UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

O-RAN y SOFTWARE CON APIs ABIERTAS EN LAS REDES DE TELEFONÍA MÓVIL DE NUEVA GENERACIÓN

Pilar Iniesta Núñez

Tutor: José Francisco Monserrat del Río

Cotutor: Sergio Pastor Tur

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2020-21

Valencia, 2 de diciembre de 2020

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

El desarrollo de las redes 5G ha hecho que los sistemas de comunicaciones que se conocen hasta hoy en día estén cambiando de manera significativa. Las altas prestaciones necesarias para implementar la nueva generación de redes móviles han motivado el avance de nuevas técnicas para su desarrollo. Uno de los mayores avances en los últimos años en la implementación de la red se encuentra en la apertura de las interfaces y de la arquitectura de red, las cuales tradicionalmente han sido privadas. Este cambio de paradigma es la respuesta más eficiente a las exigencias del diseño de la red 5G.

Este cambio en el modelo de red se implementa en la interfaz radio conocida como RAN (Radio Access Network). Concretamente, el RAN es el encargado de proveer el acceso radio y coordinar la gestión de recursos para que los usuarios puedan conectarse. Su apertura proporciona una nueva arquitectura de red, el llamado Open RAN. Se trata de una arquitectura de red basada en interfaces abiertas, con modelos totalmente virtualizados y que garantizan la interoperabilidad entre los proveedores para la entrada de otros.

Mediante la plataforma O-RAN, una organización que impulsa este tipo de arquitectura, este proyecto pretende investigar el estado actual de cambio en el paradigma en la implementación de redes, estudiar sus prestaciones y analizar los futuros cambios que va a suponer el desarrollo de este nuevo modelo de red.

Palabras Clave: redes móviles, RAN, 5G, virtualización, Open RAN, O-RAN.

Resum

El desenvolupament de les xarxes 5G ha fet que els sistemes de comunicacions que es coneixen fins avui dia estiguen canviant de manera significativa. Les altes prestacions necessàries per implementar la nova generació de xarxes mòbils han motivat l'avanç de noves tècniques per al seu desenvolupament. Un dels majors avenços en els darrers anys en la implementació de la xarxa es troba en l'obertura de les interfícies i de l'arquitectura de xarxa, les quals tradicionalment han estat privades. Aquest canvi de paradigma és la resposta més eficient a les exigències de el disseny de la xarxa 5G.

Aquest canvi en el model de xarxa s'implementa en la interfície ràdio coneguda com RAN (Ràdio Access Network). Concretament, el RAN és l'encarregat de proveir l'accés ràdio i coordinar la gestió de recursos perquè els usuaris puguin connectar-se. La seva obertura proporciona una nova arquitectura de xarxa, l'anomenat Open RAN. Es tracta d'una arquitectura de xarxa basada en interfícies obertes, amb models totalment virtualitzats i que garanteixen la interoperabilitat entre els proveïdors per l'entrada d'altres.

Mitjançant la plataforma O-RAN, una organització que impulsa aquest tipus d'arquitectura, aquest projecte pretén investigar l'estat actual de canvi en el paradigma en la implementació de xarxes, estudiar-ne les prestacions i analitzar els futurs canvis que suposarà el desenvolupament d'aquest nou model de xarxa.

Paraules clau: xarxes móbils, RAN, 5G, virtualització, Open RAN, O-RAN.

Abstract

The development of 5G networks has meant that the communication systems that are known to this day are changing significantly. The high performance required to implement the new generation of mobile networks has motivated the advancement of new techniques for their development. One of the greatest advances in the last years in the implementation of the network is in the opening of the interfaces and the network architecture, which have traditionally been private. This paradigm shift is the most efficient response to the demands of 5G network design.

This change in the network model is implemented in the radio interface known as RAN (Radio Access Network). Specifically, the RAN is in charge of providing radio access and coordinating resource management so that users can connect. Its opening provides a new network architecture, the so-called Open RAN. It is a network architecture based on open interfaces, with fully virtualized models that guarantee interoperability between providers for the entry of others.

Through the O-RAN platform, an organization that promotes this type of architecture, this project aims to investigate the current state of change in the paradigm in the implementation of networks, study its benefits and analyze the future changes that the development of this will entail. new network model.

Key words: mobile network, RAN, 5G, virtualization, Open RAN, O-RAN.

Índice

Tabla de figuras

Lista de acrónimos

3GPP: *3rd Generation Partnership*

BBU: *Band Base Unit*

CU: *Central Unit*

DU: *Distributed Unit*

eCPRI: *Common Public Radio Interface*

***eNB**: Evolved Node B*

EPC: *Evolved Packet Core*

EPS: *Evolved Packet System*

E-UTRAN: *Evolved Universal Terrestrial Access Network*

***gNB**: GNode B*

GSM: *Global System for Mobile*

HSS: *Home Subscriber Service*

ICI: *Inter-Cell Interference*

IoT: Internet of things

LTE: *Long Term Evolution*

MAC: *Medium-Access Control*

MCG: *Grupo de Comunicaciones Móviles*

MIMO: *Multiple-Input Multiple-Output*

MME: *Mobility Management Entity*

NAS: *Non-Access Stratum*

NF: *Network Functions*

NR: *New Radio*

NSA: *Non Stand Alone*

PCRF: *Policy and Charging Rules Function*

PDCP: *Packet Data Convergence Protocol*

PHY: *Physical Layer*

QoS: Quality of Service

RAN: *Radio Access Network*

RIC: RAN Intelligent Controller

RLC: *Radio-Link Control*

ROHC: *Robust Header Compression*

RRC: *Radio Resource Control*

RRH: *Remote Radio Head*

RRM: *Radio Resource Management*

SA: *Stand-Alone*

SDAP: *Service Data Adaption Protocol*

S-GW: *Serving Gateway*

SMO: *Service Management Orquestration*

TIC: *Tecnologías de la Información y Comunicaciones*

TIP: *Telecom Infra Group*

V2X: *vehicle to everything*

Capítulo 1. Introducción

1.1 Contexto

A través de las distintas generaciones, la tecnología ha favorecido los avances necesarios para que las sociedades se desarrollen y puedan hacer uso de ellas. Desde todos los ámbitos de la vida, la evolución de la tecnología ha supuesto un cambio importante. Por lo que, las comunicaciones, algo inherente en el ser humano, no iba a ser menos.

A partir de la presentación del teléfono fijo, las comunicaciones hasta entonces conocidas dieron un importante giro para convertirse en uno de los ejes principales de las Tecnologías de la Información y Comunicaciones (TIC). Con el paso a la telefonía móvil el desarrollo es todavía mayor ya que la aparición de las redes celulares marca los primeros pasos para la conexión global [1].

En menos de 40 años, estar conectado en cualquier parte del mundo y a cualquier hora ha supuesto un aumento de la demanda sin límites que ha hecho que las operadoras requieran nuevos servicios y funcionalidades en sus redes. Las distintas generaciones de redes marcaban el paso a seguir para actualizarse y hacer frente a los cambios en la sociedad.

La primera generación 1G solo permitía realizar llamadas de voz basadas en transmisiones de radio analógicas con baja velocidad y sin precisión impedían comunicarse con otros países. En la década de los 90, el aumento exponencial de los usuarios de voz provocó la aparición de una nueva generación el 2G. Su característica fundamental es la presencia de tecnología digital estandarizada en Europa a través de la tecnología GSM *(Global System for Mobile)* que permitió también enviar mensajes de texto entre varios dispositivos móviles [2].

El salto de una tecnología a la siguiente fue importante ya que aparecía en el juego un nuevo participante: internet. Además, la necesidad continua de converger los servicios de voz y datos que tuviera acceso inalámbrico a internet, junto con las mayores tasas de transmisión de datos supuso el origen de 3G y con ello la aparición de una organización conocida como 3GPP, responsable de la estandarización las diferentes tecnologías venideras.

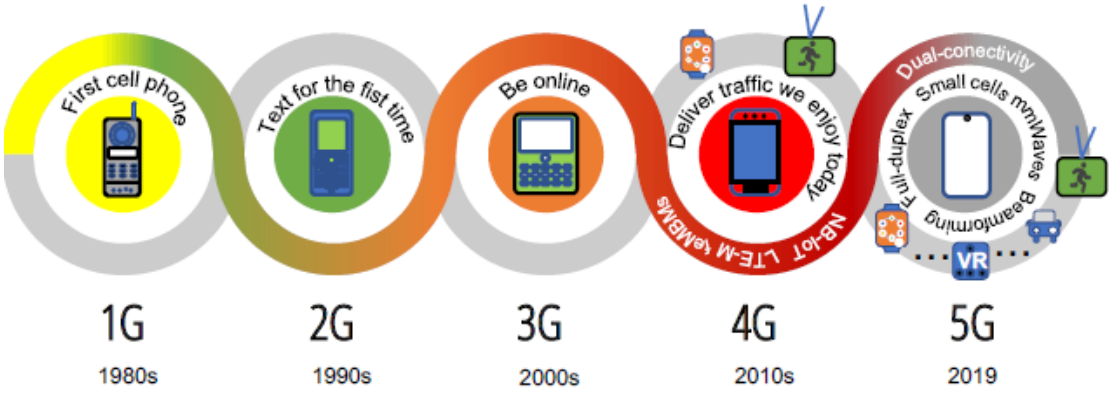


Figura 1: Evolución de las redes celulares [3].

Sin embargo, el salto más importante en las redes celulares fue, sin duda, la introducción de la tecnología 4G. Con ella, se integra por primera vez las redes banda ancha fija y móvil y tecnologías como LTE (*Long Term Evolution)* simplifican el sistema 3G, constituyendo un sistema de red basado en Internet Protocol (IP) de extremo a extremo, que provoca la reducción de costes en el despliegue, el incremento de la calidad de servicio (QoS) y mejora las tasas de transmisión de datos.

Asimismo, el crecimiento constante de la demanda de nuevos servicios para una sociedad globalizada y conectada hace que se generen nuevos retos para las comunicaciones. Por su parte, las redes móviles necesitaban dar respuesta a estas nuevas necesidades que se plantean en la

actualidad. La aparición del 5G pretende romper con todos los esquemas sobre las redes celulares existentes.

Gracias a características como la interoperabilidad con otras redes de generaciones anteriores, la virtualización y la accesibilidad sin cables, el 5G se convertirá en el protagonista del futuro. Es tal su impacto, que según Ericcson, para 2025 se esperan 2.800 millones de suscriptores en esta tecnología, un 30% de las suscripciones móviles mundiales [4].

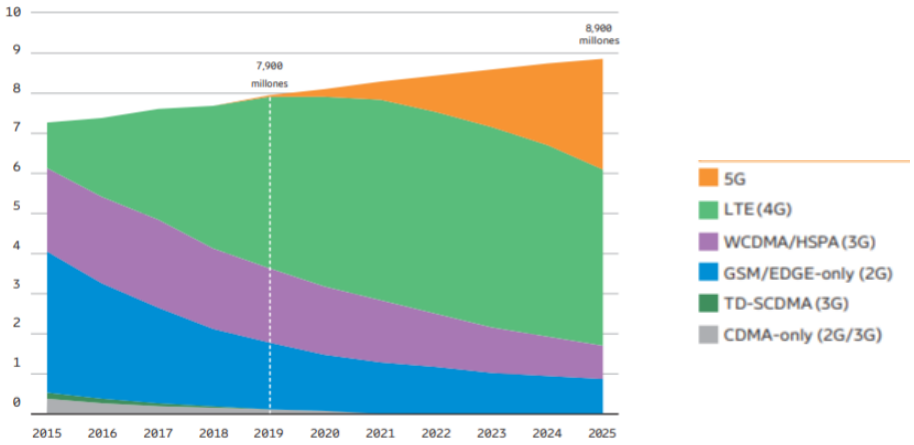


Figura 2: Suscriptores móviles por tecnología (miles de millones) [4].

Las nuevas necesidades que tiene el mercado hacen que los usuarios busquen mejores características en el manejo de las redes móviles para en definitiva tener una experiencia de uso mejorada. De la misma forma, se esperan nuevas aplicaciones relacionadas con la industria mediante el desarrollo del Internet de las cosas (IoT), la sanidad, los videojuegos con la realidad aumentada o virtual, los automóviles o con el surgimiento de ciudades u hogares inteligentes, todo un abanico de posibilidades que se harán realidad gracias al 5G.

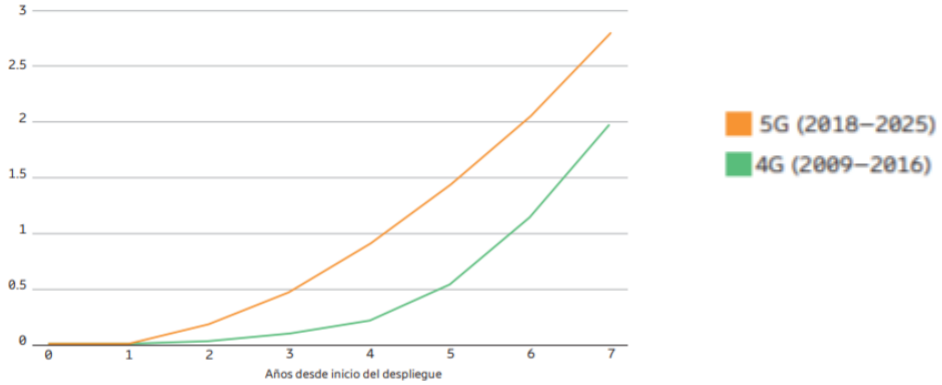


Figura 3: Comparativa entre las suscripciones de 5G Y 4G en sus primeros años de despliegue (miles de millones) [4].

Por ello, las expectativas de penetración de esta nueva tecnología son todavía mejores que las del 4G en su introducción. La sociedad quiere un cambio, está preparada para ello y así lo demuestran con datos como los de la figura anterior en los que la adopción de suscriptores al 5G se produce de forma más rápida que para 4G.

Para que todo esto sea posible, la industria de las comunicaciones se enfrenta a varios retos que debe tener cuenta. La conexión de personas, industria y transporte, entre otras, va a hacer que el tráfico de datos aumente sin precedentes, al igual que sucederá con la velocidad y la latencia que tendrán que dar un servicio óptimo para que toda esta compleja red funcione.

Por lo que, para las redes que hoy en día proporcionan un buen servicio supone un gran problema, ya que LTE no es capaz de aguantar tales características de tráfico de datos y velocidad que se requieren para estas aplicaciones. Por este motivo, el 5G va a tener un papel imprescindible en estos nuevos escenarios.

1.2 Motivación

Los avances en la tecnología están experimentando un crecimiento exponencial, haciendo lo que hace un tiempo parecía una utopía, ahora se está convirtiendo en una realidad. Un ejemplo de ello son los conceptos de ultra baja latencia, industria 4.0 o V2X (*vehicle to everything*), los cuales hace unos años eran objetivos muy lejanos de implementar y actualmente ya están apareciendo los primeros prototipados.

La tecnología en estos momentos se encuentra en un punto de ruptura con la generación anterior debido a grandes cambios sustanciales que están comenzando ahora y que se esperan tengan su gran éxito dentro de unos años. O, dicho de otra manera, a corto plazo se espera que se asienten las bases para que se una nueva revolución tecnológica.

La hiperconectividad, la automatización de procesos y los datos serán los protagonistas en el nuevo paradigma del desarrollo tecnológico. Por este motivo, la gran implicación que tiene las comunicaciones móviles es notable, no solo por su implicación en nuevas generaciones de redes sino también en el compromiso de mejorar sus propios recursos para poder cumplir con todas las expectativas.

Muchas de estas nuevas formas de entender e implementar la tecnología se encuentran todavía muy poco maduras y en fases muy tempranas para su completo despliegue de forma generalizada. Por ello, disponer de centros de investigación en entornos donde se dispongan de los recursos necesarios, suponen un importante empuje para el desarrollo tecnológico.

Este proyecto se centra en la parte radio de un sistema de comunicaciones, uno de los elementos que tradicionalmente no han desarrollado grandes cambios hasta la llegada del 5G que plantea un cambio radical en la arquitectura original. Si bien, hasta ahora, las redes dependían de un hardware específico que era propiedad de un grupo de proveedores, con la apertura de las redes se abre un nuevo camino en el despliegue.

La introducción de un nuevo paradigma en las telecomunicaciones que parte con la premisa de que las redes sean abiertas, flexibles y virtualizadas, hace que el RAN esté evolucionado hacia el conocido como Open RAN, cuyo objetivo principal es constituir un entorno donde diferentes proveedores pueden utilizar un hardware y software estándar que esté basado en la nube con interfaces abiertas. Este nuevo enfoque consiste en desagregar el software del hardware lo que permite que el software RAN se implemente en plataformas hardware comunes, ya que las interfaces abiertas soportarán la incorporación de diferentes componentes que favorecerán la interoperabilidad entre múltiples proveedores.

Esta nueva arquitectura de acceso radio virtualizada motivará el cumplimiento de las expectativas que se esperan del 5G de forma más rápida debido a permitirá funciones avanzadas entre las que destacan la integración de la parte del núcleo a las redes virtualizadas y de transporte, la automatización de las redes o la cooperación entre todos los nodos de la red.

La implementación de Open RAN permitirá entrar en escena nuevos proveedores que beneficiarán a que las redes sean más seguras y flexibles además de que fomentarán que exista un ecosistema más competitivo en todos los mercados. Esto provocará la constante actualización de estándares y soluciones para mantenerse al día en la innovación.

La colaboración entre los distintos organismos público-privados es esencial para que Open RAN se haga realidad. Organizaciones como Alianza O-RAN están haciendo que esta nueva forma de entender la RAN pueda llevarse a cabo gracias al esfuerzo conjunto para superar los desafíos.

El ecosistema Open RAN se encuentra en una fase de crecimiento, gracias a la implicación de numerosas organizaciones entre las que destaca 3GPP, Telecom Infra Project (TIP) o los proveedores más importantes del sector como Cisco, Ericsson entre otros. Cada una de estas empresas u organizaciones con sus propias perspectivas está enriqueciendo cada día a Open RAN generando gran interés en toda la industria y la investigación.

Aprovechar el potencial que la comunidad de O-RAN está aportando provoca que se trabaje de forma más rápida y se agilicen procesos para conseguir el establecimiento de arquitecturas Open RAN, lo que supondrá un éxito para las redes inalámbricas.

1.3 Objetivos

En el presente Trabajo Fin de Grado, el objetivo principal es desarrollar la arquitectura de red de acceso RAN de código abierto que está siendo desarrollada desde el 2019 por la comunidad de O-RAN. Bajo las especificaciones de 3GPP y las de O-RAN se podrá establecer el despliegue de la arquitectura Open RAN dentro del proyecto Valencia Campus 5G, en el Instituto de Telecomunicaciones y Aplicaciones Multimedia (iTEAM) de la Universidad Politécnica de Valencia.

Para su desarrollo, se determinan una serie de objetivos específicos que deben considerarse:

- Estudio completo de la arquitectura de red de acceso RAN de 4G y su evolución hasta llegar a las nuevas arquitecturas presentes en 5G.
- Implementación y configuración del software requerido por los equipos para el despliegue de la RAN de O-RAN.
- Despliegue de los componentes de gestión de RAN definidos por la última versión de O-RAN desde su plataforma de código abierto, así como sus interfaces y casos de uso desarrollados por la misma.
- Validación de los casos de uso e interfaces propuestas a través de distintas herramientas software que permiten ver el envío de paquetes y su conexión.

Capítulo 2. Estado del arte

2.1 Arquitectura de red LTE

Con la llegada de LTE (*Long Term Evolution*) se empezaron a comercializar las redes de comunicaciones de cuarta generación (4G). A partir de la *Release 8* publicada por 3GPP (*3rd Generation Partnership*), se estableció el estándar de LTE movido por una serie de necesidades como la alta demanda de datos, la complejidad de los sistemas de 3G o elevados costes que motivaron la aparición del 4G, convirtiéndose en el estándar de comunicaciones que ha marcado los pasos a seguir en el futuro [5].

Su principal objetivo era conseguir simplificar la red móvil heredada de 3G y diferenciar de manera clara la parte RAN *(Radio Access Network*) de la parte del núcleo de la red CN (Core Network). Por lo que la RAN evolucionó hacia la conocida como E-UTRAN *(Evolved Universal Terrestrial Access Network)* que implementaba nuevas tecnologías de acceso radio y el núcleo de la red pasó de una arquitectura basada en conmutación de circuitos a una arquitectura basada en TCP/IP que se llamó EPC (*Evolved Packet Core*) [6].

