.168.70.202	PFCP	205	PFCP Session Establishment Request
407	42.430779366	192.168.70.202	192.168.70.133	PFCP	158	PFCP Session Establishment Response
425	42.452205254	192.168.70.133	192.168.70.202	PFCP	178	PFCP Session Modification Request
426	42.462217653	192.168.70.202	192.168.70.133	PFCP	63	PFCP Session Modification Response
411	42.435419810	192.168.70.133	192.168.70.132	HTTP/JSON/NAS-5GS/NGAP	1120	POST /namf-comm/v1/ue-contexts/imsi-20

Figura C.3. Captura realizada durante el arranque del núcleo de red, PFCP association setup request: error solucionado

Anexo D

Ficheros de configuración

D.1. Fichero *docker-compose-basic-vpp-nrf.yaml*

A continuación, en la figura D.1 se muestra el fichero de configuración con extensión ".yaml" empleado durante el desarrollo del todos los escenarios. Este fichero contiene las instrucciones y parámetros necesarios para el despliegue de todos los elementos del núcleo de red 5G proporcionado por OAI.

```
version: '3.8'
services:
  mysql:
    container_name: "mysql"
    image: mysql:5.7
    volumes:
      - ./oai_db2.sql:/docker-entrypoint-initdb.d/oai_db.sql
      - ./mysql-healthcheck2.sh:/tmp/mysql-healthcheck.sh
    environment:
      - TZ=Europe/Paris
      - MYSQL_DATABASE=oai_db
      - MYSQL_USER=test
      - MYSQL_PASSWORD=test
      - MYSQL_ROOT_PASSWORD=linux
    healthcheck:
      test: /bin/bash -c "/tmp/mysql-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      public_net:
        ipv4_address: 192.168.70.131
  oai-udr:
    container_name: "oai-udr"
    image: oai-udr:latest
    environment:
      - TZ=Europe/Paris
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
      - UDR_INTERFACE_NAME_FOR_NUDR=eth0
      - UDR_INTERFACE_PORT_FOR_NUDR=80
```

```
      - UDR_INTERFACE_HTTP2_PORT_FOR_NUDR=8080
      - UDR_API_VERSION=v1
      - MYSQL_IPV4_ADDRESS=192.168.70.131
      - MYSQL_USER=test
      - MYSQL_PASS=test
      - MYSQL_DB=oai_db
      - WAIT_MYSQL=120
    depends_on:
      - mysql
      - oai-nrf
    networks:
      public_net:
        ipv4_address: 192.168.70.136
    volumes:
      - ./udr-healthcheck.sh:/openair-udr/bin/udr-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-udr/bin/udr-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
  oai-udm:
    container_name: "oai-udm"
    image: oai-udm:latest
    environment:
      - TZ=Europe/Paris
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
      - UDM_NAME=OAI_UDM
      - SBI_IF_NAME=eth0
      - SBI_PORT=80
      - UDM_VERSION_NB=v1
```

```
      - USE_FQDN_DNS=yes
      - UDR_IP_ADDRESS=192.168.70.136
      - UDR_PORT=80
      - UDR_VERSION_NB=v1
      - UDR_FQDN=oai-udr
    depends_on:
      - oai-udr
    networks:
      public_net:
        ipv4_address: 192.168.70.137
    volumes:
      - ./udm-healthcheck.sh:/openair-udm/bin/udm-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-udm/bin/udm-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
  oai-ausf:
    container_name: "oai-ausf"
    image: oai-ausf:latest
    environment:
      - TZ=Europe/Paris
      - INSTANCE_ID=0
      - PID_DIR=/var/run
      - AUSF_NAME=OAI_AUSF
      - SBI_IF_NAME=eth0
      - SBI_PORT=80
      - USE_FQDN_DNS=yes
      - UDM_IP_ADDRESS=192.168.70.137
      - UDM_PORT=80
      - UDM_VERSION_NB=v1
```

```
      - UDM_FQDN=oai-udm
