

# Estudio de P4 y su papel en la telemetría de red en banda para las redes 5G y beyond

Edinson Montero Beltre

December 14, 2025

# Contents

<b>Dedicatoria</b>	<b>5</b>
<b>Acrónimos</b>	<b>6</b>
<b>1 Introducción</b>	<b>10</b>
1.1 Contexto general de las redes 5G y beyond . . . . .	10
1.2 Motivación del estudio . . . . .	10
1.3 Problemática en la visibilidad y monitoreo de redes 5G y beyond	11
1.4 Hipótesis y preguntas de investigación . . . . .	11
1.5 Objetivos generales y específicos . . . . .	12
1.6 Metodología de trabajo . . . . .	12
1.7 Estructura del documento . . . . .	13
1.8 Resumen de capítulos . . . . .	13
<b>2 Fundamentos de las Redes 5G y Beyond</b>	<b>15</b>
2.1 Evolución de tecnologías móviles . . . . .	15
2.1.1 De 1G a 6G: Hitos clave . . . . .	15
2.2 Arquitectura NR en 5G . . . . .	16
2.3 Arquitectura 5G Non Standalone (NSA) . . . . .	17
2.4 Arquitectura 5G Standalone (SA) . . . . .	19
2.4.1 Visión general de la arquitectura 5G SA . . . . .	19
2.4.2 Arquitectura basada en servicios (SBA) . . . . .	20
2.4.3 Interfaces HTTP REST . . . . .	21
2.4.4 Componentes de 5G Core (5GC) . . . . .	22
2.4.5 Componentes principales de 5GC . . . . .	22
2.4.6 Pila de protocolos del plano de control y plano de usuario	24
2.4.6.1 Plano de Control . . . . .	24
2.4.6.2 Plano de Usuario . . . . .	25
2.4.7 Interfaces clave en 5GC . . . . .	25
2.4.8 Tendencias emergentes en redes beyond 5G . . . . .	26

CONTENTS	2
<b>3 Telemetría y QoS en 5G</b>	<b>28</b>
3.1 Desafíos de Telemetría en Redes 5G . . . . .	29
3.1.1 Limitaciones de mecanismos tradicionales (SNMP, Net-Flow, sFlow) . . . . .	29
3.1.2 Requisitos de visibilidad en URLLC, eMBB y mMTC . . . . .	29
3.2 In-Band Network Telemetry (INT) . . . . .	29
3.2.1 Concepto y motivación . . . . .	29
3.2.2 Arquitectura INT: source, transit y sink . . . . .	29
3.2.3 Metadatos INT-MD y su estructura . . . . .	29
3.2.4 INT vs telemetría out-of-band . . . . .	29
3.3 Telemetría aplicada en 5G . . . . .	29
3.3.1 Relevancia de INT en el plano de control N2 . . . . .	29
3.3.2 Relevancia de INT en el plano de usuario N3 . . . . .	29
3.3.3 Métricas clave para tráfico GTP-U . . . . .	29
3.3.4 Desafíos y limitaciones en redes móviles . . . . .	29
3.4 Calidad de Servicio en 5G . . . . .	29
3.5 QoS basado en 5G Flujo . . . . .	29
3.5.1 Definición de flujo en 5G . . . . .	29
3.5.2 Identificación y clasificación de flujos . . . . .	29
3.5.3 Mecanismos de gestión de QoS por flujo . . . . .	29
3.6 Señalización de QoS en 5G . . . . .	29
3.7 Características de QoS en 5G . . . . .	29
<b>4 Fundamentos de P4 y Programación del Plano de Datos</b>	<b>30</b>
4.1 Introducción a P4 . . . . .	30
4.1.1 Historia y evolución . . . . .	30
4.1.2 Modelo de arquitectura PISA . . . . .	30
4.1.3 P4_14 vs P4_16 . . . . .	30
4.2 Herramientas y ecosistema P4 . . . . .	30
4.2.1 BMv2 como switch software . . . . .	30
4.2.2 P4C: compilador . . . . .	30
4.2.3 P4Runtime: control del data plane . . . . .	30
4.3 Implementación de INT en P4 . . . . .	30
4.3.1 Definición de cabeceras INT . . . . .	30
4.3.2 Parsing y deparsing en BMv2 . . . . .	30
4.3.3 Inyección hop-by-hop de metadatos . . . . .	30
4.3.4 Limitaciones del pipeline P4 . . . . .	30
<b>5 Entorno de Desarrollo Implementado</b>	<b>31</b>
5.1 Arquitectura general del sistema . . . . .	31
5.1.1 Diseño de la red 5G SA . . . . .	32