Pero además este nuevo diseño de la arquitectura de 4G, conocido como EPS (*Evolved Packet System*) quería optimizar el tráfico de datos desde y hacia los usuarios. Bajo esta premisa, la reducción del número de nodos que están implicados en la conmutación de paquetes fue la solución y la arquitectura de 4G pasó a ser conocida como plana [7].

De esta forma, E-UTRAN es el responsable de toda la funcionalidad relacionada con la parte radio de la red general, que incluye los protocolos de retransmisión, codificación y gestión de recursos radio. Y el EPC que, aunque no se encarga de las funciones de la interfaz radio, proporciona una red de banda ancha móvil completa, que facilita la autenticación y la configuración de las conexiones extremo a extremo.

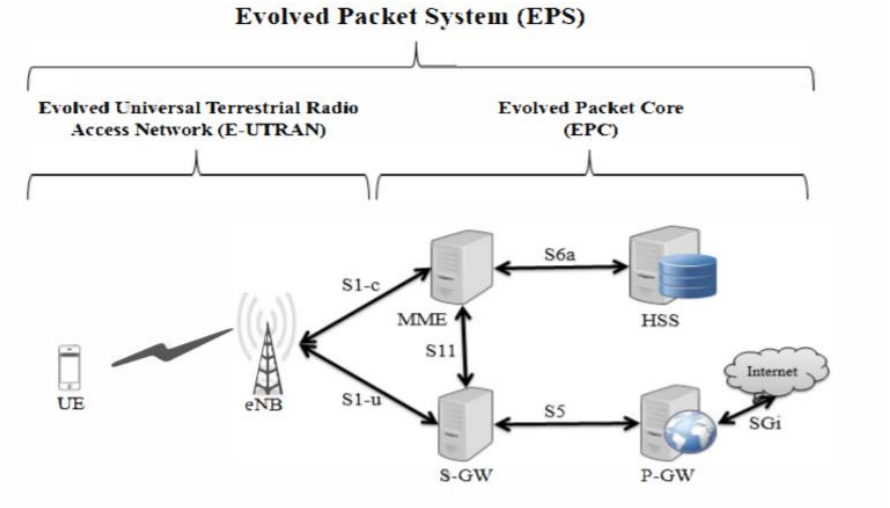


Figura 4: Arquitectura de red de los nodos de LTE [7].

2.1.1 Componentes de EPC

Como se puede ver en la Figura 4, el EPC está constituido por diferentes nodos, y aunque el estudio no se centre en la parte del núcleo sí que es importante conocer las funciones que los dos nodos principales tienen, ya que esos sí que tendrán interacción con la parte radio del sistema de LTE a través de la interfaz S1 [6]. Esta interfaz S1 separa el plano de control de señalización (S1-c) del tráfico de datos que generan los usuarios (S1-u) ya que de esta forma se mejora la escalabilidad de la red.

MME (*Mobility Management Entity*): Situado en el plano de control, es el nodo que se encarga de la señalización relativa a la movilidad y a la seguridad de los paquetes que recibe de E-UTRAN. Se encarga, por tanto, de la conexión/ liberación de las portadoras a una terminal, la gestión de los estados IDLE a ACTIVO y el establecimiento de claves de seguridad. Cabe destacar que su función operativa entre el EPC y el terminal móvil se denomina NAS (*Non-Access Stratum*).

S-GW (*Serving Gateway*): es el nodo que establece la comunicación entre la parte radio y el Core, transportando el tráfico de datos IP del usuario. Actúa como un ancla de movilidad cuando los terminales se mueven entre los diferentes nodos, así como para otras tecnologías 3GPP (GSM o 3G). El S-GW también gestiona la recopilación de información y estadísticas para la facturación.

Además, el EPC también contiene otros tipos de nodos como PCRF (*Policy and Charging Rules Function*) responsable de la gestión de la calidad del servicio QoS (*quality-of-service*) y el HSS (*Home Subscriber Service*) que contiene una base de datos sobre la información del abonado.

2.1.2 Componentes de la parte RAN E-UTRAN

Teniendo una idea clara de las funciones que tiene el EPC es necesario analizar la parte radio, que es la protagonista de esta investigación y así poder tener un estudio completo de las redes de comunicaciones de LTE.

La red de acceso por radio LTE utiliza una arquitectura plana basándose en un tipo de nodo: eNB (*Evolved Node B*). El eNB es el encargado de todas las funciones relacionadas con la parte radio en una o varias celdas. Es importante tener en cuenta que un eNB es un nodo lógico y no una implementación física, por lo que las posibilidades para su puesta en marcha son variadas.

La implementación más común de un eNB, conocida como RAN distribuido, consta de una unidad de procesamiento de banda base BBU (*Band Base Unit*) a la que se encuentran conectados varios cabezales remotos RRH (*Remote Radio Head*). Un ejemplo de esto último es una gran cantidad de celdas *indoor* (interiores) o varias celdas a lo largo de una carretera que pertenecen al mismo eNB. Por ello, es importante destacar, que una estación base no es lo mismo que un eNB, sino que es una posible implementación de este.

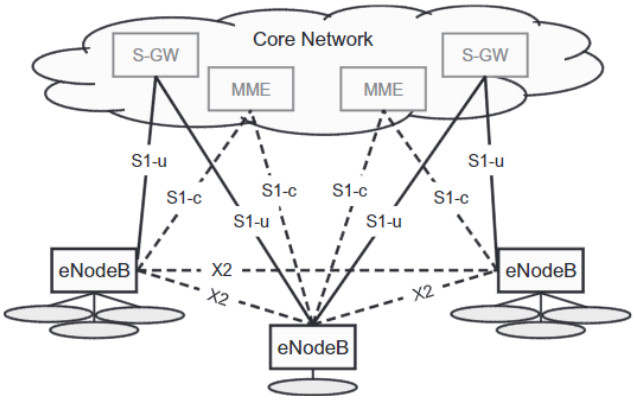


Figura 5:Interfaces de RAN [6].

Como se puede ver en la figura anterior, el eNB, se conecta al EPC por medio de la interfaz S1, más específicamente al S-GW por medio de la interfaz S1-u y al MME mediante el plano de control, S1-c. Un eNB se puede conectar a varios MME/S-GW con el fin de compartir la carga y la redundancia. Con respecto a la interfaz X2, es la que conecta los eNB entre sí, y se utiliza principalmente para permitir la movilidad en modo activo de la red. También se puede utilizar para gestión de recursos de radio RRM (*Radio Resource Management*) entre varias celdas para evitar interferencias entre las celdas ICI (*Inter-Cell Interference*) y para soportar la movilidad sin perdidas entre celdas vecinas mediante el reenvío de paquetes [8].

La Figura 6 ilustra la arquitectura del protocolo RAN en la que aparecen todas las entidades que han sido definidas por 3GPP, incluyendo las capas de los dos planos en los que se descompone un eNB.

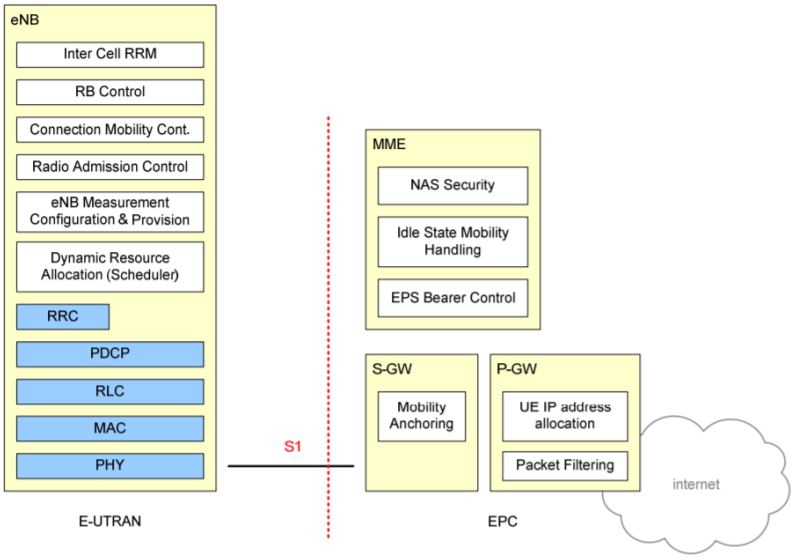


Figura 6:División funcional entre E-UTRAN y EPC [8].

Ambos planos presentan las distintas entidades que forman parte del eNB, y que comparten muchas de las funcionalidades que presenta la arquitectura. Sin embargo, existen diferencias sustanciales que hacen que cada plano tenga un papel fundamental en la comunicación de LTE. Por esta razón, se realizará un estudio completo de las partes que comparten ambos y después sus diferencias, que ya se pueden intuir según esta figura [6].

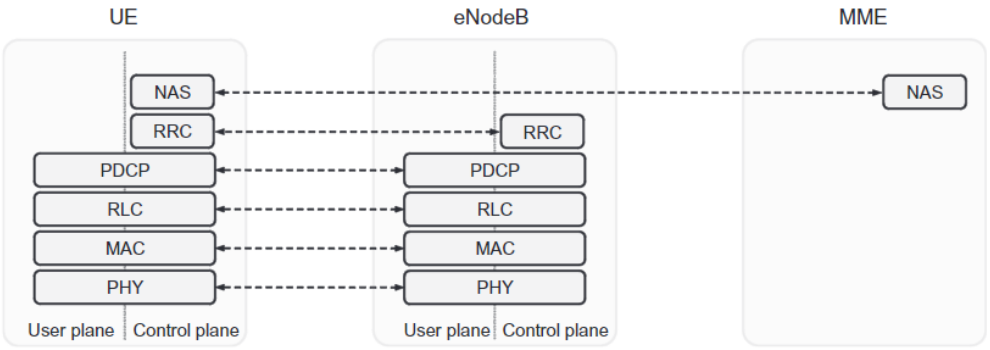


Figura 7:Protocolo de arquitectura RAN genérico [6].

Las diferentes entidades de protocolo de red de acceso a radio se describen a continuación y están representadas en la siguiente figura, las cuales forman parte de la arquitectura de red en el enlace descendente, y cuyo plano es el plano de usuario:

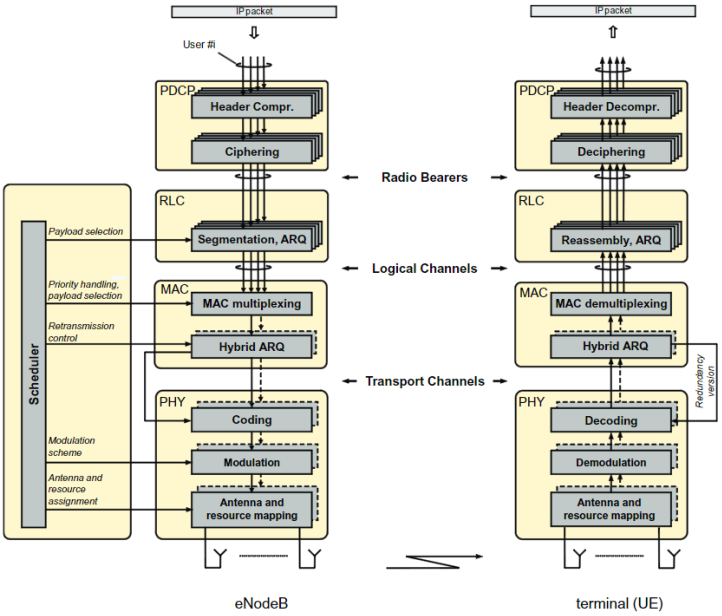


Figura 8:Protocolo de arquitectura LTE en el enlace descendente [6].

PDCP (*Packet Data Convergence Protocol*) realiza la compresión del encabezado IP para reducir el número de bits a transmitir a través de la interfaz radio. El mecanismo de compresión de encabezados se basa en *Robust Header Compression* (ROHC), un algoritmo de compresión estandarizado. También es responsable del cifrado y para el plano de control, de la protección de la integridad de los datos transmitidos y la eliminación de duplicados para la transferencia de datos. Existe una entidad PDCP por cada portadora de radio que se configura para un terminal.

RLC (*Radio-Link Control*) es responsable de la segmentación y concatenación, de la gestión de la retransmisión, de la detección de duplicados y la entrega de paquetes consecutivos a las capas superiores ya que proporciona servicios al PDCP en forma de portadoras de radio y existe una entidad RLC para cada portadora de radio en un terminal.

MAC (*Medium-Access Control*) gestiona la multiplexación de canales lógicos, retransmisiones-ARQ híbridas y los procesos de *scheduling* tanto en el enlace descendente como en el enlace ascendente del eNB. Para proporcionar los servicios necesarios a RLC, la capa MAC utiliza canales lógicos en vez de portadoras.

PHY (*Physical Layer*) es la capa que asume las tareas de codificación y decodificación, modulación y demodulación, MIMO (*Multiple-Input Multiple-Output*) y el mapeo de recursos, es decir, todas aquellas funciones que tienen que ver con la capa física. La capa PHY ofrece sus servicios a la capa MAC en forma de canales de transporte.

2.2 Evolución de RAN

La arquitectura RAN de la *Release 9*, estandarizada por el 3GPP quedó obsoleta debido a la rápida evolución de las comunicaciones. Uno de estos motivos fue el aumento de tráfico de datos con la aparición de móviles inteligentes y tabletas por lo que era necesario reconsiderar las nuevas necesidades de la red. Algunas de ellas, como la eficiencia, la flexibilidad o la escalabilidad en la red de acceso radio (RAN) forzaron a un replanteamiento radical del tradicional RAN, dando

lugar a la unidad de radio remota RRH (*Radio Remote Head*) y la unidad de banda base BBU (*Base Band Unit*).

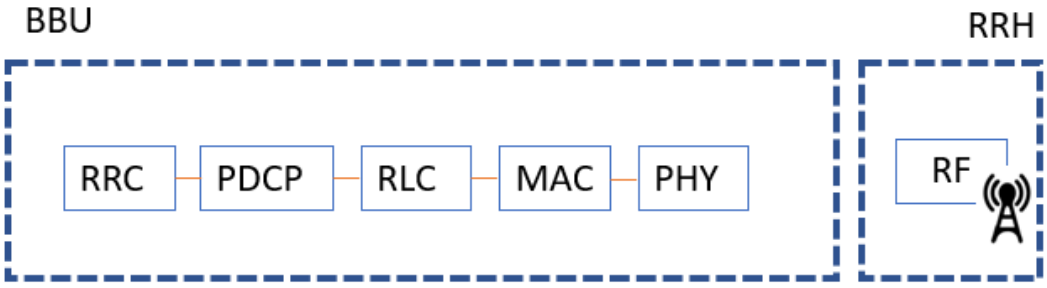


Figura 9: División de capas de LTE.

Estas dos nuevas unidades, BBU y RRH recogen las funcionalidades del eNB incorporando en ellas las distintas entidades por las que estaba formado el RAN de 4G tal y como se muestra en la Figura 9.

Para los operadores era de vital importancia aumentar la capacidad de los sistemas para soportar el creciente número de datos. En este contexto, las implementaciones de células pequeñas son una de las formas más efectivas de conseguir incrementar la capacidad del sistema ya que mejora la reutilización de recursos radio, pero sin embargo implica gran coste para los operadores. De esta forma, el modo de operación que estaba establecido era la arquitectura RAN distribuida, en el que las funciones de RRH y del BBU estaban ubicadas en sitios celulares y conectadas a una central de conmutación [9].

Para solucionar estos problemas que mermaban las cualidades requeridas por las operadoras, nació la arquitectura C-RAN una mejora a la arquitectura convencional de LTE puesto que era capaz de gestionar la gran demanda de datos en la red con un coste más bajo. El propio nombre indica la evolución ya que la "C" significa tanto centralizado como virtualizado, dos aspectos separados pero relacionados en esta nueva arquitectura [10].

La centralización de las BBU ofrece a los operadores de las redes móviles una serie de beneficios como son la reducción de costes OPEX, CAPEX o mayor desempeño gracias a una mejor coordinación entre células [11].

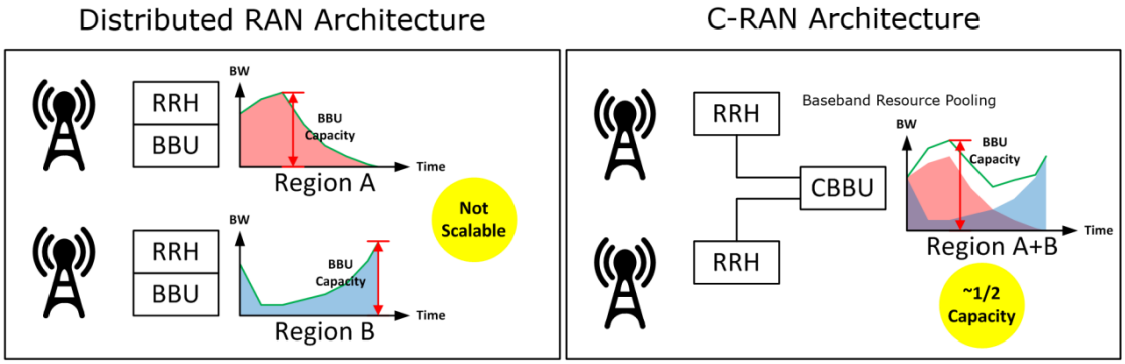


Figura 10: Comparación entre tipos de arquitectura [10].

Ahora bien, como se puede ver en la figura anterior, la arquitectura de C-RAN cambia con respecto a la arquitectura RAN distribuida. En el C-RAN el BBU y el RRH están separados físicamente. El RRH está instalado en la estación base donde se encuentra la antena y por ello cumple con funciones de radiofrecuencia y la BBU, por su parte, se encuentra situada en los

centros de red, por lo que varias unidades remotas pueden comunicarse con la agrupación de BBU.

Esta es la principal diferencia con la arquitectura tradicional, pero también su principal cualidad ya que gracias a esto se consigue, por un lado, que se aumente la eficacia y su capacidad de tráfico de datos y por otro que se reduzcan los costes como se ha explicado anteriormente.

Otro punto importante que señalar son las conexiones entre esta nueva arquitectura RAN. Al igual que ha sucedido con evolución de parte radio de la red, las comunicaciones entre todos los componentes también han cambiado [12].

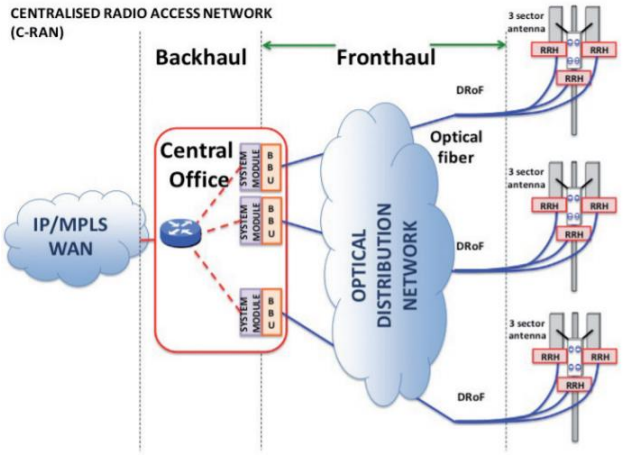


Figura 11: Aruitectura de C-RAN con segmento de fronthaul y backhaul [12]

Con la centralización del BBU y su separación del RRH nace un novedoso concepto de *fronthaul*, que es la parte que conecta los elementos radiantes con las BBU. Hasta ahora, con la arquitectura tradicional de RAN solo estaba presente el segmento *backhaul* que permitía la conexión entre el centro de redes y la estación base. Pero con la centralización del RAN, el segmento *backhaul* pasa a tener nuevas funciones encargándose de las comunicaciones entre el centro de red y el nodo de conexión a la WAN de transporte.

C-RAN combina muchos avances tecnológicos existentes en las comunicaciones inalámbricas, como es la fibra óptica o el estándar CPRI, un protocolo que permite mayores distancias en la transmisión de la señal de banda base lo que provoca el despliegue a gran escala de estaciones base centralizadas. Esto se aplica a la red del centro de datos para conseguir costes bajos, alta confiabilidad, baja latencia y gran ancho de banda en la red de interconexión de la asociación de BBU.