    depends_on:
      - oai-udm
    networks:
      public_net:
        ipv4_address: 192.168.70.138
    volumes:
      - ./ausf-healthcheck.sh:/openair-ausf/bin/ausf-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-ausf/bin/ausf-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
  oai-nrf:
    container_name: "oai-nrf"
    image: oai-nrf:latest
    environment:
      - NRF_INTERFACE_NAME_FOR_SBI=eth0
      - NRF_INTERFACE_PORT_FOR_SBI=80
      - NRF_INTERFACE_HTTP2_PORT_FOR_SBI=9090
      - NRF_API_VERSION=v1
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
    networks:
      public_net:
        ipv4_address: 192.168.70.130
    volumes:
      - ./nrf-healthcheck.sh:/openair-nrf/bin/nrf-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-nrf/bin/nrf-healthcheck.sh"
      interval: 10s
```

```
      timeout: 5s
      retries: 5
  oai-amf:
    container_name: "oai-amf"
    image: oai-amf:latest
    environment:
      - TZ=Europe/paris
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
      - MCC=208
      - MNC=95
      - REGION_ID=128
      - AMF_SET_ID=1
      - SERVED_GUAMI_MCC_0=208
      - SERVED_GUAMI_MNC_0=95
      - SERVED_GUAMI_REGION_ID_0=128
      - SERVED_GUAMI_AMF_SET_ID_0=1
      - SERVED_GUAMI_MCC_1=460
      - SERVED_GUAMI_MNC_1=11
      - SERVED_GUAMI_REGION_ID_1=10
      - SERVED_GUAMI_AMF_SET_ID_1=1
      - PLMN_SUPPORT_MCC=208
      - PLMN_SUPPORT_MNC=95
      - PLMN_SUPPORT_TAC=0xa000
      - SST_0=222
      - SD_0=123
      - SST_1=1
      - SD_1=12
      - AMF_INTERFACE_NAME_FOR_NGAP=eth0
      - AMF_INTERFACE_NAME_FOR_N11=eth0
      - SMF_INSTANCE_ID_0=1
```

```
    - SMF_FQDN_0=oai-smf
    - SMF_IPV4_ADDR_0=192.168.70.133
    - SMF_HTTP_VERSION_0=v1
    - SELECTED_0=true
    - SMF_INSTANCE_ID_1=2
    - SMF_FQDN_1=oai-smf
    - SMF_IPV4_ADDR_1=0.0.0.0
    - SMF_HTTP_VERSION_1=v1
    - SELECTED_1=false
    - MYSQL_SERVER=192.168.70.131
    - MYSQL_USER=root
    - MYSQL_PASS=linux
    - MYSQL_DB=oai_db
    - OPERATOR_KEY=63bfa50ee6523365ff14c1f45f88737d
    - NRF_IPV4_ADDRESS=192.168.70.130
    - NRF_PORT=80
    - NF_REGISTRATION=yes
    - SMF_SELECTION=yes
    - USE_FQDN_DNS=yes
    - EXTERNAL_AUSF=yes
    - NRF_API_VERSION=v1
    - NRF_FQDN=oai-nrf
    - AUSF_IPV4_ADDRESS=192.168.70.138
    - AUSF_PORT=80
    - AUSF_API_VERSION=v1
    - AUSF_FQDN=oai-ausf
    - INT_ALGO_LIST=["NIA1" , "NIA2"]
    - CIPH_ALGO_LIST=["NEA1" , "NEA2"]
  depends_on:
    - mysql
    - vpp-upf
```

```
      - oai-ext-dn
      - oai-ausf
    volumes:
      - ./amf-healthcheck.sh:/openair-amf/bin/amf-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-amf/bin/amf-healthcheck.sh"
      interval: 10s
      timeout: 15s
      retries: 5
    networks:
      public_net:
        ipv4_address: 192.168.70.132
  oai-smf:
    container_name: "oai-smf"
    image: oai-smf:latest
    environment:
      - TZ=Europe/Paris
      - INSTANCE=0
      - PID_DIRECTORY=/var/run
      - SMF_INTERFACE_NAME_FOR_N4=eth0
      - SMF_INTERFACE_NAME_FOR_SBI=eth0
      - SMF_INTERFACE_PORT_FOR_SBI=80
      - SMF_INTERFACE_HTTP2_PORT_FOR_SBI=9090
      - SMF_API_VERSION=v1
      - DEFAULT_DNS_IPV4_ADDRESS=192.168.18.129
      - DEFAULT_DNS_SEC_IPV4_ADDRESS=192.168.18.129
      - AMF_IPV4_ADDRESS=192.168.70.132
      - AMF_PORT=80
      - AMF_API_VERSION=v1