5.1.1.1	Diseño de alto nivel (HLD) . . . . .	33
5.1.1.2	Diseño de bajo nivel (LLD) . . . . .	33
5.2	Despliegue del Core 5G SA . . . . .	34
5.2.1	Configuracion de Core-Host . . . . .	34
5.2.1.1	Obtención del código fuente 5GC . . . . .	34
5.2.1.2	Configuracion de Docker Compose . . . . .	36
5.2.1.2.1	DB . . . . .	36
5.2.1.2.2	NRF . . . . .	37
5.2.1.2.3	AUSF . . . . .	38
5.2.1.2.4	NSSF . . . . .	38
5.2.1.2.5	PCF . . . . .	39
5.2.1.2.6	UDM . . . . .	39
5.2.1.2.7	UDR . . . . .	40
5.2.1.2.8	Despliegue del CHF . . . . .	41
5.2.1.2.9	Despliegue del NEF . . . . .	42
5.2.1.2.10	Despliegue del WebUI . . . . .	42
5.2.1.2.11	Configuracion de Red y Volumenes .	43
5.2.1.3	Construccion y despliegue . . . . .	43
5.2.2	Configuración de AMF . . . . .	45
5.2.2.1	Construcción de la imagen Docker del AMF .	46
5.2.2.2	Despliegue del contenedor AMF . . . . .	46
5.2.2.3	Registro del AMF en el NRF . . . . .	47
5.2.3	Configuración de SMF . . . . .	49
5.2.3.1	Instalacion del modulo GTP . . . . .	49
5.2.3.2	Obtención e instalación del código fuente SMF	55
5.2.3.3	Configuración de Docker Compose . . . . .	55
5.2.3.4	Registro del SMF en el NRF y conexión N4 con UPF . . . . .	56
5.2.4	Configuración de UPF . . . . .	58
5.2.4.1	Configuración de Docker Compose . . . . .	58
5.2.4.2	Registro del UPF en el NRF . . . . .	59
5.2.4.3	Conección N4 con SMF . . . . .	60
5.2.5	Integración con UERANSIM . . . . .	62
5.3	Implementación de la red de distribución P4 . . . . .	62
5.3.1	Diseño de los programas P4 . . . . .	62
5.3.2	Tablas y acciones configuradas . . . . .	62
5.3.3	Inserción INT-MD en tráfico N2 (SCTP) . . . . .	62
5.3.4	Inserción INT-MD en tráfico N3 (GTP-U) . . . . .	62
5.4	Servidor de recolección y análisis . . . . .	62
5.4.1	Implementación del sink INT . . . . .	62
5.4.2	Procesamiento y parsing de metadatos . . . . .	62

5.4.3	Modelo de base de datos . . . . .	62
5.4.4	Visualización en Grafana . . . . .	62
5.5	Tecnologías empleadas . . . . .	62
5.5.1	Docker / Docker Compose . . . . .	62
5.5.2	GNS3 / GNS3 VM . . . . .	62
5.5.3	Wireshark y herramientas auxiliares . . . . .	62
5.6	Despliegue del NGRAN con UERANSIM . . . . .	62
5.6.1	Despliegue de UERANSIM . . . . .	63
5.6.2	Configuración del gNB . . . . .	63
5.6.3	Configuración del UE . . . . .	64
5.7	codigo switch int source . . . . .	65
5.8	Implicación del código switch INT source . . . . .	73
<b>6</b>	<b>Resultados Experimentales</b>	<b>75</b>
6.1	Metodología de evaluación . . . . .	76
6.1.1	Escenarios de prueba . . . . .	76
6.1.2	Tráfico analizado . . . . .	76
6.1.3	Métricas recolectadas . . . . .	76
6.2	Validación funcional . . . . .	76
6.2.1	INT en tráfico N2 . . . . .	76
6.2.2	INT en tráfico N3 . . . . .	76
6.2.3	Recepción de metadatos en el servidor . . . . .	76
6.3	Resultados cuantitativos . . . . .	76
6.3.1	Latencia hop-by-hop . . . . .	76
6.3.2	Carga adicional introducida por INT . . . . .	76
6.3.3	Impacto en el plano de usuario . . . . .	76
6.4	Resultados cualitativos . . . . .	76
6.4.1	Dashboards obtenidos . . . . .	76
6.4.2	Interpretación de tendencias . . . . .	76
6.4.3	Problemas encontrados . . . . .	76
<b>7</b>	<b>Discusión</b>	<b>77</b>
7.1	Análisis crítico de los resultados . . . . .	77
7.1.1	Comparación con el estado del arte . . . . .	77
7.1.2	Validación de la hipótesis planteada . . . . .	77
7.2	Beneficios del uso de P4 en redes 5G . . . . .	77
7.3	Desafíos de escalabilidad . . . . .	77
7.4	Consideraciones de seguridad para INT . . . . .	77
7.5	Limitaciones del trabajo realizado . . . . .	77