Para lograr la asignación dinámica de recursos compartidos en el centro de BBUs, se utiliza una plataforma abierta y tecnologías de virtualización en tiempo real basadas en la computación en la nube ya que gracias a esto la red puede soportar a múltiples proveedores y múltiples tecnologías. Además, con la centralización de BBU se consigue una mayor densidad de cobertura gracias al despliegue a gran escala de RRH que pueden estar conectados al grupo de BBU.

2.3 New Radio (NR)

A medida que LTE maduraba, los operadores móviles comenzaron a considerar cuidadosamente la posibilidad de actualización de la red. La aparición del 5G con la *Realase 15*, aprobada por 3GPP, se proporcionaron las primeras capacidades de banda ancha móvil mejoradas. Por ello, a diferencia de LTE, el objetivo principal de 5G no solo es aumentar las velocidades de datos y la capacidad de la red, sino que promete una revolución industrial ya que la red 5G se adapta a las condiciones del entorno y no al revés [13].

Se espera que el 5G aporte a los usuarios una mejora sustancial en términos de velocidad de datos y latencia ultra baja. Además, de mejorar los servicios de banda ancha móvil (eMBB), 5G en la *Release 16* presenta nuevas aplicaciones para su red mediante uRLLC (comunicaciones ultra fiables con baja latencia) y mMTC (comunicaciones masivas entre máquinas) [14].

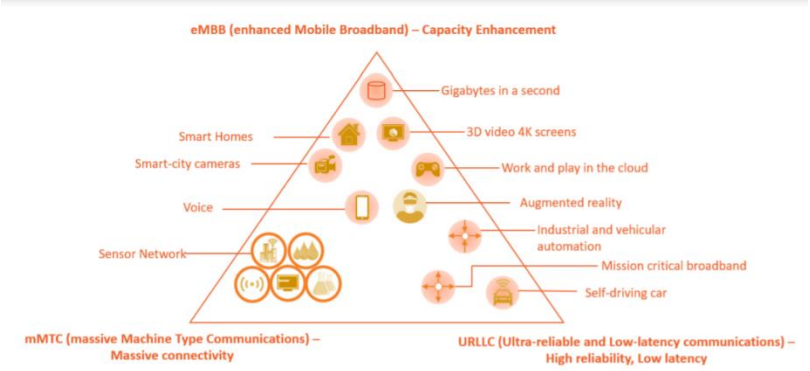


Figura 12: Servicios de 5G [15] .

La introducción del 5G es la oportunidad para que muchos operadores móviles creen, integren y actualicen su infraestructura para enriquecer los servicios y permitir la creación de nuevos casos de uso de la red. Muchos de MNOs quieren aprovechar al máximo el despliegue ya hecho para que el coste sea mínimo pero el funcionamiento sea óptimo.

Por el momento, los operadores pueden elegir entre dos modos de implementación de red, por un lado, la posibilidad de crear una red 5G SA (*Stand-Alone*), independiente y completa o bien la posibilidad de reutilizar la infraestructura de 4G y desplegar una red 5G NSA (*Non Stand Alone*) [16]. Ambos escenarios están representados en la siguiente figura y especificados en la *Release 15* desarrollada por 3GPP entre 2018-2019 [17].

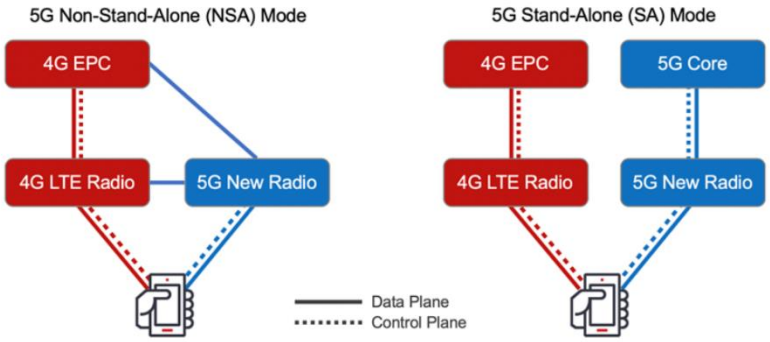


Figura 13: Modos de despligue de 5G [18]

El primero de los modos es el conocido como 5G- NSA, en el cual la RAN está formada por eNB (4G), que se comporta como el nodo maestro, y un gNB (5G), que actúa de nodo secundario y que se encarga de gestionar el tráfico en el plano de usuario. Ambos elementos están interconectados por la interfaz x2.

En 5G-NSA no se incorporan todas las cualidades que inicialmente presenta 5G, ya que este tipo de implementación está diseñado para el caso de uso de eMBB no incluyendo otras capacidades como la capacidad de trocear la red en subredes (*Network Slicing*), calidad de servicio QoS o latencia ultra baja que presenta NR.

Respecto al escenario 5G-SA, el RAN es completamente independiente al eNB y su implementación no depende del núcleo de red de 4G. De esta forma, todo el tráfico generado en la red es gestionado por sistemas 5G completos. Del mismo modo, 5G-SA puede tener interacción con 4G a través del núcleo de red.

Finalmente, con el desarrollo de SA se proporcionará el rendimiento de extremo a extremo para cumplir con todas las características de NR. Esto será posible gracias a *Network Slicing*, a la calidad de servicio QoS y a los tres casos de uso (eMBB,uRLLC, mMTC). Por lo que la *Release* 16 marcará los pasos a seguir para hacer una red 5G completa e independiente [18].

2.3.1 Arquitectura de Red de NR (New Radio)

Las generaciones anteriores de redes móviles de acceso por radio RAN se basaban en funciones de red tradicionales correspondientes a los elementos físicos de la red. Desde el inicio de la estandarización de 5G, el grupo de trabajo 3GPP RAN acordó un nuevo concepto fundamental para 5G RAN y es la división arquitectónica en módulos básicos. En este nuevo concepto las funciones de red NF (*Network Functions*) se definen con la granularidad adecuada para ambos planos el de control y el de usuario.

La definición de NF tanto para la RAN como para la red central permitirá crear una red modular de extremo a extremo para redes 5G. Este enfoque proporcionará a los operadores una calidad de servicio QoS garantizada de extremo a extremo más fácil y posibilitará a los operadores avanzar hacia una red basada en software que utiliza hardware *White box*, facilitando las actualizaciones futuras ya que esta división de redes es una de las capacidades más importantes de 5G en el futuro [15].

Bajo esta premisa, está claro que la red de acceso por radio RAN tiene que evolucionar para adaptarse a las necesidades de 5G. Entre las arquitecturas de RAN encaminadas hacia el 5G, el C-RAN es uno de los elementos esenciales de NR (*New Radio*) debido a su división de módulos. Esta arquitectura centralizada ha demostrado sus ventajas en el rápido despliegue de la red, el ahorro de diversos costes y la eficiencia energética como se ha visto previamente.

En la era del 5G, C-RAN ha evolucionado por sí misma con la inclusión de características como la virtualización, redes definidas por software y nuevas soluciones *fronthaul*. La idea básica de C-RAN es centralizar las unidades de procesamiento, virtualizarlos en un grupo de recursos y asignarlos dinámicamente según las necesidades.

En la Figura 14, se muestra el paso de una arquitectura de red 4G a un RAN de 5G, en el que la parte radio de NR es dividida en la unidad central CU (*Central Unit*) y la unidad distribuida DU (*Distributed Unit*). [19] Pero no solo se produce ese cambio, el *eNB* pasa a llamarse gNB (*GNode B*) el cual está formado por tres unidades lógicas CU, DU y RRH. Para entenderlo, se puede ver cada un gNB como un mini-C-RAN [14].

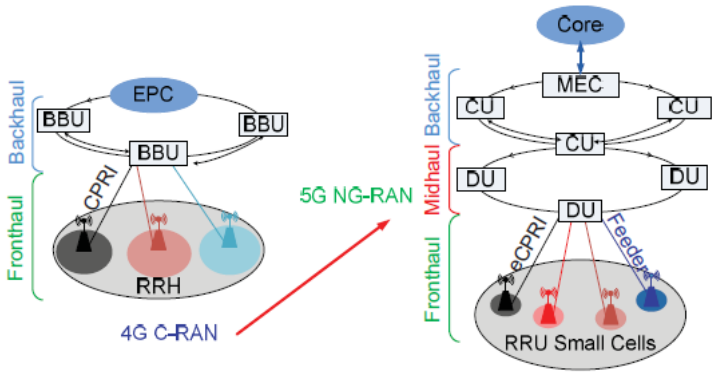


Figura 14: Transición de 4G a 5G [14].

La aparición de nuevos componentes en la arquitectura de red de 5G están motivados por un requisito muy necesario y contemplado en la *Release 15* que es la baja latencia. Esta división del BBU, tiene como objetivo la virtualización de la parte RAN del 5G para de esta forma conseguir uno de los servicios que pretende dar 5G [20].

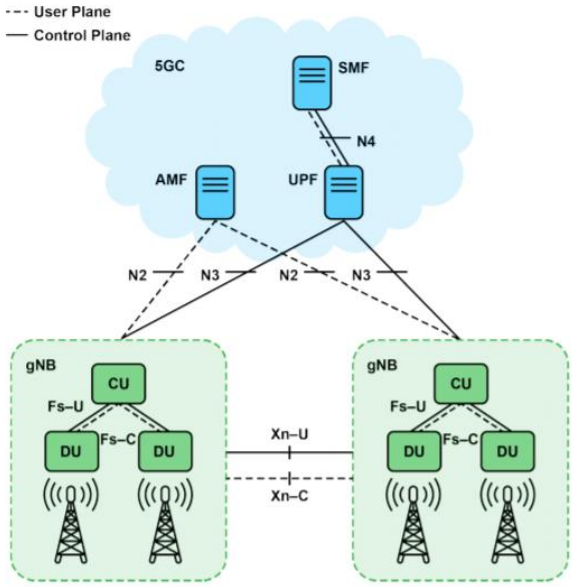


Figura 15: Arquitectura de red de 5G [15].

La arquitectura 3GPP, desarrollada en la *Release 15*, de la red 5G se muestra en la Figura 15. El NGRAN (*New Generation RAN*) consta de un conjunto de gNB conectados al núcleo del 5G a través de la interfaz NG y los gNBs, se pueden conectar entre ellos mediante la interfaz Xn. La composición de cada gNB consiste en una unidad centralizada gNB-CU y una o varias cada unidades distribuidas gNB-DU, conectadas entre sí mediante las interfaces Fs. Esta arquitectura dividida permite a la red 5G utilizar diferentes distribuciones de protocolo de pila entre CU y DU dependiendo del *fronthaul* disponible y de los criterios de diseño de red.

Como parte del marco 3GPP, la división funcional dinámica entre CU y DU presenta diferentes escenarios en los que se van a establecer la posición de las diferentes entidades que forman parte de la RAN. Si bien las CU mantienen funcionalidades parecidas al BBU en todo tipo de división será el DU quien adquiera diferentes funciones según el tipo de división.

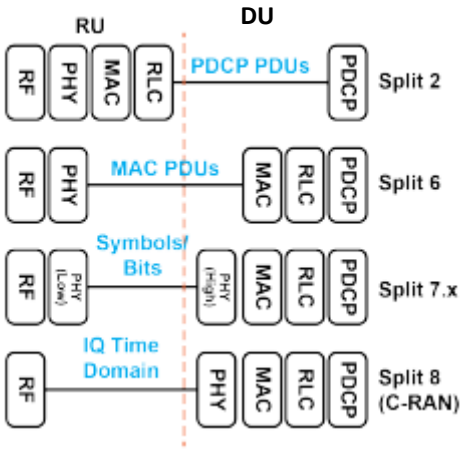


Figura 16: Escenarios de división de RU y DU [15].

En la anterior figura se pueden las diferentes divisiones que 3GPP ha estandarizado. Partiendo del C-RAN de 4G hasta llegar a las distintas posibilidades que ofrece 5G, debido a que uno de los objetivos que se quiere conseguir con estas separaciones, es mover cada vez más funciones de procesamiento de datos más cerca de *edge*, especialmente las funciones en tiempo real.

La incorporación de más inteligencia en *edge* permite a la RAN hacer frente al crecimiento de los datos móviles o proporcionar conexiones de menor latencia más cerca del usuario. Bajo este contexto, cabe destacar que *edge* se refiere a la computación de filo o en el borde, cuya función es procesar los datos de los dispositivos muy próximos al lugar donde se generan en vez de transportar la información una y otra vez a los centros de procesamiento.

Este nuevo tipo de arquitectura hace posible implementar las funciones de CU y DU relacionadas con un servicio específico, ya sea de manera centralizada o cerca de RU, lo que proporciona mayor flexibilidad para satisfacer diversas necesidades de servicio con la misma arquitectura RAN [21].

2.4 Open RAN

Tradicionalmente, los sistemas de radio RAN son propiedad de un solo proveedor por lo que un operador no puede desplegar una red usando la tecnología de otro proveedor distinto al suyo, ya que combinar y mezclar sitios celulares de diferentes proveedores conduce a una reducción del rendimiento. El resultado, es que la mayoría de los operadores de la red han desplegado redes con un único proveedor, a pesar de que podían soportar tecnologías de otros [22].

Este es uno de los motivos por el que nace las redes abiertas Open RAN con el fin de construir redes usando componentes de distintos proveedores. Pero es que, además, a medida que aumenta el tráfico móvil, las redes móviles y los equipos que la conforman deben transformarse en soluciones software, virtualizadas, flexibles, inteligentes y energéticamente eficientes.

La Alianza O-RAN está comprometida con la evolución del acceso por radio, haciendo las redes más abiertas e inteligentes que las generaciones anteriores. Características como el análisis en tiempo real, o los módulos de *backend* de inteligencia artificial, impulsan los sistemas de aprendizaje automático que potencian la capacidad de esta arquitectura.

Los elementos de red virtualizados adicionales con interfaces abiertas y estandarizadas son aspectos clave de los diseños de referencia desarrollados por la Alianza O-RAN. Las tecnologías de código abierto y elementos de software inteligentes son algunas de las cualidades que presenta la alianza O-RAN en las redes inalámbricas de próxima generación. Esto se debe a que para la construcción de una RAN ágil y más rentable, se requieren modelos accesibles. Las interfaces abiertas son esenciales para permitir a los proveedores y operadores más pequeños introducir

rápidamente sus propios servicios o permitir a los operadores personalizar la red, con el fin de satisfacer sus propias necesidades. Las interfaces abiertas también permiten múltiples implementaciones de proveedores, lo que da lugar a un ecosistema más competitivo y dinámico. Del mismo modo, la presencia de diseños software y hardware de código abierto posibilita una innovación más rápida, democrática y sin permisos. Y en segundo lugar por la necesidad de que las redes sean capaces de ser autónomas. En este sentido, con la llegada del 5G, las redes se están volviendo cada vez más complejas con aplicaciones más ricas y exigentes, por lo que no se pueden usar los medios tradicionales de despliegue u optimización en una red.

Por este motivo la alianza O-RAN, incorpora inteligencia en todas sus capas de la arquitectura RAN. Esta inteligencia aplicada tanto a nivel de componentes como de red permite la asignación dinámica de recursos de radio y optimiza la eficiencia de esta. Aprovechar el nuevo aprendizaje basado en tecnologías para automatizar las funciones operativas de la red y así reducir costes es una de las principales características que tiene esta tecnología.

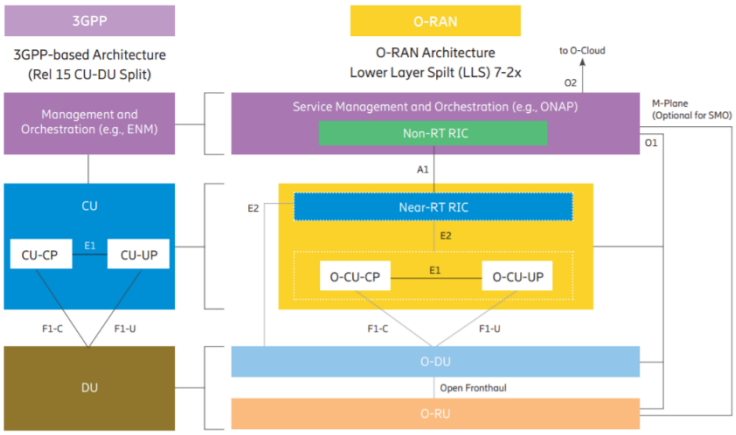


Figura 17: Comparación entre la arquitectura de 3GPP y la de O-RAN [23].

Asimismo, el proceso de 3GPP se centra en las especificaciones globales, por lo que la apertura de las interfaces entre elementos específicos dentro de la RAN, funciones adicionales o la red central es clave para desglosar la red y facilitar el acceso a proveedores adicionales. La alianza O-RAN está desarrollando especificaciones para estas interfaces abiertas de forma complementaria a los estándares promovidos por 3GPP para ciertas interfaces que no habían sido aprobadas por 3GPP.

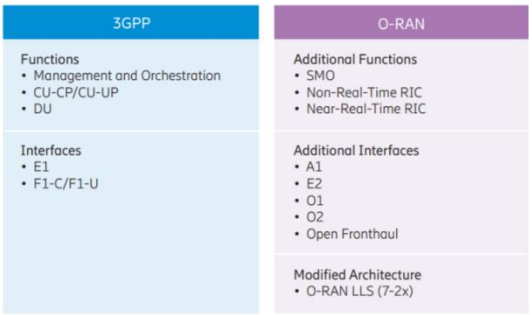


Figura 18: Comparación entre estándar 3GPP y O-RAN [23].

2.4.1 Arquitectura O-RAN

La arquitectura de referencia de O-RAN está diseñada para ser la infraestructura RAN de próxima generación. Definida por principios de inteligencia y apertura, la arquitectura O-RAN es la base para construir la RAN virtualizada en hardware abierto, con control de radio e impulsado por la inteligencia artificial.

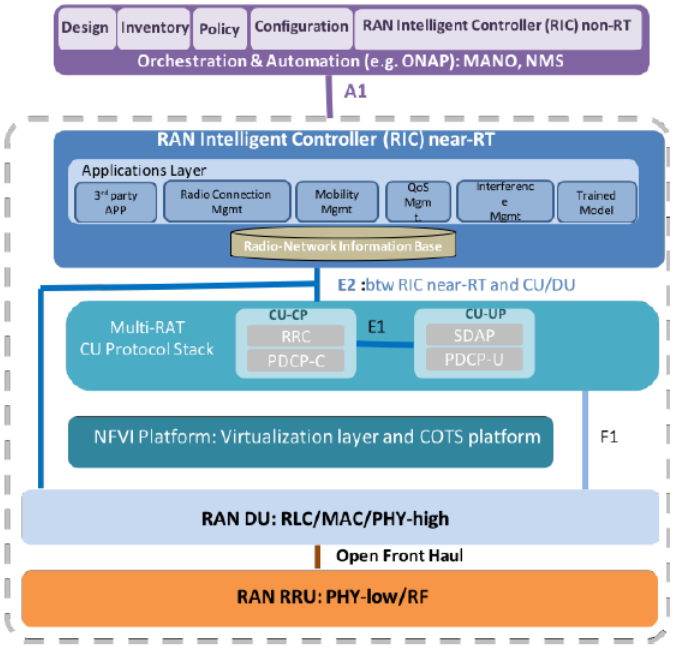


Figura 19: Arquitectura O-RAN [22]

La arquitectura se centra en interfaces definidas y estandarizadas para favorecer un suministro abierto e interoperable apoyado y complementario a los estándares promovidos por 3GPP. Para conseguirlo, O-RAN establece una serie de módulos funcionales entre los que destacan las capas de gestión y administración de redes, las capas inferiores (RRU, DU, CU) y las interfaces internas.

En primer lugar, la primera entidad que se encuentra en las capas más altas de la arquitectura es la plataforma SMO (*Service Management Orquestration*). En su interior se encuentra RAN Intelligent Controller (RIC) en tiempo no real (non-RT) que incluye las funciones en el servicio y gestión de políticas, análisis de RAN y formación de modelos para la funcionalidad de near-RT RAN mediante modelos entrenados.