      - AMF_FQDN=oai-amf
      - UDM_IPV4_ADDRESS=192.168.70.137
```

```
      - UDM_PORT=80
      - UDM_API_VERSION=v1
      - UDM_FQDN=oai-udm
      - UPF_IPV4_ADDRESS=192.168.70.202
      - UPF_FQDN_0=gw1.vppupf.node.5gcn.mnc95.mcc208.3gppnetwork.org
      - NRF_IPV4_ADDRESS=192.168.70.130
      - NRF_PORT=80
      - NRF_API_VERSION=v1
      - NRF_FQDN=oai-nrf
      - REGISTER_NRF=yes
      - DISCOVER_UPF=yes
      - USE_FQDN_DNS=yes
      - USE_NETWORK_INSTANCE=yes
      - DNN_RANGE1=12.1.1.2 - 12.1.1.128
      - DNN_RANGE0=12.2.1.2 - 12.2.1.128
      - DNN_NI1=default
    extra_hosts:
      - "gw1.vppupf.node.5gcn.mnc95.mcc208.3gppnetwork.org:192.168.70.202"
    depends_on:
      - oai-amf
    volumes:
      - ./smf-healthcheck.sh:/openair-smf/bin/smf-healthcheck.sh
    healthcheck:
      test: /bin/bash -c "/openair-smf/bin/smf-healthcheck.sh"
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      public_net:
        ipv4_address: 192.168.70.133
  vpp-upf:
```

```
    image: oai-upf-vpp:latest
    privileged: true
    container_name: "vpp-upf"
    environment:
      - NWI_N3=access.oai.org
      - NWI_N6=core.oai.org
      - GW_ID=1
      - MNC03=95
      - MCC=208
      - REALM=3gppnetwork.org
      - NETWORK_UE_IP=12.2.1.0/24
      - N3_IPV4_ADDRESS_REMOTE=192.168.72.141 #   GNB  IP Address
      - N4_IPV4_ADDRESS_REMOTE=192.168.70.133 #   SMF  IP Address
      - N6_IPV4_ADDRESS_REMOTE=192.168.73.135 # EXT-DN IP Address
      - VPP_MAIN_CORE=0
      - VPP_CORE_WORKER=1
#       - VPP_PLUGIN_PATH=/usr/lib64/vpp_plugins/            # RHEL7
      - VPP_PLUGIN_PATH=/usr/lib/x86_64-linux-gnu/vpp_plugins/ # Ubntu18.04
      - INTERFACE_ACCESS=eth1
      - INTERFACE_CORE=eth2
      - INTERFACE_CP=eth0
      - NSSAI_SD_0=123
      - SST=222
      - DNN=default
      - REGISTER_NRF=yes
      - NRF_IP_ADDR=192.168.70.130
      - NRF_PORT=9090
      - HTTP_VERSION=2
    depends_on:
      - oai-nrf
    healthcheck:
```

```
      test: /bin/bash -c "pgrep vpp"
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      public_net:
        ipv4_address: 192.168.70.134
      public_net_access:
        ipv4_address: 192.168.72.134
      public_net_core:
        ipv4_address: 192.168.73.134
  oai-ext-dn:
    image: ubuntu:bionic
    privileged: true
    container_name: "oai-ext-dn"
    entrypoint: /bin/bash -c \
      "apt update; apt install -y iptables iproute2 iperf3 iputils-ping;"\
      "iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE;"\
      "ip route add 12.1.1.0/24 via 192.168.73.202 dev eth0; sleep infinity"
    command: ["/bin/bash", "-c", "trap : TERM INT; sleep infinity & wait"]
    depends_on:
      - vpp-upf
    networks:
      public_net_core:
        ipv4_address: 192.168.73.135
networks:
  public_net:
    driver: bridge
    name: demo-oai-public-net
    ipam:
      config:
```

```
        - subnet: 192.168.70.0/24
    driver_opts:
      com.docker.network.bridge.name: "demo-oai"
public_net_access:
    name: oai-public-access
    ipam:
      config:
        - subnet: 192.168.72.0/24
    driver_opts:
      com.docker.network.bridge.name: "cn5g-access"
public_net_core:
    name: oai-public-core
    ipam:
      config:
        - subnet: 192.168.73.0/24
    driver_opts:
      com.docker.network.bridge.name: "cn5g-core"
```