CONTENTS	5
<b>8 Conclusiones y Trabajo Futuro</b>	<b>78</b>
8.1 Conclusiones principales . . . . .	78
8.2 Contribuciones del trabajo . . . . .	78
8.3 Limitaciones del estudio . . . . .	78
8.4 Líneas futuras de investigación . . . . .	78
8.4.1 INT en 6G . . . . .	78
8.4.2 UPF programable con P4 . . . . .	78
8.4.3 IA para análisis de telemetría . . . . .	78
<b>A Apéndices</b>	<b>79</b>
A.1 Código P4 . . . . .	79
A.2 Topologías de red . . . . .	79
A.3 Configuraciones del core 5G . . . . .	79
A.4 Capturas de tráfico . . . . .	79
A.5 Dashboards de Grafana . . . . .	79
A.6 Scripts y herramientas auxiliares . . . . .	79

# **Dedicatory**

A quienes me apoyaron en este proyecto.

# Acrónimos

Acrónimo	Significado
5G	Fifth Generation (Quinta Generación)
5GC	5G Core (Núcleo de Red 5G)
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
DNS	Domain Name System (Sistema de Nombres de Dominio)
INT	In-band Network Telemetry (Telemetría en Banda)
IP	Internet Protocol (Protocolo de Internet)
MIMO	Multiple Input Multiple Output (Entrada Múltiple Salida Múltiple)
NFV	Network Functions Virtualization (Virtualización de Funciones de Red)
P4	Programming Protocol-Independent Packet Processors
QoS	Quality of Service (Calidad de Servicio)
SDN	Software-Defined Networking (Redes Definidas por Software)
TCP	Transmission Control Protocol (Protocolo de Control de Transmisión)
TLS	Transport Layer Security (Seguridad de la Capa de Transporte)
UDP	User Datagram Protocol (Protocolo de Datagrama de Usuario)
UE	User Equipment (Equipo de Usuario)
UL	Uplink (Enlace Ascendente)
UMTS	Universal Mobile Telecommunications System (Sistema Universal de Telecomunicaciones Móviles)
UPF	User Plane Function (Función del Plano de Usuario)
URLLC	Ultra-Reliable Low-Latency Communications (Comunicaciones de Ultra Fiabilidad y Baja Latencia)

# **Contents**

# List of Figures

2.1	Opciones de implementación de NR en 5G: NSA y SA [1]. . . . .	17
2.2	Arquitectura 5GC basada en interfaces basadas en servicios [4]. . . . .	20
2.3	Arquitectura 5GC con interfaces punto a punto. [4]. . . . .	21
2.4	Pila de protocolos del plano de control en 5GC [1]. . . . .	24
2.5	Pila de protocolos del plano de usuario en 5GC. [1, 2]. . . . .	25
5.1	Arquitectura general . . . . .	32
5.2	Topología de red implementada en GNS3 . . . . .	33
5.3	Clonación del repositorio de free5gc . . . . .	34
5.4	Clonación de base de NFs adicionales . . . . .	35
5.5	Compilación de NFs adicionales . . . . .	36
5.6	Construcción de imágenes en Core-Host . . . . .	44
5.7	Despliegue de contenedores en Core-Host . . . . .	45
5.8	Construcción de la imagen Docker del AMF . . . . .	46
5.9	Registro del AMF en el NRF . . . . .	48
5.10	Verificación del registro del AMF en el NRF . . . . .	49
5.11	Instalación de kernel compatible con GTP . . . . .	50
5.12	Edición del archivo y actualización de /etc/default/grub . . . . .	51
5.13	Clonación del repositorio del módulo GTP . . . . .	52
5.14	Compilación e instalación del módulo GTP . . . . .	53
5.15	Carga y verificación del módulo GTP . . . . .	54
5.16	Verificación del módulo GTP en los logs del sistema . . . . .	55
5.17	Registro del SMF en el NRF . . . . .	57
5.18	UPF registro y preparacion de interfaz N3 . . . . .	60
5.19	UPF registro y preparacion de interfaz N3 . . . . .	61
5.20	Configuración del gNB en UERANSIM . . . . .	64
5.21	Configuración del UE en UERANSIM . . . . .	65