La A1 es la interfaz entre la capa de orquestación que contiene el RIC non-RT y el *eNB*/ gNB que contiene el RIC-near RT. Con la introducción de A1, las aplicaciones de gestión de red en RIC-non-RT pueden recibir y actuar sobre los datos proporcionados por la CU y DU en un formato estandarizados, es decir, los mensajes generados a partir de políticas habilitadas para modelos de entrenamiento de IA que se transmiten a RIC near-RT.

El algoritmo de RIC non-RT tiene la capacidad de modificar los comportamientos de RAN mediante el despliegue de diferentes modelos optimizados para las políticas de cada operador y los objetivos de optimización.

La capa del controlador inteligente RAN (RIC) en tiempo real (*Near RT*) es el módulo que se encuentra entre la interfaz interna A1 y las capas inferiores correspondientes al gNB. Es una entidad lógica que permite el control casi en tiempo real y la optimización de un subconjunto de funciones de la gestión de recursos radio realizados por el gNB.

El RIC-near RT se compone de una plataforma software con aplicaciones de control conocidas como xApps. Cada aplicación puede habilitar el RIC para controlar una o múltiples funciones RRM, mediante el intercambio de datos entre las xApps y el gNB sobre la interfaz E2 [23].

A través de estas aplicaciones de control, el RIC puede controlar la movilidad y el equilibrio de carga mediante el intercambio de información entre una xApp específica o sobre el CU y la interfaz E2. También se puede controlar las políticas programación con tecnologías avanzadas entre estas aplicaciones y el DU.

Centrando la atención a las capas inferiores se pueden observar las tres partes fundamentales que forman parte de esta nueva arquitectura RU, DU y CU. Todas ellas, representan a las entidades tradicionales de arquitectura RAN, pero evolucionadas a las condiciones de este nuevo entorno.

Es destacable la capa de O-CU que presenta en dos planos diferentes, el plano de usuario y el plano de control, al igual que ocurría en el RAN tradicional, en O-RAN también se establece esta división.

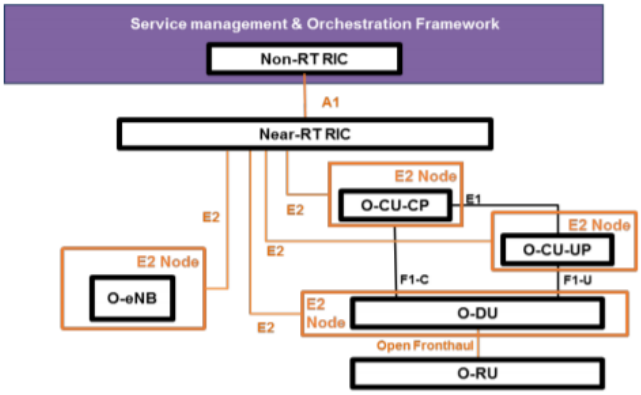


Figura 20: Arquitectura de O-RAN con las interfaces definidas [20].

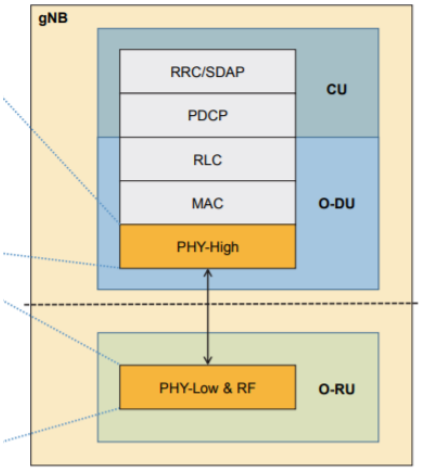


Figura 21: División de capas en la arquitectura O-RAN [33].

En cada plano se definen distintas funciones, si bien del lado del usuario se encarga de la entidad de SDAP (*Service Data Adaption Protocol*), el plano de control es responsable de RRC (*Radio Resource Control*).

La O-RU, como se conoce a RU en este tipo de arquitectura, se encarga de convertir las señales de radio enviadas hacia y desde la antena en una señal digital que se puede transmitir a las capas superiores a través de una red de paquetes. Los tres elementos principales de la RU son la función de radiofrecuencia (RF), el procesamiento PHY inferior para reducir el ancho de banda y la interfaz eCPRI (*Common Public Radio Interface*) para mapear y desmapear datos radio en el protocolo ethernet.

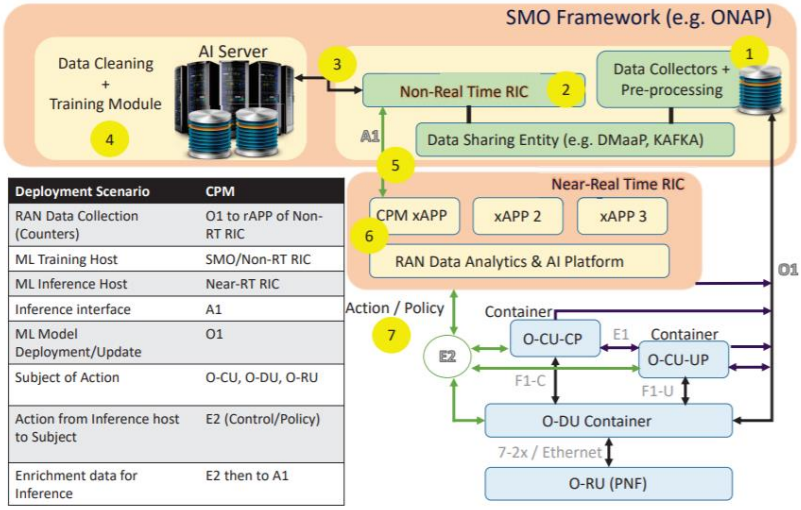
En relación con eCPRI cabe señalar que es la evolución de CPRI, una interfaz basada en el transporte de señales de radio digitales directamente sobre una fibra óptica que permitía la conexión entre el RU y BBU en 4G, y que en O-RAN establece la conexión con el O-DU.

Dado que ya se han considerado todas las partes imprescindibles para el desarrollo de la arquitectura O-RAN, es momento de describir cómo será la arquitectura de implementación en las capas más altas del esquema de O-RAN. De modo que, en la Figura 22 se refleja el recorrido entre las distintas entidades sobre un escenario genérico de implementación.

En primer lugar, la plataforma SMO es la responsable de recolectar los datos procedentes de las unidades CU y DU de las capas inferiores de la arquitectura O-RAN a través de la interfaz O1. Cabe resaltar que en este caso es la plataforma de automatización de redes abiertas ONAP quien se encarga de recolectar estos datos que serán compartidos con no RT RIC, el cual tratará estos datos y mediante mecanismos de Aprendizaje Automático entrenará al sistema para que cree algoritmos que automaticen los procesos de gestión de las redes.

El siguiente paso, consistirá en enviar un mensaje también conocido como inferencia al RIC near-RT, notificando los resultados y las políticas del mensaje en las que se hace una predicción de la congestión de la red de las capas más bajas. A través de la interfaz A1 se transportan estos datos

que son procesados entonces por las xApps donde se predice la ocurrencia de congestión para finalmente aplicar la solución correspondiente en DU y CU a través de la interfaz E2 [24].



Deployment Scenario	CPM
RAN Data Collection (Counters)	O1 to rAPP of Non-RT RIC
ML Training Host	SMO/Non-RT RIC
ML Inference Host	Near-RT RIC
Inference interface	A1
ML Model Deployment/Update	O1
Subject of Action	O-CU, O-DU, O-RU
Action from Inference host to Subject	E2 (Control/Policy)
Enrichment data for Inference	E2 then to A1

Figura 22. Estructura de alto nivel de una posible implementación de la arquitectura O-RAN [24].

2.4.2 Ecosistema de Open RAN

El concepto y el movimiento de Open RAN no es nuevo, ya que, durante los últimos años, los operadores móviles y proveedores tecnológicos ya han estado desarrollando soluciones, realizando ensayos y desplegando redes con estas características. Y es que uno de los puntos importantes que tiene esta nueva arquitectura es que está siendo implementada por los principales operadores en todo el mundo [25].

Aunque ya se conoce el concepto de redes Open RAN basado en interfaces abiertas que pueden ser ejecutadas mediante un hardware comercial, es importante conocer como se ha establecido de forma real. Por un lado, Open RAN hace referencia al grupo de proyectos que forman parte de *Telecom Infra Group* (TIP) cuyo objetivo principal era el despliegue de soluciones RAN y software desagregado. Y, por otro lado, la Alianza de O-RAN que se centraba en estandarizar interfaces internas entre las diferentes entidades.

La unión entre las dos organizaciones fue un paso muy importante en el desarrollo del ecosistema de Open RAN ya que permitió a los dos grupos, trabajar de manera conjunta, compartir información, integrar nuevas especificaciones y aunar esfuerzos con el fin de conseguir capacitar todas sus funcionalidades.

El desarrollo de soluciones Open RAN se ha hecho realidad en distintos proveedores de software y hardware mundiales. Destacan empresas como Aliostar, Mavenir, Parellel Wireless o Ericsson que han implementado O-RAN en redes comerciales y que son utilizados por distintas operadoras como Vodafone u Orange.

Según el informe realizado por la empresa iGR, existen 22 operadoras en todo el mundo que ya trabajan sobre redes Open RAN, las cuales son responsables del 21,8% de la base de los abonados móviles en todo el mundo. Además, suponiendo que las pruebas actuales se conviertan en implementaciones comerciales, se estima que para 2024, la arquitectura de Open RAN será desplegada todavía más, lo que supondrá el 47,2% del total de suscriptores globales [25].

Figura 23: Despliegue de Open RAN por distintas operadoras [26] .

Como se puede ver en la figura anterior, Open RAN ya no es una solución regional, ni una que se aplique únicamente para operadores totalmente nuevos, sino que junto con nuevos operadores y los grandes dueños del sector esta tecnología se está desplegando para dar un servicio de uso sin precedentes.

Capítulo 3. Consideraciones previas

En este capítulo se presentan las líneas generales de la solución propuesta para este trabajo. Por otro lado, se describe de forma detallada que equipos se van a usar y la configuración inicial necesaria para dar cabida estos sistemas de la arquitectura O-RAN. Además, se establecen los requisitos mínimos en los componentes externos al sistema.

3.1 Descripción General

El objetivo del trabajo es implementar dos entidades que forman parte de la arquitectura de O-RAN. La propuesta se basa en desarrollar el non-RIC y RIC near-RT, dos entidades parecidas por la forma en la que están implementadas, pero totalmente diferentes en su funcionamiento y en sus casos de uso.

El proyecto de O-RAN está disponible en la comunidad software de O-RAN, la cual es una colaboración entre la Alianza de O-RAN y la fundación de Linux para la creación de un software RAN de código abierto. De esta forma, el conjunto inicial de proyectos que dispone la plataforma está formados por el non RT RAN, el controlador inteligente casi en tiempo real RIC near-RT y las unidades que forman parte del gNB.

Si bien destaca esta comunidad, es porque se apoya en dos tipos de proyectos, por un lado, la comunidad software que utiliza la licencia de Apache 2 para contribuciones de software de código abierto y, por otro lado, la licencia propia de O-RAN que proporciona las licencias esenciales para su despliegue. Conocido esto, la comunidad O-RAN presentó una primera versión de código abierto en noviembre de 2019, bautizada con el nombre de *Ambar* que asentó el comienzo de un nuevo modelo de la arquitectura RAN. Pero sus continuos avances hicieron que en 2020 presentarán una nueva versión llamada *Bronze* que fue completada en junio de 2020, y por este motivo, este proyecto está basado en la *Release Bronze.*

Bajo esta premisa, el despliegue de estas dos entidades se lleva a cabo para cumplir con los principales objetivos, que en esta versión se basan principalmente en el desarrollo de dos casos de uso denominados *Health Check* y *Traffic Steering.* Ambos dos, hacen comprender el alcance O-RAN y sus posibilidades de despliegue. Para hacer realidad la implementación y el modelado de estos dos casos de uso no solo es necesario implementar el SMO y el RIC, sino que también es necesario llevar a cabo dos implementaciones diferentes [27].

En ambos módulos, aparecen una serie de conceptos que es necesario conocer. En primer lugar, la tecnología de Docker, basada en contenedores que permiten ejecutar varios procesos por separado gracias a la modularidad de su sistema y a la gran flexibilidad que presentan estos módulos a la hora de optimizar sus aplicaciones que son totalmente virtuales de tal forma que su papel será de gran importancia en el desarrollo dada sus características. En segundo lugar, la plataforma software de código abierto ONAP que permite la orquestación y automatización a tiempo real de distintas partes de las entidades de SMO (*Service Management Orcherstation*) gracias a que permite la sincronización.Y en último lugar, destaca el grupo de *kubernetes* (*k8s cluster*) cuya característica esencial es la capacidad de programar contenedores dentro de distintas máquinas virtuales ya que cada agrupación tiene un estado deseado que permite su configuración a través de archivos tipo *yaml* Finalmente, estos contenedores, que en *Kubernetes* se llaman *pods* serán imprescindibles para el correcto uso de cada una de las entidades de O-RAN [28].

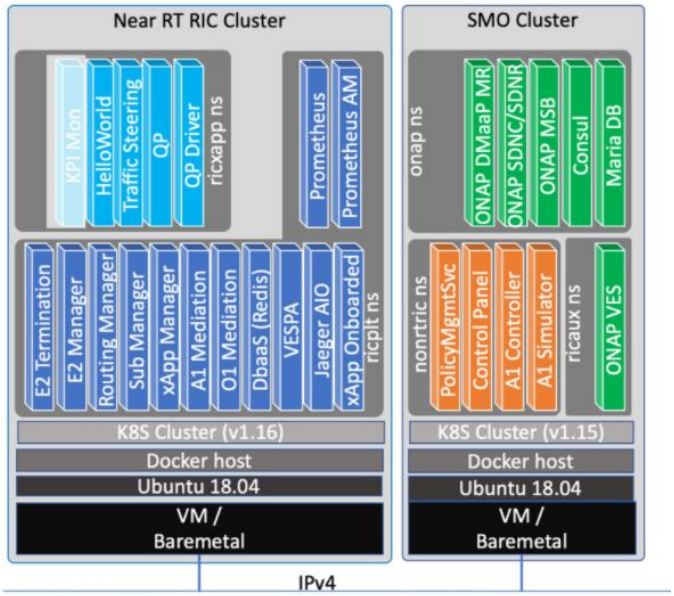


Figura 24: Contenedores que forman parte de SMO y RIC [27].

Cada entidad, por tanto, está formada por una serie de contenedores que almacenan la información necesaria para configurar cada una de las secciones. Mediante dos máquinas virtuales se implementarán estas entidades para después completar el proyecto con la realización de los dos casos de uso.

A medida que esta nueva arquitectura RAN avanza, es importante definir en qué consisten los casos de uso que enfocarán las implementaciones hacia una estructura totalmente operativa. En estos primeros estadios, se establece el caso de uso *Health Check* que permite implementar un conjunto inicial de capacidades que se puedan monitorizar y comprobar el estado de Open RAN [29].

De tal forma, que el establecimiento de la interfaz A1 entre las dos entidades va a permitir la realización de este caso de uso gracias a la comunicación que existen entre los diferentes contenedores que definen a cada entidad.

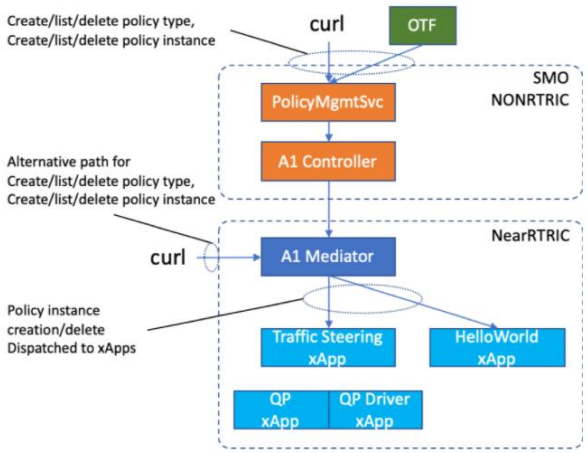


Figura 25: Flujo de datos entre SMO y RIC a través de la interfaz A1.

En la imagen anterior se puede ver cómo se va a establecer la comunicación entre ambas entidades. Partiendo de su implementación que se detallará en la siguiente sección hasta llegar a

la comunicación entre ambas mediante interfaces virtuales y su comprobación mediante una interfaz de usuario web.

Para llevar a cabo el siguiente caso de uso conocido como *Traffic Steering* es necesario recurrir al despliegue de xApps dentro de la entidad RIC, cuyas funciones van a estar relacionadas con el análisis, la recopilación de datos ya que en esta versión se centran en la integración y pruebas a nivel muy básico incluyendo las primeras versiones de 4 xApps para este caso uso.

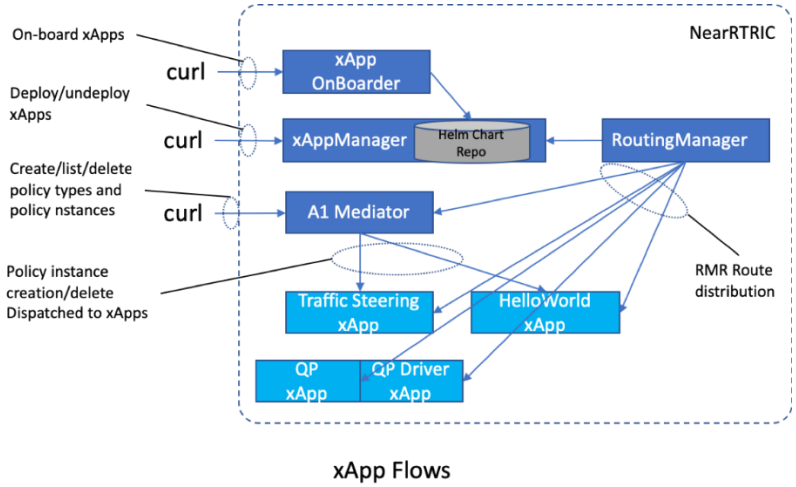


Figura 26: Flujo de datos de xApps en el RIC.

El objetivo clave es que los elementos de O-RAN puedan realizar autocontroles de chequeo para ver la correcta funcionalidad e implementación de este, y verificar el estado a través de los resultados de implementación. Para ello, se establecen distintos flujos para validar el correcto funcionamiento de las entidades y de sus interfaces. Este proyecto se centra en desarrollar aquellos que tienen que ver con las entidades de RIC y de SMO.

Cumplir estos objetivos pone las bases para conseguir una red modular, intercomunicada y flexible que haga que el RAN se convierta en una herramienta virtual y de código abierto. Aunque todavía los proyectos accesibles tienen un alcance limitado, su estandarización y puesta en marcha es un gran avance para esta nueva tecnología.

Por otra parte, el proyecto va a constar de la simulación en la que RIC se conecta con la entidad de *O-DU high* mediante la interfaz E2. Antes de seguir avanzando cabe resaltar que la *Release Bronze* ha divido la unidad distribuida en dos entidades, por un lado, se encuentra las capas altas de o-du y por otro las capas más bajas como se puede ver en la Figura 27. En este caso, se va a establecer comunicación a través de la interfaz definida por la *Release* como E2AP, en la cual se intercambian mensajes genéricos, estableciéndose como un punto de preespecificación de esta interfaz.

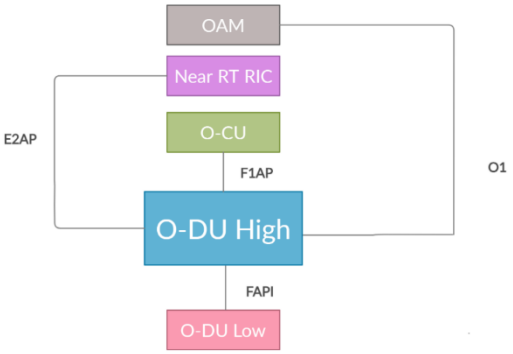


Figura 27: Interfaz E2AP entre RIC y O-DU high.

Por lo que, conocer el entorno de trabajo donde encuentra toda su documentación y configuración es el primer paso antes de la instalación de ambas entidades y del desarrollo de los dos casos de uso disponibles en esta versión *Bronze*. Como O-RAN es una comunidad de software de código abierto existen distintas formas en las que se puede localizar su código y su manual de usuario.