Tabla D.1: Fichero de configuración $docker - compose - basic - vpp - nrf.yaml$

D.2. Fichero *docker-compose-gnbsim-vpp.yaml*

```
version: '3.8'
services:
  gnbsim-vpp:
    container_name: gnbsim-vpp
    image: gnbsim:latest
    privileged: true
    environment:
      - MCC=208
      - MNC=95
      - GNBID=5
      - TAC=0x00a000
      - SST=222
      - SD=00007b
      - PagingDRX=v32
      - RANUENGAPID=0
      - IMEISV=35609204079514
      - MSIN=0000000031
      - RoutingIndicator=1234
      - ProtectionScheme=null
      - KEY=0C0A34601D4F07677303652C0462535B
      - OPc=63bfa50ee6523365ff14c1f45f88737d
      - DNN=default
      - URL=http://www.asnt.org:8080/
      - NRCellID=1
      - USE_FQDN=no
#     - USE_FQDN=yes
#     - AMF_FQDN=amf.oai-5gc.eur
      - NGAPPeerAddr=192.168.70.132
      - GTPuLocalAddr=192.168.72.141
      - GTPuIfname=eth1
    networks:
```

```
      public_net:
        ipv4_address: 192.168.70.141
      public_net_access:
        ipv4_address: 192.168.72.141
    healthcheck:
      test: /bin/bash -c "ifconfig gtp-gnb"
      interval: 10s
      timeout: 5s
      retries: 5
networks:
  public_net:
    external:
      name: demo-oai-public-net
  public_net_access:
    external:
      name: oai-public-access
```

Tabla D.2: Fichero de configuración $docker - compose - gnbsim - vpp.yaml$

D.3. Fichero *docker-compose-ueransim-vpp.yaml*

```
version: '3.8'
services:
  ueransim:
    container_name: ueransim
    image: ueransim:latest
    privileged: true
    environment:
      # GNB Congig Parameters
      - MCC=208
      - MNC=95
      - NCI=0x000000010
      - TAC=0xa000
      - LINK_IP=192.168.70.141
      - NGAP_IP=192.168.70.141
      - GTP_IP=192.168.72.141
      - NGAP_PEER_IP=192.168.70.132
      - SST=222
      - SD=123
      - IGNORE_STREAM_IDS=true
      # UE Config Parameters
      - NUMBER_OF_UE=5
      - IMSI=208950000000031
      - KEY=0C0A34601D4F07677303652C0462535B
      - OP=63bfa50ee6523365ff14c1f45f88737d
      - OP_TYPE=OPC
      - AMF_VALUE=8000
      - IMEI=356938035643803
      - IMEI_SV=0035609204079514
      - GNB_IP_ADDRESS=192.168.70.141
      - PDU_TYPE=IPv4
      - APN=default
```

```
      - SST_C=1
      - SD_C=1
      - SST_D=1
      - SD_D=1
    networks:
      public_net:
        ipv4_address: 192.168.70.141
      public_net_access:
        ipv4_address: 192.168.72.141
    healthcheck:
      test: /bin/bash -c "ifconfig uesimtun0"
      interval: 10s
      timeout: 5s
      retries: 5
networks:
  public_net:
    external:
      name: demo-oai-public-net
  public_net_access:
    external:
      name: oai-public-access
```

Tabla D.3: Fichero de configuración *docker − compose − ueransim − vpp.yaml*