# List of Tables

5.1 Diseño de bajo nivel (LLD) . . . . .	33
--	----

# **Chapter 1**

## **Introducción**

### **1.1 Contexto general de las redes 5G y beyond**

Las redes de quinta generación (5G) representan un avance significativo en la evolución de las telecomunicaciones móviles, ofreciendo mayores velocidades, menor latencia y una capacidad mejorada para conectar dispositivos masivos. Con la creciente demanda de servicios en tiempo real y aplicaciones críticas, como la realidad aumentada, los vehículos autónomos y la telemedicina, en donde la latencia y la fiabilidad son esenciales debido a que cualquier retraso puede tener consecuencias graves, la necesidad de una visibilidad y monitoreo efectivos de la red se ha vuelto crucial. En esta evolución, las bajas latencias son cruciales para los nuevos servicios en la actualidad y los futuros, incluyendo la cuarta revolución industrial.

### **1.2 Motivación del estudio**

La complejidad y dinamismo de las redes 5G y superiores, plantean desafíos significativos para la gestión y el monitoreo de la red. Los métodos tradicionales de telemetría, como SNMP y NetFlow, a menudo no proporcionan la granularidad y la rapidez necesarias para detectar y resolver problemas en tiempo real otorgando una visibilidad de extremo a extremo sobre el estado de la red de forma precisa. La telemetría en banda (In-Band Network Telemetry, INT) emerge como una solución prometedora para abordar estas limitaciones, permitiendo la recopilación de datos detallados directamente desde los paquetes que atraviesan la red. Este estudio se centra en explorar el papel de P4, un lenguaje de programación para definir el comportamiento

de los dispositivos de red, en la implementación y optimización de soluciones de telemetría en banda para redes 5G y beyond.

### **1.3 Problemática en la visibilidad y monitoreo de redes 5G y beyond**

A medida que las redes 5G se vuelven más complejas, la visibilidad y el monitoreo efectivos se convierten en desafíos críticos. La diversidad de servicios y la naturaleza dinámica del tráfico generan dificultades para identificar cuellos de botella, latencias elevadas y otros problemas de rendimiento. Además, la necesidad de cumplir con estrictos requisitos de calidad de servicio (QoS) y experiencia del usuario (QoE) exige soluciones de monitoreo que puedan adaptarse rápidamente a las condiciones cambiantes de la red. La problemática radica en cómo implementar mecanismos de telemetría que sean capaces de proporcionar datos precisos y en tiempo real sin introducir una sobrecarga significativa en la red.

### **1.4 Hipótesis y preguntas de investigación**

La hipótesis central de este estudio es que la implementación de telemetría en banda utilizando P4 puede mejorar significativamente la visibilidad y el monitoreo de las redes 5G y beyond, permitiendo una gestión más eficiente y una mejor calidad de servicio. Las preguntas de investigación que guían este estudio incluyen:

- ¿Cómo puede P4 facilitar la implementación de soluciones de telemetría en banda en redes 5G y beyond?
- ¿Qué beneficios específicos ofrece la telemetría en banda en comparación con los métodos tradicionales de monitoreo?
- ¿Cuáles son los desafíos y limitaciones asociados con el uso de P4 para telemetría en banda en entornos 5G y beyond?
- ¿Cómo afecta la telemetría en banda al rendimiento general de la red y a la experiencia del usuario?

## 1.5 Objetivos generales y específicos

El objetivo general de este estudio es evaluar el papel de P4 en la implementación de telemetría en banda para mejorar la visibilidad y el monitoreo de las redes 5G y beyond. Los objetivos específicos incluyen:

- Analizar las capacidades de P4 para definir y programar el comportamiento de los dispositivos de red en el contexto de la telemetría en banda, específicamente INT-MD.
- Diseñar e implementar un prototipo de telemetría en banda (INT-MD) utilizando P4 en un entorno de red 5G SA simulado.
- Evaluar el rendimiento y la efectividad de la solución propuesta en términos de visibilidad, latencia y sobrecarga de la red.
- Identificar los desafíos y limitaciones encontrados durante la implementación y proponer posibles soluciones o mejoras.