Para este proyecto, el acceso a la instalación viene proporcionado por *Git Hub*, una herramienta de flujo de trabajo y de almacenamiento de código y para el manual de usuario, se usa *Confluence*, donde se encuentra toda la información relevante para entender su funcionamiento.

3.1.1 Requisitos mínimos

Para poder llevar a cabo este proyecto, se requiere en un mismo entorno de trabajo distintos elementos que permitan que la conexión y la implementación de ambos módulos sea posible. Por ello, el equipo debe tener una serie de requisitos que garantizará un correcto funcionamiento.

	SMO	Near RT RIC
Sistema Operativo	Ubuntu 18.04 LTS	
Procesador	8 núcleos	4 núcleos
Memoria RAM	32GB	16GB
Almacenamiento	160GB	160GB

Tabla 1: Requisitos de O-RAN

En el manual de usuario de O-RAN se establecía el uso de dos máquinas virtuales que permitieran desarrollar cada uno de los sistemas que forma parte de la RAN. Estos mismos manuales determinaban cuales eran los requisitos mínimos. Se debe tener en cuenta que, al ser dos módulos diferentes, las condiciones para su ejecución también lo eran. Así en la Tabla 1 se muestran los distintos requisitos de ambas instalaciones [30].

Respecto a las necesidades de cada sistema varían en función de las especificaciones del manual en el que se obtiene el código abierto de implementación. En el caso de SMO se requieren como mínimo 8 núcleos de red y 32 de memoria RAM [31] . Sin embargo, en el caso de la Near RT RIC, estas condiciones mínimas de uso son menos restrictivas con la mitad de los núcleos necesarios. Para el enrutamiento de red se hará de manera virtual.

3.1.2 Especificaciones del entorno de trabajo

Para el desarrollo de este proyecto se utiliza el servidor rack *SuperServer1029PWTR* de la empresa *Super Micro*, donde sus características más importantes en concordancia con las condiciones necesarias para el proyecto se recogen en la Tabla 2.

Sistema Operativo	Ubuntu 18.04 LTS
Procesador	Intel Xeon Silver 4216 (32 núcleos @ 2,10 GHz)
Memoria RAM	64 GB
Almacenamiento	960 GB SSD

Tabla 2: Requisitos del servidor de trabajo

Con respecto a la estación de trabajo, la máquina que se encuentra en el laboratorio del Grupo de Comunicaciones Móviles (MCG) del iTEAM, recibe el nombre de “mcg” y trabaja sobre el sistema operativo especificado anteriormente, que es Ubuntu 18.04 LTS. En la siguiente figura se puede ver el entorno donde se desarrolla el proyecto.

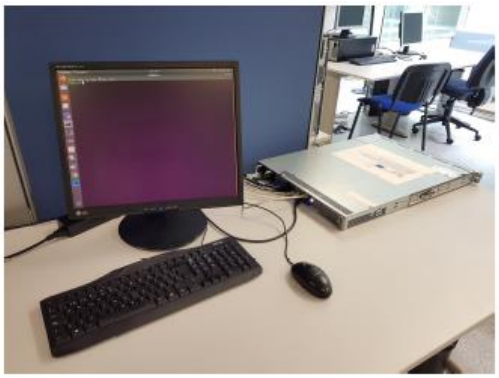


Figura 28: Equipo de Trabajo *SuperServer* 1029

Si bien ya se ha presentado el entorno de trabajo físico, es importante presentar cuales van a ser las herramientas virtuales que se van a usar como medio de trabajo.

3.2 Previa instalación

Antes de comenzar con la instalación de las entidades de O-RAN es necesario implementar las máquinas virtuales de cada módulo. En este caso como se ha comentado anteriormente se ha decidido que la mejor máquina virtual es KVM ya que su instalación es muy intuitiva.

Para su puesta en marcha, se accederá través de la terminal de Ubuntu del equipo "mcg" y se ejecutarán los siguientes comandos para iniciar su instalación. Teniendo en cuenta que el comando sudo solo indica una serie de permisos, estos son los comandos indicados para la instalación de todos los paquetes necesarios. Una vez instalados es necesario reiniciar el servidor.

```
$ sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
```

El siguiente paso consiste en instalar una herramienta gráfica que permita gestionar las nuevas máquinas virtuales. En esta ocasión, se ha utilizado Virtual Machine Manager (VMM) para realizar estas funciones. Mediante esta aplicación nos permite establecer el nombre de la máquina, sus características detalladas anteriormente y configurar la red.

Por otro lado, la asignación de nombres de cada viene determinado por la entidad que se va a ejecutar en cada una de ellas, así en la figura siguiente se puede ver la instalación completa. Al necesitar distintas máquinas virtuales cada una de ellas se instalará de forma análoga a la anterior.

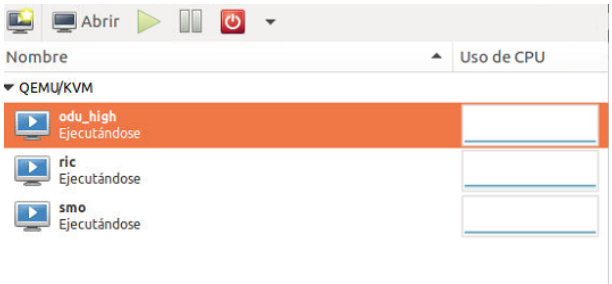


Figura 29: Gestor de máquinas virtuales VMM

Capítulo 4. Despliegue de nonRT RIC

En esta sección se lleva a cabo la explicación completa del despliegue de nonRT RIC también conocido como SMO, dentro del proyecto desarrollado por la Alianza O-RAN. Para ello, es necesario describir los pasos a seguir para su despliegue y configuración posterior.

Partiendo de la base de que el objetivo principal es respaldar la optimización de la RAN mediante orientación basadas en políticas, gestión de modelos de aprendizaje automático u optimización de los recursos radio en la RAN. Pero, sin embargo, el alcance de esta entidad hasta el momento es limitada ya que en esta versión presenta un diseño inicial para el modelado y la implementación de la plataforma de SMO.

4.1 Instalación SMO

En primer lugar, se establecerá el usuario *root* para realizar la instalación y poder obtener las ventajas que ofrece al no necesitar el comando sudo para los permisos. De esta forma en la terminal de la máquina virtual SMO se indican los siguientes comandos:

```
$ sudo -i
$ git clone http://gerrit.o-ran-sc.org/r/it/dep -b bronze
```

La rama donde se encuentra el repositorio de instalación de *Git Hub* es *dep* por lo que una vez descargado, se accede a él y se obtienen los scripts y gráficos de implementación.

```
$ cd dep
$ git submodule update –init –recursive --remote
```

Ahora es necesario generar un script que permitirá configurar el clúster de kubernetes en donde se instala los componentes de SMO. Hay que tener en cuenta que la versión de ONAP requerida para la instalación de SMO es diferente por lo que es necesario editar el archivo llamado infra.rc. para acceder a él lo haremos a través de la siguiente ruta:

```
$ cd tools/k8s/etc
$ nano  ./infra.rc
```

Se procede entonces a seleccionar la versión de ONAP Frankfurt en la que se describe la versión utilizada en cada uno de los elementos necesarios para la generación de los scripts de instalación, quedando de esta manera el archivo como se puede ver en la siguiente figura.

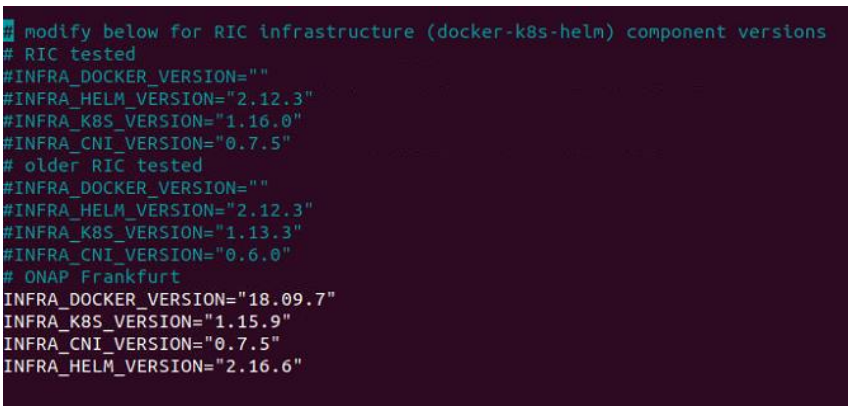


Figura 30: Selección de versiones para SMO

Después, es necesario crear el script que contendrá toda la información para la posterior instalación de Kubernetes, de Helm y de Docker. Con este archivo se podrá preparar el clúster k8s para que posteriormente sea integrado en el near-RT RIC.

```
$ cd tools/k8s/bin
$ ./gen-cloud-init.sh
$ ./k8s-1node-cloud-init-k_1_15-h_2_16-d_18_09.sh
```

Antes de realizar la instalación del clúster k8s era imprescindible para que se completase la ejecución, modificar dos parámetros dentro del propio script de k8s. En primer lugar, fue necesaria cambiar la versión del Docker ya que la que inicialmente se indicaba en el script infra.sr no coincidía con la que contenía el archivo y en segundo lugar la versión de Ubuntu del archivo tampoco coincidía por lo que hubo que adaptarla a la adecuada.

```
INFRA_DOCKER_VERSION="19.03.6"
#INFRA_HELM_VERSION="2.12.3"
#INFRA_K8S_VERSION="1.13.3"
#INFRA_CNI_VERSION="0.6.0"
# ONAP Frankfurt
#INFRA_DOCKER_VERSION="18.09.7"
INFRA_K8S_VERSION="1.15.9"
INFRA_CNI_VERSION="0.7.5"
INFRA_HELM_VERSION="2.16.6"
```

Figura 31. Cambio de la versión de Docker.

```
UBUNTU_RELEASE=$(lsb_release -r | sed 's/^[a-zA-Z:\t ]\+//g')
if [[ ${UBUNTU_RELEASE} == 16.* ]]; then
  echo "Installing on Ubuntu $UBUNTU_RELEASE (Xenial Xerus) host"
  if [ ! -z "${DOCKERV}" ]; then
    DOCKERVERSION="${DOCKERV}-0ubuntu1~16.04.5"
  fi
elif [[ ${UBUNTU_RELEASE} == 18.* ]]; then
  echo "Installing on Ubuntu $UBUNTU_RELEASE (Bionic Beaver)"
  if [ ! -z "${DOCKERV}" ]; then
    DOCKERVERSION="${DOCKERV}-0ubuntu1~18.04.2"
  fi
else
  echo "Unsupported Ubuntu release ($UBUNTU_RELEASE) detected.  Exit."
  exit
fi
```

Figura 32. Cambio de la versión de Ubuntu

Una vez resueltos los problemas, se puede comprobar como su funcionamiento es el correcto ya que los 9 contenedores que pertenecen a la agrupación de k8s están en estado *running*. Para su comprobación se utiliza el comando siguiente.

```
$ kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-5d4dd4b4db-5tw7f	1/1	Running	6	6d23h
kube-system	coredns-5d4dd4b4db-tqdf2	1/1	Running	6	6d23h
kube-system	etcd-mcg	1/1	Running	6	6d23h
kube-system	kube-apiserver-mcg	1/1	Running	6	6d23h
kube-system	kube-controller-manager-mcg	1/1	Running	6	6d23h
kube-system	kube-flannel-ds-bcw76	1/1	Running	53	6d23h
kube-system	kube-proxy-tlxkk	1/1	Running	6	6d23h
kube-system	kube-scheduler-mcg	1/1	Running	6	6d23h
kube-system	tiller-deploy-666f9c57f4-rl5m8	1/1	Running	7	6d23h
ricxapp	ricxapp-hwxapp-6d49b695fb-lrdkk	1/1	Running	5	6d22h

Figura 33: Estado de los contenedores en la instalación de SMO

A simple vista, parece que el entorno de ejecución para la instalación de SMO está listo, sin embargo, aún faltaba un paso más. Con respecto a los gráficos de ONAP faltaba un paso

fundamental que hacía que no se ejecutara de forma completa la puesta marcha. Mediante una serie de mensajes de aviso en el terminal, el propio sistema indicaba que no podía finalizar y entraba en bucle.

```
===> Deploying OAM (ONAP Lite)
=======> Deploying ONAP-lite
Error: unknown command "deploy" for "helm"
Run 'helm --help' for usage.
======> Waiting for ONAP-lite to reach operatoinal state
0/7 SDNC-SDNR pods and 0/7 Message Router pods running
0/7 SDNC-SDNR pods and 0/7 Message Router pods running
```

Figura 34: Error en la instalación de SMO.

La solución fue cargar de manera manual todos gráficos de ONAP para que estuvieran presentes en la instalación final. Así, primero se llevó a cabo la ejecución de los gráficos necesarios y a continuación se realizó la instalación final.

```
$ cd dep/smo/bin/smo-deploy/smo-oom/kubernetes
$ sudo make all
$ cd dep/smo/bin
$ ./install
```

Finalmente, se obtuvo el resultado esperado ya que mediante el comando anteriormente especificado para comprobar el éxito de la ejecución se mostraba 8 contenedores de *nonrtric*, 27 contenedores de ONAP y 2 contenedores de *ricaux* todos ellos en el estado *running*.

```
NAMESPACE     NAME                                        READY   STATUS              RESTARTS   AGE
kube-system   coredns-5d4dd4b4db-5tw7f                    1/1     Running             6          7d5h
kube-system   coredns-5d4dd4b4db-tqdf2                    1/1     Running             6          7d5h
kube-system   etcd-mcg                                    1/1     Running             6          7d5h
kube-system   kube-apiserver-mcg                          1/1     Running             6          7d5h
kube-system   kube-controller-manager-mcg                 1/1     Running             6          7d5h
kube-system   kube-flannel-ds-bcw76                       1/1     Running             55         7d5h
kube-system   kube-proxy-tlxkk                            1/1     Running             6          7d5h
kube-system   kube-scheduler-mcg                          1/1     Running             6          7d5h
kube-system   tiller-deploy-666f9c57f4-rl5m8              1/1     Running             7          7d5h
nonrtric      a1-sim-osc-0                                1/1     Running             0          40s
nonrtric      a1-sim-osc-1                                0/1     Running             0          16s
nonrtric      a1-sim-std-0                                1/1     Running             0          40s
nonrtric      a1-sim-std-1                                0/1     Running             0          17s
nonrtric      a1controller-64c4f59fb5-ntlg5               0/1     Running             0          40s
nonrtric      controlpanel-8bd4748f4-cmj24                1/1     Running             0          40s
nonrtric      db-549ff9b4d5-kh8n6                         1/1     Running             0          40s
nonrtric      policymanagementservice-64d4f4db9b-c8pmf    1/1     Running             0          40s
onap          dev-consul-68d576d55c-29sqh                 1/1     Running             0          7m28s
onap          dev-consul-server-0                         1/1     Running             0          7m28s
onap          dev-consul-server-1                         1/1     Running             0          7m6s
onap          dev-consul-server-2                         1/1     Running             0          6m56s
onap          dev-kube2msb-9fc58c48-68nt8                 1/1     Running             0          7m24s
onap          dev-mariadb-galera-0                        1/1     Running             0          7m25s
onap          dev-mariadb-galera-1                        1/1     Running             0          6m10s
onap          dev-mariadb-galera-2                        1/1     Running             0          5m28s
onap          dev-message-router-0                        1/1     Running             0          7m26s
onap          dev-message-router-kafka-0                  1/1     Running             1          7m26s
onap          dev-message-router-kafka-1                  1/1     Running             1          7m26s
onap          dev-message-router-kafka-2                  1/1     Running             1          7m26s
onap          dev-message-router-zookeeper-0              1/1     Running             0          7m26s
onap          dev-message-router-zookeeper-1              1/1     Running             0          7m26s
onap          dev-message-router-zookeeper-2              1/1     Running             0          7m26s
onap          dev-msb-consul-65b9697c8b-g4cxl             1/1     Running             0          7m24s
onap          dev-msb-discovery-54b76c4898-9l9gv          2/2     Running             0          7m24s
onap          dev-msb-eag-76d4b9b9d7-xlmbz                2/2     Running             0          7m24s
onap          dev-msb-iag-65c59cb86b-2qdgc                2/2     Running             0          7m24s
onap          dev-sdnc-0                                  2/2     Running             0          7m16s
onap          dev-sdnc-db-0                               1/1     Running             0          7m16s
onap          dev-sdnc-dmaap-listener-5c77848759-kphqn    1/1     Running             0          7m16s
onap          dev-sdnc-sdnrdb-init-job-l9vzk              0/1     Completed           0          7m14s
onap          dev-sdnrdb-coordinating-only-9b9956fc-dc5ts 2/2     Running             0          7m16s
onap          dev-sdnrdb-master-0                         1/1     Running             0          7m16s
onap          dev-sdnrdb-master-1                         1/1     Running             0          6m16s
onap          dev-sdnrdb-master-2                         1/1     Running             0          6m
ricaux        bronze-infra-kong-68657d8dfd-l4lf8          0/2     Running             1          7s
ricaux        deployment-ricaux-ves-65db844758-vnpwg      0/1     ContainerCreating   0          0s
```

Figura 35: Estado de los contenedores de la instalación de SMO.

Capítulo 5. Instalación de near-RT RIC

En este capítulo se va a proceder a la implementación del near-RT RIC también conocido como RIC siguiendo el procedimiento que la comunidad software de O-RAN ha establecido para esta entidad. Teniendo claro los requisitos mínimos para su despliegue anteriormente detallados se puede iniciar su instalación.

Esta entidad tiene un objetivo claro ser la plataforma de alojamiento de xApps que servirá para llevar a cabo el caso de uso que tiene que ver con la dirección de tráfico *Traffic Steering*. En esta versión se ha querido mejorar el algoritmo con respecto a la primera versión en términos de latencia y rendimiento además de comportarse de manera más sólida para almacenar su estado en situaciones de fallo, reinicio o actualización.

5.1 Instalación de RIC

Para completar esta instalación no es necesario empezar desde el inicio ya que se puede reutilizar parte de los repositorios usados para la instalación de SMO. Por ello, el punto de partida se establece en la generación del script que será necesario para la implementación completa de RIC.

```
$ cd tools/k8s/bin
$ ./gen-cloud-init.sh
$ ./k8s-1node-cloud-init-k_1_16-h_2_12-d_cur.sh
```

En esta ocasión es necesario lanzar el comando *.sh* dos veces para que la instalación sea la correcta. Para corroborar esto, se vuelve a ejecutar el comando siguiente y se obtiene los contenedores que forma parte de *k8s* pero para near-RT RIC.

```
$ kubectl get pods --all-namespaces
```

A continuación, se procede a implementar de forma completa el RIC, para ello es necesario a acceder a la carpeta *bin* del repositorio descargado anteriormente y proceder a su ejecución.

```
$ cd dep/bin
$ ./deploy-ric-platform – f ../RECIPE_EXAMPLE/PLATFORM/example_recipe.yaml
```

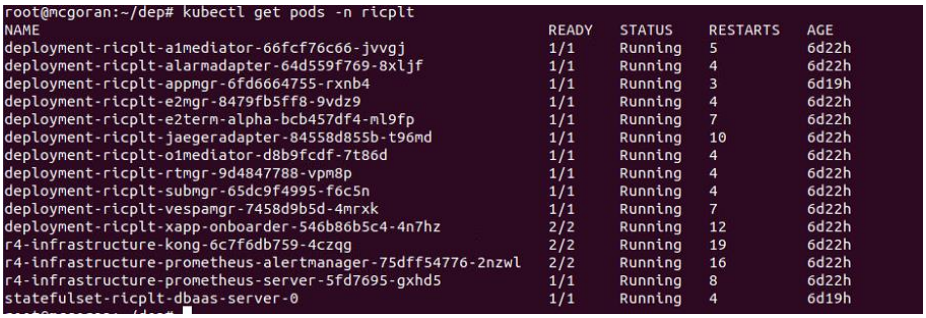
```
Run: helm ls --all r4-jaegeradapter; to check the status of the release
Or run: helm del --purge r4-jaegeradapter; to delete it
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "local" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈Happy Helming!⎈
```

Figura 36: Implementación de la plataforma RIC

Una vez ejecutado el RIC, es conveniente constatar que el proceso se ha desarrollado de manera correcta y para proceder a su comprobación es necesario correr el comando que aparece abajo, para de esta manera saber si los dieciséis contenedores que deben estar en estado running lo están y por lo tanto la implementación del RIC también.

```
$  kubectl get pods -n ricplt
```

Se comprueba que esto es así y que todos los contenedores poseen el estado adecuado. Estos contenedores serán fundamentales para la ejecución de ambos casos de uso ya que entre ellos van a tener que comunicarse.



```
root@mcgoran:~/dep# kubectl get pods -n ricplt
NAME                                                    READY   STATUS    RESTARTS   AGE
deployment-ricplt-a1mediator-66fcf76c66-jvvgj           1/1     Running   5          6d22h
deployment-ricplt-alarmadapter-64d559f769-8xljf         1/1     Running   4          6d22h
deployment-ricplt-appmgr-6fd6664755-rxnb4               1/1     Running   3          6d19h
deployment-ricplt-e2mgr-8479fb5ff8-9vdz9                1/1     Running   4          6d22h
deployment-ricplt-e2term-alpha-bcb457df4-ml9fp          1/1     Running   7          6d22h
deployment-ricplt-jaegeradapter-84558d855b-t96md        1/1     Running   10         6d22h
deployment-ricplt-o1mediator-d8b9fcdf-7t86d             1/1     Running   4          6d22h
deployment-ricplt-rtmgr-9d4847788-vpm8p                 1/1     Running   4          6d22h
deployment-ricplt-submgr-65dc9f4995-f6c5n               1/1     Running   4          6d22h
deployment-ricplt-vespamgr-7458d9b5d-4mrxk              1/1     Running   7          6d22h
deployment-ricplt-xapp-onboarder-546b86b5c4-4n7hz       2/2     Running   12         6d22h
r4-infrastructure-kong-6c7f6db759-4czqg                 2/2     Running   19         6d22h
r4-infrastructure-prometheus-alertmanager-75dff54776-2nzwl   2/2     Running   16         6d22h
r4-infrastructure-prometheus-server-5fd7695-gxhd5       1/1     Running   8          6d22h
statefulset-ricplt-dbaas-server-0                       1/1     Running   4          6d19h
```

Figura 37: Estado de los contenedores de RIC.

Capítulo 6. Configuración de la interfaz A1 y banco de resultados

En este momento del desarrollo, se presenta el primero de los dos casos de uso que lleva integrada está nueva versión de código abierto. Para su realización es fundamental que ambas entidades estén implementadas en máquinas virtuales diferentes ya que de ello dependerá poder comprobar la salud que tienen estos componentes a través del caso uso *Health Check*. Este va a ser el objetivo de esta sección.

6.1 Configuración de la interfaz A1

Después de completar la implementación de ambas entidades en máquinas virtuales diferentes, se puede establecer la conexión entre ellas a través de la interfaz A1. Antes de comenzar con el desarrollo, es necesario comprobar dos cosas. Por una parte, se debe encontrar la IP propia de cada una de las máquinas virtuales en las que se encuentran alojados la implementación de SMO y near-RT RIC. Y en según lugar, se debe constatar que el estado de cada uno de los contenedores que forman parte del despliegue de ambas entidades está corriendo.

Para ello, se necesita ejecutar el comando *ifconfig* en cada una de las terminales de Ubuntu. De ahí se puede extraer cual es la IP que se va a usar en la configuración. Con respecto al estado, como se ha visto anteriormente se confirma que el estado es el correcto por lo que ya se puede iniciar la conexión entre ambas entidades.

IP SMO	192.168.122.85
IP RIC	192.168.122.197

Tabla 3. IPs de las entidades SMO y RIC.

El primer paso entonces es cambiar la IP del archivo de configuración *example_recipe.yaml*, que se encuentre alojado en la máquina virtual de SMO. Para su configuración es importante corroborar en la máquina virtual del RIC cuál es el puerto IP en el cual se encuentra la infraestructura *Kong-proxy*.

```
RIC$ kubectl get service r4-infrastructure-kong-proxy -n ricplt -o yaml
```

De tal forma, que ejecutando el anterior comando se obtiene que el puerto deseado es el 32080 y que la IP externa de RIC que se va a utilizar es 192.168.122.197. En la terminal aparece otra IP interna pero no es la requerida en el sistema de desarrollo.

```
metadata:
  creationTimestamp: "2020-11-19T10:22:12Z"
  labels:
    app.kubernetes.io/instance: r4-infrastructure
    app.kubernetes.io/managed-by: Tiller
    app.kubernetes.io/name: kong
    app.kubernetes.io/version: "1.4"
    helm.sh/chart: kong-0.36.6
  name: r4-infrastructure-kong-proxy
  namespace: ricplt
  resourceVersion: "880"
  selfLink: /api/v1/namespaces/ricplt/services/r4-infrastructure-kong-proxy
  uid: a6ee1737-4326-4b25-b89c-2addfbc41ccc
spec:
  clusterIP: 10.111.146.238
  externalTrafficPolicy: Cluster
  ports:
  - name: kong-proxy
    nodePort: 32080
    port: 32080
    protocol: TCP
    targetPort: 32080
  - name: kong-proxy-tls
    nodePort: 32443
    port: 32443
    protocol: TCP
    targetPort: 32443
  selector:
    app.kubernetes.io/component: app
    app.kubernetes.io/instance: r4-infrastructure
    app.kubernetes.io/name: kong
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Figura 38. Puerto de Kong-Proxy desde RIC

Continuando con el primer paso, se accede al archivo de configuración a través de la siguiente ruta y establecen los cambios en la línea *de baseUrl*, adaptándola a la IP y al puerto donde se encuentra el RIC. Cuando se han hecho las modificaciones oportunas el archivo. yaml queda como se muestra a continuación.

```
SMO$ cd dep/smo/bin/smo-deploy/smo-dep/RECIPE_EXAMPLE/NONRTRIC
SMO$ nano example_recipe.yaml
```

```
ric: |
  [
    {
        "name":"ric1",
        "baseUrl":"http://192.168.122.197:32080/a1mediator",
        "controller": "controller1",
        "managedElementIds":[
           "kista_1",
           "kista_2"]
    }
```

Figura 39. URL cambiada a la IP de RIC.

Configurado el archivo *example_recipe.yaml*, es imprescindible confirmar que no va a dar problemas a la hora de su ejecución. Por este motivo, es conveniente volver a implementar el SMO para poder así reafirmarlo.

```
SMO$ cd dep/smo/bin/smo-deploy/smo-dep/bin

SMO$ ./undeploy-nonrtric && sudo ./deploy-nonrtric -f
../RECIPE_EXAMPLE/NONRTRIC/example_recipe.yaml

SMO$ sudo kubectl get service -n nonrtric
```

Seguidamente se corre el comando para encontrar las IP internas de cada uno de los contenedores de SMO que son necesarios para revisar si el RIC está disponible y por tanto que la conexión existe. Para llevarlo a cabo en primer lugar se debe conocer el puerto y la IP del contenedor llamado *policymanagementservice* y en segundo lugar comprobarlo.

```
root@mcgoran1:~/dep/smo/bin/smo-deploy/smo-dep/bin# sudo kubectl get service -n nonrtric
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
a1-sim                  ClusterIP   None             <none>        8085/TCP,8185/TCP
a1controller            ClusterIP   10.103.241.234   <none>        8282/TCP,8383/TCP
controlpanel            NodePort    10.104.54.254    <none>        8080:30091/TCP,8081:30092/TCP
dbhost                  ClusterIP   10.99.36.212     <none>        3306/TCP
policymanagementservice NodePort    10.111.81.167    <none>        9080:30093/TCP,9081:30094/TCP
sdnctldb01              ClusterIP   10.109.65.11     <none>        3306/TCP
```

Figura 40. Puerto del contenedor *policymanagementservice.*

Por lo que se puede ver como el puerto es el 9080 y la IP interna es 10.111.81.167, a continuación, se verifica en el terminal que existe la conexión entre el SMO y el RIC.

```
SMO$ curl -X GET http://10.111.81.167:9080/rics
```

```
root@mcgoran1:~/dep# curl -X GET http://10.111.81.167:9080/rics
[{"ricName":"ric1","managedElementIds":["kista_1","kista_2"],"policyTypes":["20008"],"state":"AVAILABLE"
```

Figura 41. Estado del *ric1* al establecer la conexión entre RIC y SMO

Antes de continuar es reseñable describir de qué forma se definió el tipo de política (*policy-type*) ya que va a ser de suma importancia para que la conexión entre ambas sea definitiva y correcta.

Para ello, y desde la máquina virtual de RIC, el primer paso es crear un nuevo tipo de política con el identificador “20008”. Dicha política va a ser creada en el contenedor llamado *A1Mediator* por lo que el siguiente paso es consultar mediante una petición *GET* de HTTP si ha recibido la petición y ha sido aceptada.

```
RIC$ POLICY_TYPE_ID="20008"

RIC$ echo '{ "name": "tsapolicy", "description": "tsa parameters",
"policy_type_id": 20008, "create_schema": { "$schema": "http://json-
schema.org/draft-07/schema#", "title": "TS Policy", "description": "TS
policy type", "type": "object", "properties": { "threshold": { "type":
"integer", "default": 0 } }, "additionalProperties": false } } ' > ts-
policy-type-20008.json

RIC$ curl -X GET --header "Content-Type: application/json" --header
"accept: application/json"   http://192.168.122.197:32080/a1mediator/a1-
p/policytypes
```

```
*   Trying 192.168.122.197...
* TCP_NODELAY set
* Connected to 192.168.122.197 (192.168.122.197) port 32080 (#0)
> PUT /a1mediator/a1-p/policytypes/20008 HTTP/1.1
> Host: 192.168.122.197:32080
> User-Agent: curl/7.58.0
> accept: application/json
> Content-Type: application/json
> Content-Length: 329
>
* upload completely sent off: 329 out of 329 bytes
< HTTP/1.1 201 CREATED
< Content-Type: application/json
< Content-Length: 3
< Connection: keep-alive
< Date: Mon, 30 Nov 2020 10:51:47 GMT
< X-Kong-Upstream-Latency: 6
< X-Kong-Proxy-Latency: 0
< Via: kong/1.4.3
```

Figura 42. Creación del tipo de política.

Por último, se pretende generar una instancia de política que proporcione a sus propiedades valores adecuados. En esta ocasión se ha elegido alguna de las propiedades como que el umbral de peticiones sea cinco.

```
RIC$ POLICY_ID="tsapolicy145"

RIC$ curl -X PUT --header "Content-Type: application/json"   --data
"{\"threshold\" : 5}"   http://$(hostname):32080/a1mediator/a1-
p/policytypes/${POLICY_TYPE_ID}/policies/${POLICY_ID}
```

Con este paso se completa el desarrollo para la conexión de las entidades SMO y RIC mediante la interfaz A1. A continuación tendrá lugar la última fase del despliegue donde se obtendrán los resultados. Con ellos, se comprenderá que el proyecto se ha concluido con éxito y que la comunicación entre ambos terminales existe.

6.2 Resultados

La generación del banco de pruebas que O-RAN proporciona para este caso de uso es una interfaz de usuario web, una forma más visual para constatar los resultados del despliegue y la conexión. Para que la web funcione es necesario correr un comando que permitirá establecer la conexión con el puerto 8088. Conectados a la url http://localhost:8088 , se puede observar el entorno de la web.

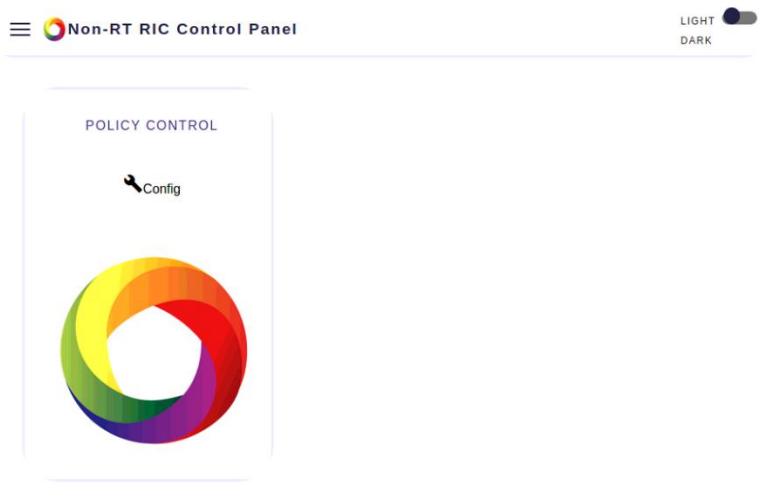


Figura 43. Entorno de la web.

Para su puesta en marcha se necesita ejecutar el siguiente comando port forward que cuya funcionalidad esencial es enviar paquetes de un puerto a otro, en este caso envia paquetes del puerto 8080 al puerto 8088 y establecer la conexión necesaria.

```
SMO$ kubectl port-forward controlpanel-8bd4748f4-vmw72 8088:8080 -n
nonrtric
```

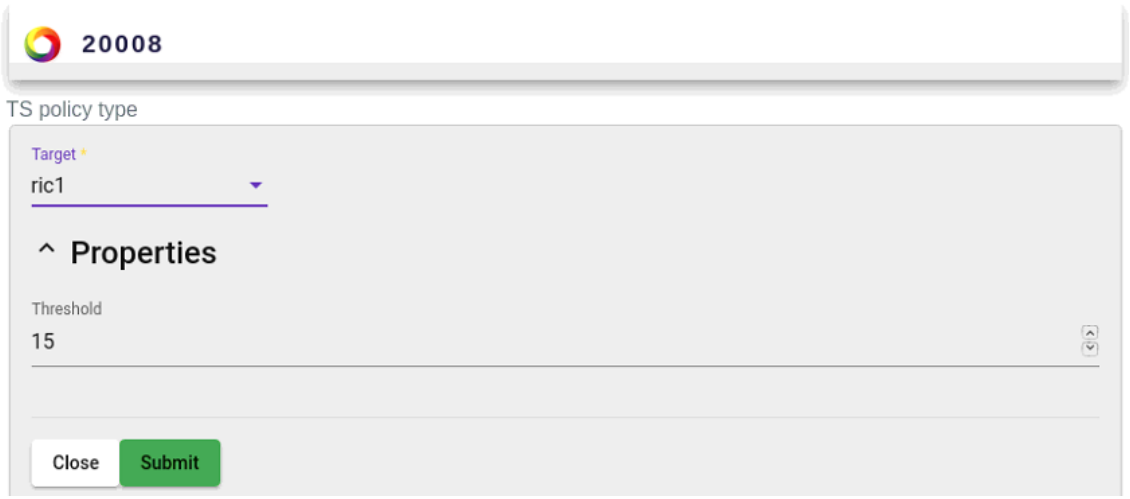
Figura 44. Conexión a la interfaz de usario web a través del puerto 8088.

Antes de seguir avanzando, es importante aclarar a que se refiere el puerto 8080. Pues bien este puerto pertenece al contenedor llamado *controlpanel* y en la siguiente captura del terminal se puede ver como pertenece a este contenedor.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-11-29T18:02:48Z"
  labels:
    app: nonrtric-controlpanel
    chart: controlpanel-2.0.0
    heritage: Tiller
    release: r2-dev-nonrtric
  name: controlpanel
  namespace: nonrtric
  resourceVersion: "8153"
  selfLink: /api/v1/namespaces/nonrtric/services/controlpanel
  uid: 1e6f7b11-e0bc-4fd2-91d4-4902f61241a2
spec:
  clusterIP: 10.104.54.254
  externalTrafficPolicy: Cluster
  ports:
  - name: http
    nodePort: 30091
    port: 8080
    protocol: TCP
    targetPort: 8080
  - name: https
    nodePort: 30092
    port: 8081
    protocol: TCP
    targetPort: 8082
  selector:
    app: nonrtric-controlpanel
    release: r2-dev-nonrtric
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Figura 45. Puerto del contenedor *controlpanel*.

Finalmente, se accede a la interfaz web, en la que se debe configurar el tipo de ric al que pertenece en este caso *ric1* y su instancia. Como se puede comprobar aparece también el tipo de política que se ha creado explicada anteriormente y cuyo identificador es “20008”.



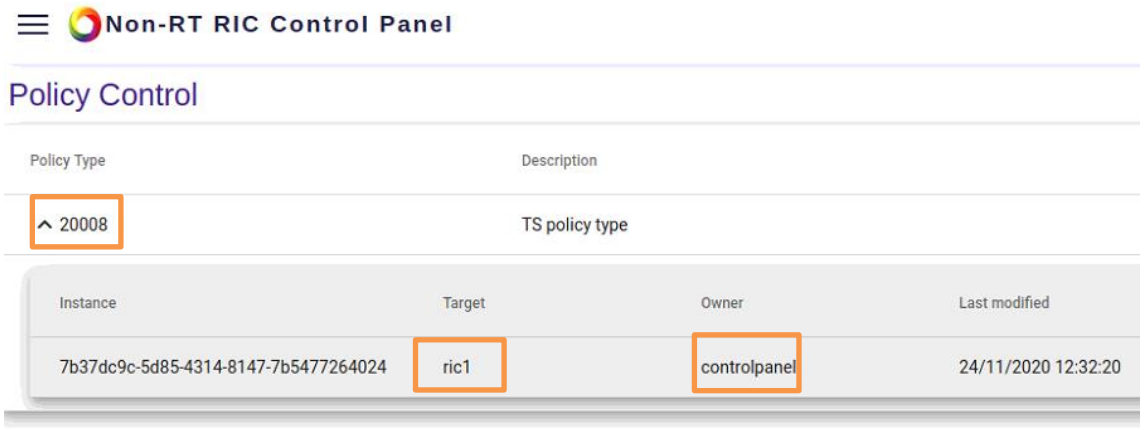


Figura 46. Entorno de la web.

Para constatar que se ha instalado instancia de política desde la máquina virtual de RIC se puede ver a través del registro que mensajes que el RIC hace a SMO.

```
RIC$ kubectl logs -f deployment-ricplt-a1mediator-66fcf76c66-jvvgj  -n
ricplt
```

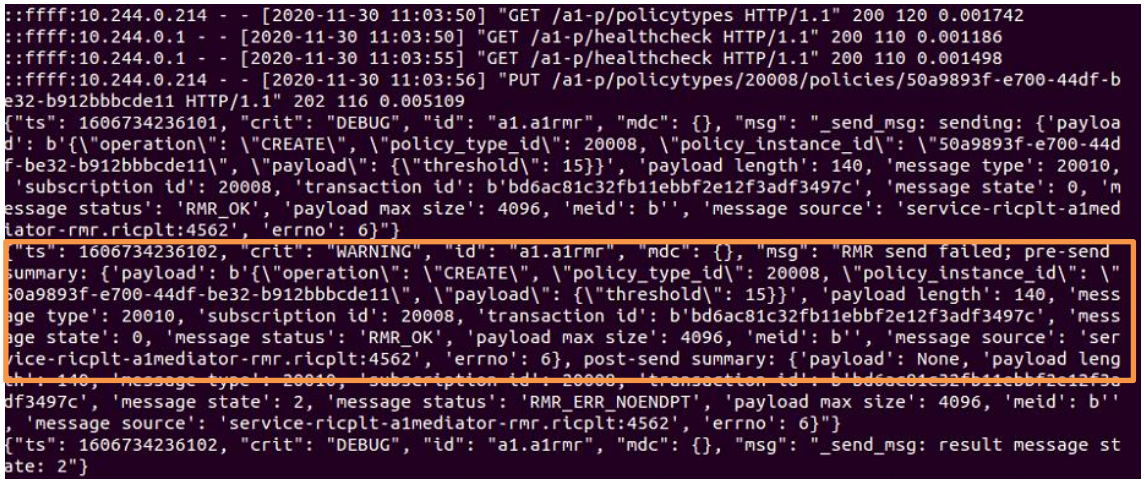


Figura 47. Registro de mensajes entre RIC y SMO

Teniendo en cuenta que ya se ha establecido la comunicación entre ambas entidades, en la interfaz web de usuario se puede gestionar la instancia y crear un servicio llamado *ExampleSvc* sobre la que ya aparece. De modo que, en primer lugar, se crea el servicio y después se genera la nueva instancia que se va a llamar *policyTh15*.

```
SMO$  curl -X PUT "http://10.111.81.167:9080/service" -H "Content-Type:
application/jso -d '{ "callbackUrl": "",  "keepAliveIntervalSeconds": 0,
"serviceName": "exampleSvc" }'

SMO$ curl -X GET --header "Content-Type: application/json"
"http://10.111.81.167:9080/policies"
```

En la terminal de SMO se puede ver la nueva instancia, el tipo de política, el servicio y el RIC sobre el que se está ejecutando.

```
root@mcgoran1:~/dep# curl -X GET --header "Content-Type: application/json" "http://10.111.81.167:9080/po
licies"
[{"id":"policyTh15","type":"20008","ric":"ric1","json":{"threshold":15.0},"service":"exampleSvc","lastMo
dified":"2020-11-30T11:31:16.608313Z"},{"id":"50a9893f-e700-44df-be32-b912bbbcde11","type":"20008","ric"
:"ric1","json":{"threshold":15.0},"service":"controlpanel","lastModified":"2020-11-30T11:03:55.703085Z"}
```

Figura 48. Creación de un nuevo servicio e instancia entre SMO y RIC.