## 1.6 Metodología de trabajo

Este estudio adoptará un enfoque experimental y analítico para investigar el papel de P4 en la telemetría en banda para redes 5G y beyond. La metodología incluirá las siguientes etapas:

- Revisión bibliográfica: Se realizará una revisión exhaustiva de la literatura existente sobre telemetría en banda, P4 y redes 5G y beyond para establecer un marco teórico sólido.
- Diseño del prototipo: Se diseñará un prototipo de telemetría en banda utilizando P4, definiendo las tablas y acciones necesarias para insertar metadatos INT-MD en el tráfico N2 (SCTP) y N3 (GTP-U).
- Implementación: El prototipo se implementará en un entorno emulado usando herramientas como GNS3 y VMware workstation que contará con Routers Cisco y switches P4 (BMv2), los cuales serán programados para soportar INT-MD y por último, un servidor sink para la recolección y análisis de los datos de telemetría usando influxDB y Grafana para representar los datos. Este prototipo integrará un core 5G SA básico y una red de distribución programable con P4.

- Evaluación: Se llevarán a cabo pruebas para evaluar el rendimiento del prototipo, midiendo métricas como la latencia, la sobrecarga de la red y la precisión de los datos recopilados, así como el impacto de la telemetría en banda en la calidad del servicio.
- Análisis de resultados: Los resultados obtenidos se analizarán críticamente para identificar beneficios, desafíos y áreas de mejora.

## 1.7 Estructura del documento

El documento se estructura en varios capítulos que abordan diferentes aspectos del estudio:

- Capítulo 1: **Introducción** - Presenta el contexto, la motivación, los objetivos y la metodología del estudio.
- Capítulo 2: **Redes 5G** - Proporciona una visión general de las redes 5G, sus características y desafíos.
- Capítulo 3: **Telemetría en Redes 5G** - Explora la necesidad de telemetría, los métodos tradicionales y la telemetría en banda (INT).
- Capítulo 4: **Fundamentos de P4** - Describe el lenguaje P4, su arquitectura y capacidades relevantes para la telemetría.
- Capítulo 5: **Entorno de Desarrollo** - Detalla el entorno utilizado para implementar y probar el prototipo de telemetría en banda.
- Capítulo 6: **Resultados** - Presenta los resultados obtenidos durante las pruebas del prototipo.
- Capítulo 7: **Discusión** - Analiza críticamente los resultados, comparándolos con el estado del arte y discutiendo beneficios y desafíos.
- Capítulo 8: **Conclusiones y Trabajo Futuro** - Resume las conclusiones principales, contribuciones, limitaciones y propone líneas futuras de investigación.

## 1.8 Resumen de capítulos

Cada capítulo del documento se enfoca en aspectos específicos del estudio, proporcionando una comprensión integral del papel de P4 en la telemetría

en banda para redes 5G. A lo largo del documento, se integran conceptos teóricos con aplicaciones prácticas, culminando en un análisis crítico de los resultados obtenidos y su relevancia para el campo de las telecomunicaciones móviles.

# Chapter 2

## Fundamentos de las Redes 5G y Beyond

En este capítulo se presentan los conceptos fundamentales de las redes 5G y beyond, incluyendo su evolución desde generaciones anteriores, las características clave de la arquitectura 5G, y las tendencias emergentes hacia futuras generaciones de redes móviles. Se abordan aspectos técnicos como la arquitectura del núcleo de red 5GC, las funciones principales del plano de control y del plano de usuario, así como las tecnologías habilitadoras que sustentan el despliegue y operación de estas redes avanzadas.

### 2.1 Evolución de tecnologías móviles

La evolución de las tecnologías móviles ha sido un proceso continuo que ha transformado la forma en que las personas se comunican y acceden a la información. Desde la primera generación (1G) hasta la actual quinta generación (5G) y más allá, cada etapa ha introducido avances significativos en términos de velocidad, capacidad, latencia y funcionalidad. A continuación, se presenta un resumen de la evolución de las tecnologías móviles:

#### 2.1.1 De 1G a 6G: Hitos clave

- **1G:** Introducción de la comunicación analógica.
- **2G:** Digitalización de la voz y servicios básicos de datos (SMS).
- **3G:** Introducción de datos móviles y servicios multimedia.
- **4G:** Redes IP y mayor velocidad de datos.