Conjuntamente a estos cambios, se puede revisar que estas modificaciones aparecen en la interfaz de usuario web, de igual manera que desde el registro de mensajes del RIC donde también se muestran estos cambios.

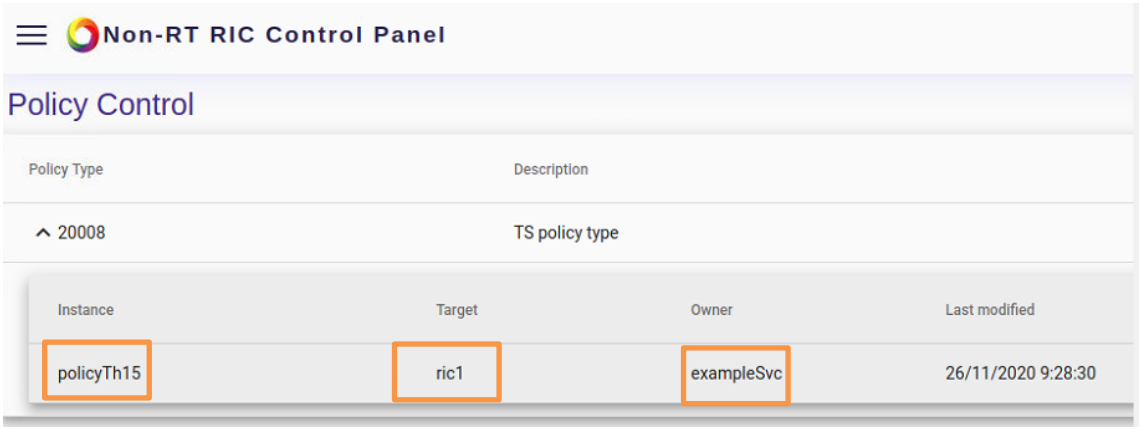


Figura 49. Entorno web después de las modificaciones.

Figura 50. Registro de mensajes entre RIC y SMO después de los cambios.

Capítulo 7. Configuración de xApps y resultados

7.1 Introducción

Con la configuración de las xApps se completan los dos casos de uso que están disponibles hasta este momento. En el caso, del direccionamiento del tráfico (*Traffic Steering*) se establecerá la conexión dentro del propio RIC donde están alojadas estas aplicaciones para controlar el todo momento el estado en que se encuentra la red móvil. Como esto todavía no está disponible se generan mensajes distintos y de forma premeditada ya que no está unido al gNB.

Por lo que, el flujo de mensajes que se crea ahora es muy intuitivo. Teniendo claro cómo se crea un tipo de política y su clase se puede establecer el primer mensaje entre el A1mediator y la primera xApp en el que se describe que tipo de política es en este caso es la "20008" estos datos harán que se pasan por las diferentes xApps existentes, las cuales requieren la predicción y devuelven información del usuario.

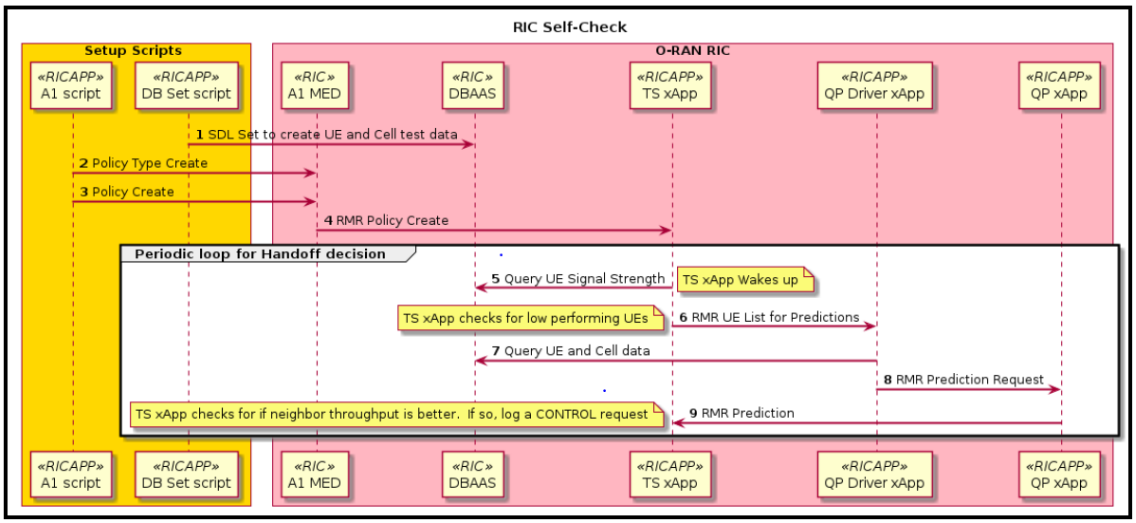


Figura 51. Flujo de mensajes en el RIC

En esta ocasión tanto las predicciones como los datos del usuario son predeterminados por el caso de uso por lo que los resultados que aparecerán más adelantes son solamente un ejemplo de la funcionalidad de estas aplicaciones para este caso de uso. A lo largo del desarrollo, se irán planteado las diferentes xApps y su implementación para tener un análisis completo del caso de uso.

7.2 Configuración de xApps

Existen cuatro posibles xApps que se pueden implementar dentro del RIC en estos momentos. Sus nombres son, *Hello Word xApp, QP (QoP Predictor) xApp, QP (driver xApp) y Traffic Steering xApp*, todas ellas dispuestas dentro del RIC y son utilizadas, por tanto, para este caso de uso.

En primer lugar, con el comando *echo*, se ejecutará cada una de las xApps mediante el cual se imprimirá por pantalla un mensaje con la función que se esté realizando. Para este paso, la función *pulling* indica que está tirando todas las imágenes de Docker de forma manual de cada una de las xApps.

```
root@mcgoran:~/dep# ^C
root@mcgoran:~/dep# echo && echo Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-hw:1.0.6

Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-hw:1.0.6
root@mcgoran:~/dep# echo && echo Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-qp:0.0.2

Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-qp:0.0.2
root@mcgoran:~/dep# echo && echo Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-qp-driver:1.0.9

Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-qp-driver:1.0.9
root@mcgoran:~/dep#
root@mcgoran:~/dep# echo && echo Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-ts:1.0.11

Pulling nexus3.o-ran-sc.org:10002/o-ran-sc/ric-app-ts:1.0.11
```

Figura 52. Captura de la función Pulling de las xApps

De forma independiente, el puerto 32080 está siendo bloqueado por *Kong-Proxy* ya que está siendo usado para la interfaz A1 en el contenedor de *r4-infrastructure-kong*, por lo que hay que solucionar este problema que impide que este puerto sea utilizado para otra función. La solución óptima, fue usar la función *port-forwarding* que permite enviar información de un puerto a otro y de esta forma no había que modificar el puerto ya utilizado. Simplemente con el comando *curl* se cambió el *localhost* indicando que ahora el nuevo puerto era el "32088".

```
RIC$ curl --location --request POST
"http://localhost:32088/onboard/api/v1/onboard/download" --header
'Content-Type: application/json' --data-binary "@./onboard.hw.url"
```

```
^Croot@mcgoran:~/dep# sudo kubectl port-forward r4-infrastructure-kong-6c7f6db75
4czqg 32088:32080 -n ricplt
Forwarding from 127.0.0.1:32088 -> 32080
Forwarding from [::1]:32088 -> 32080
Handling connection for 32088
Handling connection for 32088
Handling connection for 32088
Handling connection for 32088
Handling connection for 32088
Handling connection for 32088
```

Figura 53. Envío de datos del puerto 32080 al puerto 32088.

Una vez que el puerto ha sido cambiado y solucionado el problema, se abre un terminal nuevo, ya que el anterior está realizando el envío entre puertos y se prepara las descripciones de las xApps mediante el comando *echo*. El resultado permite que se puedan crear las cuatro xApps de manera correcta.

El paso siguiente consiste en enviar una petición *post* que hace que se genere el estado creado de cada una de las aplicaciones. Como se puede ver en la figura posterior conexión establece a través del puerto "32088" como se había especificado anteriormente y que se produzca que las xApps se pongan en funciones *onboard.*

```
root@mcgoran:~/dep# sudo curl --location --request POST "http://localhost:32088/onboard/api/v1/onboard/d
ownload"  --header 'Content-Type: application/json' --data-binary "@./onboard.hw.url"
{
    "status": "Created"
}
root@mcgoran:~/dep# {
> ^C
root@mcgoran:~/dep# sudo curl --location --request POST "http://localhost:32088/onboard/api/v1/onboard/d
ownload"  --header 'Content-Type: application/json' --data-binary "@./onboard.qp.url"
{
    "status": "Created"
}
root@mcgoran:~/dep# sudo curl --location --request POST "http://localhost:32088/onboard/api/v1/onboard/d
ownload"  --header 'Content-Type: application/json' --data-binary "@./onboard.qpd.url"
{
    "status": "Created"
}
root@mcgoran:~/dep# sudo curl --location --request POST "http://localhost:32088/onboard/api/v1/onboard/d
ownload"  --header 'Content-Type: application/json' --data-binary "@./onboard.ts.url"
{
    "status": "Created"
}
```

Figura 54: Estado de las aplicaciones previo a su desarrollo.

Para continuar con el proceso de su puesta en marcha es conveniente verificar que el cuadro de dirección de las xApps existe y se han construido de forma adecuada. Para ello se ejecuta el siguiente comando una petición *get* que informará de su situación.

```
RIC$ curl --location --request GET
"http://localhost:32088/onboard/api/v1/charts"
```

Figura 55: Estado de xApps

Para finalizar el desarrollo se ejecuta una petición *post*, la cual implementa de forma definitiva las xApps. De forma análoga a la siguiente figura el terminal recibía el siguiente mensaje con el despliegue de cada una de las aplicaciones. Destaca el mensaje *HTTP 201 Created* además de la versión, y estado de la aplicación.

```
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 32088 (#0)
> POST /appmgr/ric/v1/xapps HTTP/1.1
> Host: localhost:32088
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 18
>
* upload completely sent off: 18 out of 18 bytes
< HTTP/1.1 201 Created
< Content-Type: application/json
< Content-Length: 67
< Connection: keep-alive
< Date: Fri, 27 Nov 2020 10:51:18 GMT
< X-Kong-Upstream-Latency: 3702
< X-Kong-Proxy-Latency: 1
< Via: kong/1.4.3
<
{"instances":null,"name":"qp","status":"deployed","version":"1.0"}
```

Figura 56. Despliegue de las xApps recibido en el servidor

Para verificar que el procedimiento ha sido el correcto y que las aplicaciones han sido desplegadas correctamente, se ejecuta el siguiente comando y se revisa que todas ellas están en estado *running*.

```
NAME                                  READY   STATUS    RESTARTS   AGE
ricxapp-hwxapp-684d8d675b-qm2x2       1/1     Running   5          7d21h
ricxapp-qp-5f6fc7b746-mkbr8           1/1     Running   0          7m44s
ricxapp-qpdriver-6b89bb66c-r78d2      1/1     Running   0          5m21s
ricxapp-trafficxapp-96bc7bd6b-ljkrf   1/1     Running   0          4m4s
```

Figura 57. Estado de xApp

7.3 Resultados

Como ya se han implementado las xApps es momento de materializar el caso de uso de *Traffic Steering*. Para ello es necesario clonar un repositorio de *Git Hub* dentro del repositorio *dep* que se creó para instalar las entidades de RIC y SMO y que da acceso a los diferentes archivos.

```
RIC$ cd home/dep
RIC$ git clone http://gerrit.o-ran-sc.org/r/ric-app/ts -b bronze
```

Una vez clonado el repositorio, se debe acceder a la carpeta llamada *populatedb* donde se encuentra alojado el archivo *Dockerfile* y que hay que modificar y actualizar a una versión nueva para que se construya de manera adecuada.

```
RIC$ cd home/dep/ts/test/populatedb
RIC$ nano Dockerfile
```

```
FROM nexus3.o-ran-sc.org:10004/o-ran-sc/bldr-ubuntu18-c-go:9-u18.04 as buildenv

RUN mkdir /playpen
RUN mkdir /playpen/src

ENV LD_LIBRARY_PATH=/usr/local/lib

RUN apt-get install -y cpputest
RUN apt-get remove -y libboost-all-dev
RUN apt-get install -y  libboost-all-dev
RUN apt-get install -y libhiredis-dev
RUN apt-get install -y valgrind
```

Figura 58. Versión de Docker actualizada.

Cambiado ese parámetro, ya se puede correr la imagen de *Docker* que tiene la versión que se ha añadido anteriormente. Después se constata que la imagen de *Docker* ya está establecida para el Ubuntu.

```
RIC$ docker pull nexus3.o-ran-sc.org:10004/o-ran-sc/bldr-ubuntu18-c-go:9-u18.04
RIC$ docker images | grep ubuntu18
```

El siguiente paso, va a permitir la comprobación del caso de uso que toca. Por lo que en primer lugar se tiene que ejecutar el siguiente comando y después revisar que su puesta en marcha se ha completado.

```
RIC$ ./populate_db.sh
RIC$ sudo kubectl get jobs -A
```

```
root@mcgoran:~/dep/ts/test/populatedb# sudo kubectl get jobs -A
NAMESPACE   NAME                       COMPLETIONS   DURATION   AGE
ricinfra    tiller-secret-generator    1/1           7s         8d
ricplt      dbprepopjob                1/1           1s         71s
```

Figura 59: Comprobación de la correcta inicialización

Por último, para su visualización del comportamiento de las xApps dentro del caso de uso, es determinante ejecutar un comando para que se proporcione la información relativa a las métricas de RAN, y solicite las predicciones para los usuarios y se muestre el rendimiento de los usuarios también.

En el caso de uso de *Traffic Steering*, el funcionamiento consiste en enviar un mensaje de predicción de calidad de experiencia (QoE) cada vez que se identifica a un usuario siendo este mensaje de tipo “30000”. Cuyo cuerpo de mensaje es el que aparece en la Figura.

Conjuntamente, se genera una respuesta que es recibida en la QP xApp como se ha podido ver en la Figura 51, que devuelve un mensaje de tipo “30002” y cuyo cuerpo es el que se puede ver en la siguiente Figura.

```
In get_sdl_ue_data()
message body {"UEPredictionSet": ["12345"]}
payload length 30
In get_sdl_ue_data()
Prediction Callback got a message, type=30002 , length=182
payload is { "12345" : { "310-680-200-555001" : [ 2000000 , 1200000 ],              "310-680-200-555002
" : [  800000 , 400000  ],              "310-680-200-555003" : [  800000 , 400000  ] } }
Prediction for 12345
```

Figura 60. Tipo y cuerpo del mensaje recibido por la QP xApp.

Este mensaje, proporciona predicciones para el usuario *UE ID 12345* y comprueba si el rendimiento es mayor que el de una celda vecina. Si esto es así, TF xApps registra su intención de realizar una petición de control para realizar el traspaso.

```
Prediction for 12345

WE WOULD SEND A CONTROL REQUEST NOW
UE ID: 12345
Source cell 310-680-200-555002
Target cell 310-680-200-555001
```

Figura 61. Petición de control para realizar el traspaso

Capítulo 8. Interfaz E2 entre RIC y o-du high

En el último capítulo de desarrollo, se pondrá en marcha la interfaz más crítica para la arquitectura O-RAN ya que es la que representa la unión entre las capas más bajas de esta arquitectura con las capas de gestión y modelado de la misma. Su inicialización y posterior comprobación de mensajes entre RIC y el o-du high servirá para asentar las bases de las posibilidades que esta interfaz tiene en la arquitectura O-RAN.

8.1 Instalación de o-du high

Previo a la instalación de esta entidad, es necesario descargarse la última versión de la *Release Bronze* donde está alojado la demo de o-du high y que es la que se construirá en esta sección. Por ello, se procede a descargarse el repositorio comprimido *o-ran-sc-bronze-20200810.tar.gz* y después se descomprime para tener acceso a todo su contenido.

Posteriormente a esto, se accede por la terminal mediante el siguiente comando que permite entrar en el archivo de configuración donde hay que realizar algunos cambios.

```
ODU$ cd o-ran-sc-bronze-20200810/it.dep/demos/bronze
ODU$ nano odu-high.sh
```

Estas modificaciones están encaminadas a establecer las direcciones IP tanto del RIC como de la máquina virtual en la que se va a instalar el o-du. Como en otra ocasión anterior, mediante el comando *ifconfig* se obtiene la IP que se busca de ambas entidades.

IP O-DU	192.168.122.22
IP RIC	192.168.122.197

Tabla 4. Direcciones IP de O-DU y RIC

De manera que, se añaden estas direcciones en el archivo de ejecución y se construye el o-du-high.

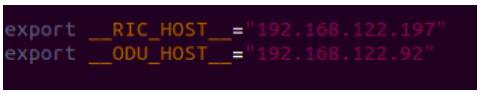


Figura 62. Direcciones IP añadidas para O-DU.

```
ODU$ sudo ./odu-high.sh
```

8.2 Configuración interfaz E2

Teniendo en cuenta que se ha completado la construcción de la demo de o-du high, ahora toca desplegar los contendores que tienen la información necesaria para poder implementar la comunicación por la interfaz E2.

El primer paso, entonces, consiste en encontrar el puerto y la dirección IP de los ficheros *e2term* y *e2mgr*. Así, desde la máquina virtual del RIC donde se encuentran alojados estos ficheros se obtendrá lo que se busca. Para ello, en primer lugar, es necesario hallar el nombre de estos ficheros e inicializarlos.

```
RIC$ kubectl get pods -n ricplt
```

Figura 63. Nombre de los ficheros e2mgr y e2term.

Sabiendo el nombre se pueden inicializar cada uno a través de los siguientes comandos.

```
RIC$ sudo kubectl logs -f deployment-ricplt-e2mgr-8479fb5ff8-9vdz9 -n
ricplt

RIC$ sudo kubectl logs -f deployment-ricplt-e2term-alpha-bcb457df4-ml9fp -
n ricplt
```

Después, ya se puede localizar tanto la dirección IP como el puerto.

```
RIC$ sudo kubectl get service -A | grep e2term

RIC$ sudo kubectl get service -A | grep e2mgr
```

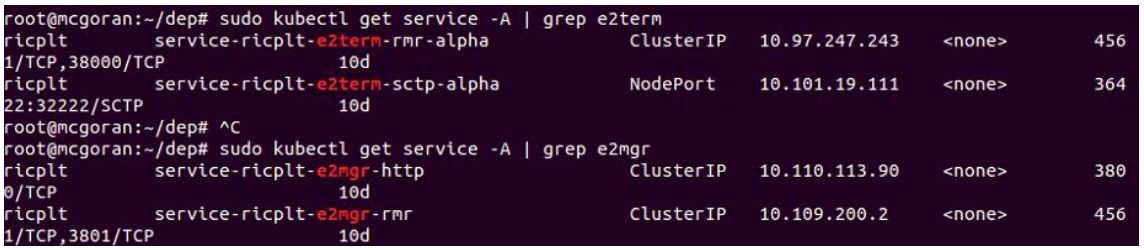


Figura 64. Dirección IP y puerto de e2mgr y e2term.

De esta forma, que el puerto y la dirección IP son 38000 y 10.97.247.243 de e2term respectivamente. En cuanto a *e2mgr* el puerto es 3800 y la dirección IP es la 10.110.113.90. Esto servirá para ver si existe conexión entre el o-du y el RIC y también para ver si existe algún gNB conectado.

Por lo que a continuación se procede a establecer la interfaz E2 a través de la inicialización del o-du desde una carpeta diferente.

```
ODU$ cd l2/bin/odu

ODU$ sudo ./odu
```

8.3 Resultados

Para finalizar el desarrollo, se debe verificar que esta interfaz está activa y que ambas entidades están comunicándose entre ellas. De tal forma que se introduce un gNB llamado *gnb:311-048-00010201* y junto con las IP y los puertos anteriormente detallados, se establece el flujo de mensajes.

Una vez que se está ejecutando el o-du, inicia un proceso de envío de paquetes para establecer su conexión y la del gNB. El o-du envía mensajes al RIC para avisarle que existe este gNB como se puede ver en el registro de la interfaz E2 dentro del RIC en la Figura 66. Estos paquetes son recibidos en el o-du como se puede ver en la Figura 66.

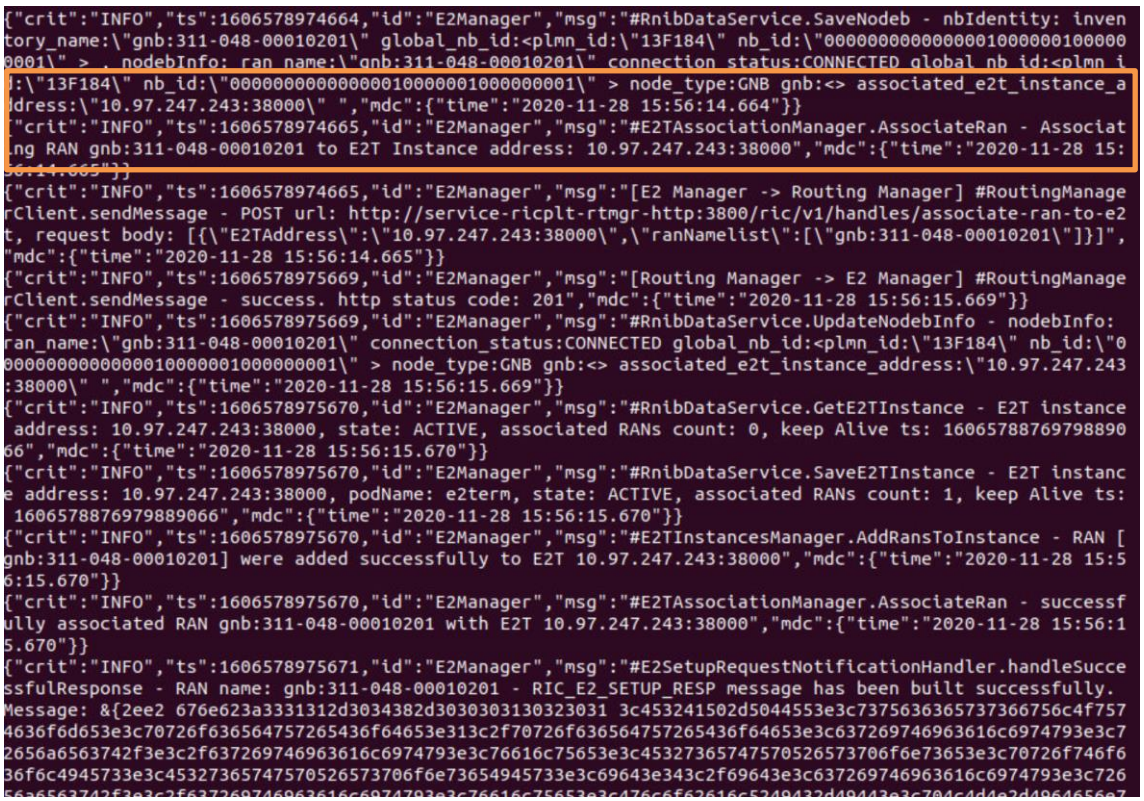


Figura 65.Registro de mensajes de RIC procedentes de o-du a través de la interfaz E2.

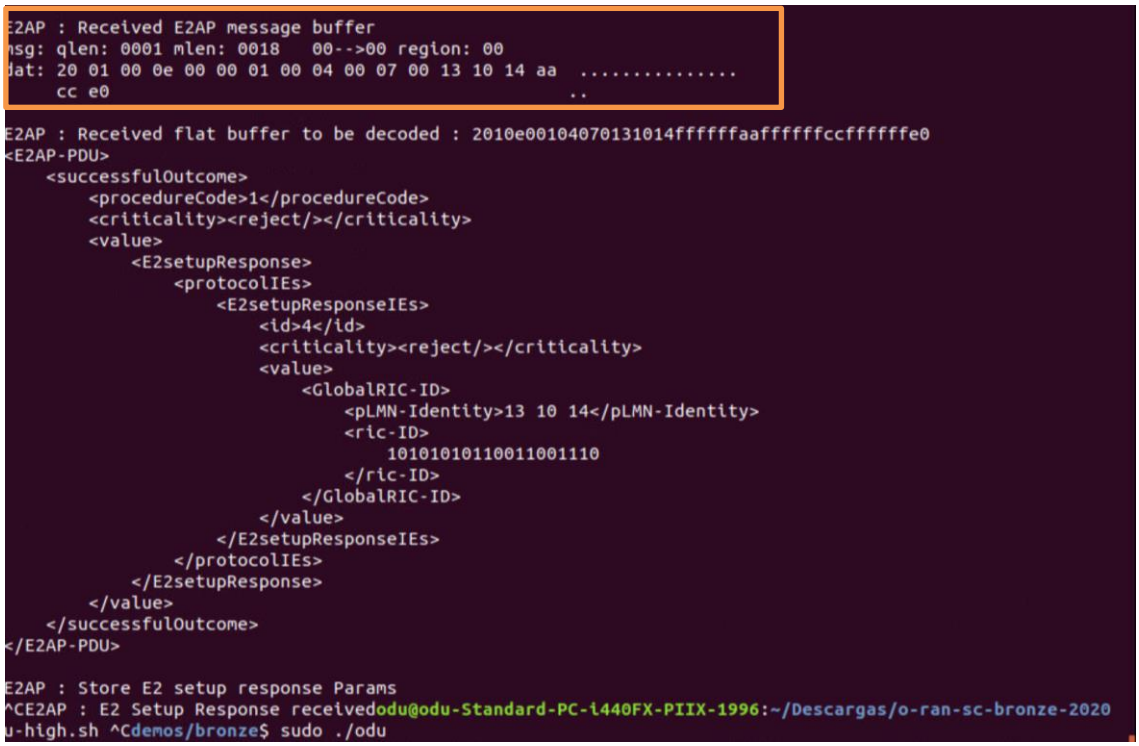


Figura 66. El flujo de mensajes en O-DU.

Para confirmar que se está produciendo la conexión y que es correcta, a través del comando *curl* y de las direcciones IP anteriores se obtienen todos estos datos. En la Figura 67 se muestra que el puerto 3800 con la dirección IP 10.110.113.90 ha establecido la conexión correctamente con el o-du y que el gNB está conectado también.

```
RIC$ curl http://10.110.113.90:3800/v1/nodeb/ids | jq .
RIC$ curl http://10.110.113.90:3800/v1/e2t/list | jq .
```

Figura 67. Comprobación de la comunicación entre RIC y el o-du.

Capítulo 9. Conclusiones y trabajo futuro

Este trabajo ha supuesto la iniciación de una línea de investigación acerca de la parte RAN de un sistema de comunicaciones. Se han asentado los primeros pasos para la construcción de redes abiertas y virtualizadas que harán que las redes celulares cambien de forma abrupta en los próximos años.

En las múltiples etapas de desarrollo de este proyecto se han necesitado diferentes conocimientos para llevar a cabo su realización. Para empezar, este proyecto requería la adaptación de los modelos tradicionales de la parte radio a una arquitectura completamente nueva que se presentaba totalmente virtualizada y dividida en distintas entidades que poco tenían que ver con la forma tradicional. Por ello, la investigación se centró en un fuerte análisis del estado del arte para comprender el paso de una tecnología a otra.

Esta nueva arquitectura se mostraba en una plataforma software de código abierto donde el entorno de trabajo se centraba en la utilización de máquinas virtuales de Ubuntu, contenedores y otras tecnologías que hicieron que se necesitara un fuerte aprendizaje continuado durante todo el proceso para su implementación. También, cabe resaltar los límites que la propia plataforma poseía, ya que, en muchas ocasiones el propio código estaba incompleto o contenía algún error y al ser una implementación tan poco madura y en continuo desarrollo, el aprendizaje era todavía mayor.

Finalmente, se consideró que el desarrollo de esta nueva arquitectura debía ir encaminado a la construcción de las capas más altas ya que eran las capas de gestión, administración de políticas y generación de recursos que más desarrolladas estaban en la plataforma de O-RAN. Con respecto a las capas más bajas, que representan a un gNB, su implementación en código abierto aún no estaba completa y por tanto no era posible su despliegue total.

El proceso, entonces, se ha centrado en la puesta en marcha de las entidades de SMO y RIC, correspondientes a las capas más altas, y a las distintas interfaces abiertas que han sido definidas para los casos de uso que la *Release Bronze* describe para la comunidad O-RAN. Junto a su implementación, también se estableció la interfaz A1 que comunica SMO con RIC y la interfaz E2 que comunica el RIC con las capas más bajas pertenecientes al gNB. En esta versión solo está desarrollada la conexión con una de las unidades que forma parte de un gNB, el o-du high, por lo que se realizó la prueba de concepto con esa entidad.

Durante el proyecto, se ha investigado en detalle el funcionamiento de la nueva arquitectura planteada por la Alianza O-RAN para posteriormente analizar su implementación en la *Release Bronze.* Teniendo en cuenta eso, se ha diseñado el despliegue de la plataforma en el hardware/software disponible. Para ello se ha utilizado un equipo de computación de altas prestaciones y un sistema operativo de software abierto. Sobre este, se han instalado las máquinas virtuales necesarias para ejecutar las instancias de la plataforma de código abierto. Como se ha expuesto anteriormente, las entidades implementadas han sido el RIC y el SMO, así como sus diferentes interfaces, por lo que se han utilizado dos máquinas virtuales.

En una máquina virtual, se ha desarrollado el SMO. La cual incluye las funciones en el servicio y gestión de políticas y análisis de las capas más bajas de la arquitectura. En la otra máquina virtual se instala el RIC que es una entidad lógica que permite el control casi en tiempo real y la optimización de un subconjunto de funciones de la gestión de recursos radio realizados por el gNB.

Además, se han configurado la interfaz la A1 que comunica ambas entidades. Con esta conexión, se ha establecido el caso de uso de *Health Check* a través de la interfaz de usuario web en la que se ha incorporado uno de los contenedores de kubernetes conocido como *controlpanel* mediante la creación de un tipo de política en el RIC.

Con respecto al otro caso de uso, que recibe el nombre de *Traffic Steering*, se ha podido realizar gracias al despliegue de las cuatro xApps que están definidas para esta *Release* dentro del módulo de RIC. Por lo que estas aplicaciones generan un flujo de mensajes encaminados a controlar el todo momento el estado en que se encuentra la red móvil. Este intercambio de paquetes se basa en sencillos mensajes caracterizados por una predicción.

Por otro lado, se ha podido instalar parte del o-du, concretamente el o-du high. Este bloque es el responsable de las capas de RLC y MAC del estándar del 3GPP. Y mediante su implementación se ha procedido al despliegue de la interfaz E2 que conecta el RIC con las capas inferiores, en este caso el o-du high y su comprobación a través del flujo de paquetes que genera cuando ambas entidades estén conectadas.

En conclusión, se ha verificado el correcto funcionamiento de las entidades RIC y SMO, así como el caso de *Health Check* a través de la interfaz A1. También se ha podido verificar la comunicación y el funcionamiento de o-du high y su conexión con el RIC gracias a la interfaz definida entre ellos la E2. Y finalmente, con el desarrollo de las xApps se ha podido llevar a cabo el otro caso de uso, el *Traffic Steering.*

De forma adicional a las conclusiones y resultados extraídos, cabe destacar, el gran potencial que esta arquitectura tiene para las siguientes versiones. Pero no solo eso, la Alianza O-RAN que es la organización que ha impulsado esta nueva forma de entender el RAN durante los últimos meses, tiene cada día más colaboradores ya que cuenta con las empresas de telecomunicaciones más importantes del sector, englobando tanto las operadoras como a los proveedores de sistemas de comunicación.

Ver como cada día esta organización tiene más apoyos que están poniendo todos sus recursos disponibles para que esta nueva arquitectura sea un hecho, es un gran estímulo para el futuro y para la importancia de dar hoy los primeros pasos con el despliegue de este proyecto. Además, la implicación que supone que exista una plataforma de código abierto que garantiza el funcionamiento inicial de la RAN es un gran paso para la reducción de costes y para la entrada de nuevos participantes en el mercado. Los pasos siguientes no van a ser sencillos, pero se espera un futuro muy prometedor para O-RAN.

El trabajo se ha completado siguiendo las especificaciones que la plataforma de O-RAN ofrecía según la *Release Bronze*, aunque a principios de noviembre se publicó una nueva versión que seguirá mejorando las capacidades que este sistema espera poseer. Por lo que el trabajo futuro de O-RAN y de estas entidades, es inmenso.

En este proyecto no se ha podido realizar un despliegue completo, ya que, no estaban definidos todos los módulos que forman parte de la arquitectura O-RAN en esta *Release*. En este caso, tanto las unidades CU y DU no estaban especificadas y por tanto no ha sido posible su desarrollo. Así, una línea de trabajo futuro es seguir investigando acerca de estos componentes y su implementación en esta arquitectura y especialmente dentro de los componentes que forman el gNB.

Las funcionalidades que tienen las entidades y las interfaces abiertas son muy limitadas, por lo que introducir las actualizaciones que la plataforma O-RAN haya hecho a estas entidades e interfaces, hará que mejoren los casos de uso.

En el entorno de laboratorio donde se ha podido llevar a cabo este proyecto, también existen posibles líneas de trabajo futuro a partir de lo desarrollado. De tal manera, que investigar la conectividad del RIC a través de la interfaz E2 con un gNB de los que dispone el laboratorio, es uno los siguientes pasos para verificar la interoperabilidad del ecosistema O-RAN.

A nivel industrial, la línea futura de trabajo más importante que esta nueva arquitectura y la comunidad O-RAN presenta, es poder desplegar una red comercial a gran escala mediante código abierto. Es cierto, que ya existan implementaciones de esta arquitectura, pero todavía pertenecen

a grandes operadoras por lo que el gran reto que tiene esta organización es conseguir una red de código abierto que sea completa y funcional.

Capítulo 10. Bibliografía

[1] Arturo Mrozowski Handzel, «Implementación del núcleo de red LTE/5G Virtualizado,» septiembre 2020.

[2] Mejía, Danilo Geovanny Barreno Naranjo y Darwin Paúl Carrión Buenaño y Iván Tenecora, «Evolución de la tecnología móvil. Camino a 5G" Revista Contribuciones a las Ciencias Sociales,» 2016.

[3] Luis Felipe Ariza Vesga, «Optimización de problemas de varios objetivos desde un enfoque de eficiencia energetica aplicado a redes celulares heterogeneas 5G usando un marco de conmutación de celadas pequeñas,» Bogotá, Colombia, 2020.

[4] Ericcson, «Ericsson Mobility Report,» Junio 2020.

[5] Jean-Gabriel Rémy, Charlotte Letamendia, LTE Standards, John Wiley & Sons, 10-Nov-2014.

[6] Erik Dahlman, Stefan Parkvall, and Johan Sköld, 4G LTE/LTE-Advanced for Mobile Broadband, Academic Press is an imprint of Elsevier, 2011.

[7] Mohammed Aly Abdrabou, Ashraf Diaa Eldien Elbayoumy, and Essam Abd EI-Wanis , «LTE Authentication Protocol (EPS-AKA) Weaknesses Solution,» IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS'15) , 2015.

[8] 3GPP TS 36.300 version 9.4.0 Release 9, «LTE, Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN),» 2010.

[9] Shinobu NAMBA, Takayuki WARABINO, Shoji KANEKO, «BBU-RRH Switching Schemes for Centralized RAN,» IEEE xplore, 2012.

[10] S. Perrin, «Envolving to an Open C-RAN Arquitecture for 5G,» Heavy Readind chite paper for Fujitsu, September 2017.

[11] Antonio de la Oliva, José Alberto Hernández, David Larrabeiti, and Arturo Azcorra, «An Overview of the CPRI Specification and Its Application to C-RAN-Based LTE Scenarios,» IEEE Communications Magazine, February 2016.

[12] José Manuel Galve, Ivana Gasulla, Salvador Sales and José Capmany, «Fronthaul design for Radio Access Networks using Multicore Fibers,» ISSN 1889-8297, Valencia, 2015.

[13] Prakash Suthar; Vivek Agarwal; Rajaneesh Sudhakar Shetty; Anil Jangam, «Migration and Interworking between 4G and 5G,» IEEE 3rd 5G World Forum (5GWF), 2020.

[14] Chih-Lin I, Han Li, Jouni Korhonen, Jinri Huang and Liuyan Han, «RAN Revolution With NGFI (xhaul) for 5G,» IEEE, JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 36, NO. 2, JANUARY 15, 2018, 2018.

[15] Parellel Wireless, «5G NR Logical Architecture and its Functional Splits,» [En línea]. Available: https://www.parallelwireless.com/wp-content/uploads/5GFunctionalSplits.pdf.

[16] S. Teral, «5G best choice architecture,» IHS Markit Technology, 30 January 2019.

[17] GSMA, «5G Implementation Guidelines: NSA Option 3,» February 2020.

[18] Brian Lavallée, «ciena,» 13 Agosto 2020. [En línea]. Available: https://www.ciena.com.mx/insights/articles/spotlight-on-4g-5g-backhaul-networks_es_LA.html.

[19] Nishant Kumar, Karun Rawat and Fadhel M. Ghannouchi, «Multi-Band All-Digital Transmission for 5G NG-RAN Communication,» IEEE 978-1-7281-7299, 2020.

[20] ITU-T (International Telecomunication Union), «Transport network support of IMT-2020/5G,» 9 February 2018.

[21] Comcores, «Design & Reuse, Opening the 5G Radio Interface,» [En línea]. Available: https://www.design-reuse.com/articles/48289/opening-the-5g-radio-interface.html.

[22] O.-R. Alliance, «O-RAN: Towards an Open and Smart RAN,» October 2018.

[23] Ericsson, «Security considerations of Open RAN,» August 2020.

[24] Solmaz Niknam, Abhishek Roy, Harpreet S. Dhillon, Sukhdeep Singh, Rahul Banerji, Jeffery H. Reed, Navrati Saxena, and Seungil Yoon, «Intelligent O-RAN for Beyond 5G and 6G Wireless Networks,» 17 May 2020.

[25] iGR, «Open RAN integration: Run with it,» April 2020.

[26] Telecom Infra Project, «TIP OpenRAN Project Group is Streamlined to Accelerate Development and Deployments,» October 2020.

[27] O.-R. C. Software, «Release Bronze,» [En línea]. Available: https://wiki.o-ran-sc.org/display/RSAC/Release+Bronze.

[28] Kubernetes, «What is Kubernetes?,» [En línea]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. [Último acceso: 20 11 2020].

[29] O.-R. C. Software, «Health-Check Use Case,» [En línea]. Available: https://wiki.o-ran-sc.org/display/RSAC/Health-Check+Use+Case.

[30] O-RAN, «Alliance O-RAN,» [En línea]. Available: https://wiki.o-ran-sc.org/display/GS/Getting+Started.

[31] O-RAN, «SMO Installation,» [En línea]. Available: https://wiki.o-ran-sc.org/display/GS/SMO+Installation?&#comments.

[32] P. Siva, «ONF SD-RAN: Open Source RIC, xApps & Integration with DU/CU,» August 2020.

[33] A. Umesh, Tatsuro Yajima, Turo Uchino y Suguru Okuyama, «Overview of O-RAN Fronthaul Specifications,» Radio Access Network Development Department.