- **LTE y LTE-Advanced:** Mejoras en eficiencia espectral y latencia.
- **5G Non-Standalone (NSA):** Uso combinado de 4G y 5G para una transición suave. Aumentando la capacidad y velocidad de la red.
- **5G Standalone (SA):** Arquitectura completamente nueva basada en servicios y virtualización, permitiendo nuevas funcionalidades y mejoras en latencia y confiabilidad.
- **5G Advanced:** Mejoras en IA, eficiencia energética y soporte para nuevas aplicaciones, como XR y comunicaciones vehiculares.
- **Beyond-5G:** Investigación en tecnologías emergentes como comunicaciones cuánticas y redes holográficas, con miras a la futura generación 6G.
- **Visión de 6G:** Redes ultra confiables, latencia casi nula y capacidades de inteligencia artificial integradas de manera nativa.
- **Tecnologías emergentes:** Comunicaciones terahertz, redes auto-organizadas y computación en el borde (Edge Computing).
- **Aplicaciones futuras:** Realidad extendida (XR), ciudades inteligentes (Smart Cities) y redes de sensores masivos (IoT masivo).
- **Desafíos:** Seguridad, privacidad y sostenibilidad ambiental.

## 2.2 Arquitectura NR en 5G

3GPP introdujo 6 opciones de implementación para la tecnología New Radio (NR) como se muestra en la figura 2.1. Estas opciones se dividen en dos categorías principales: **Non-Standalone (NSA)** y **Standalone (SA)**. SA representa una arquitectura con NR como única tecnología de acceso, mientras que NSA combina NR con la infraestructura existente de 4G LTE. Las opciones 1, 2, 5 son implementaciones SA, mientras que las opciones 3, 4 y 7 son implementaciones NSA. Cabe destacar que la opción 1 es puramente LTE sin NR.

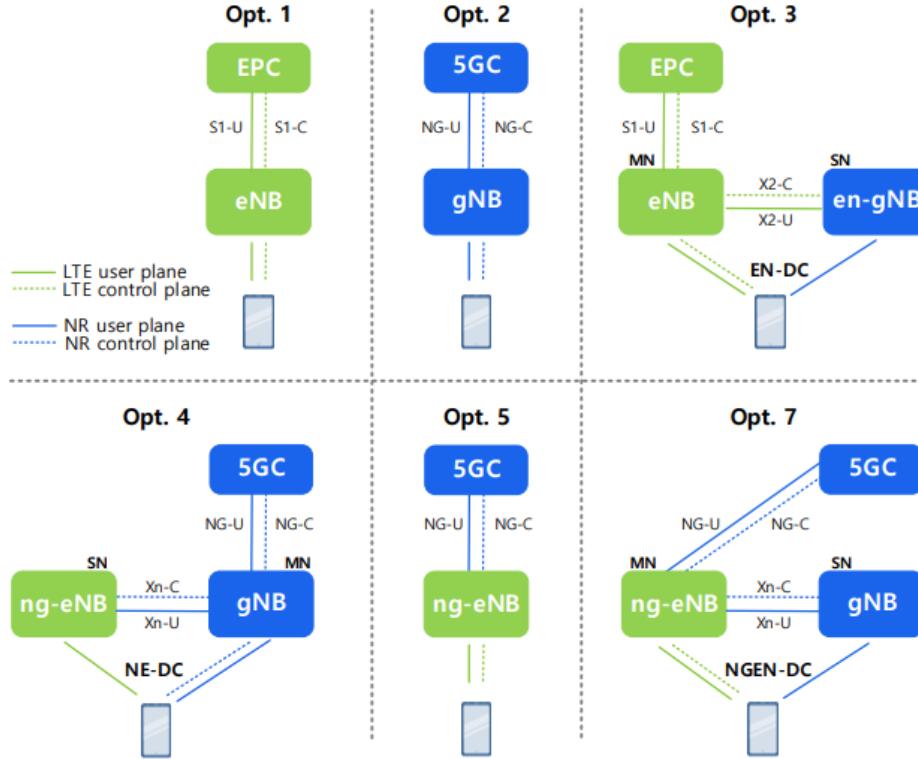


Figure 2.1: Opciones de implementación de NR en 5G: NSA y SA [1].

## 2.3 Arquitectura 5G Non Standalone (NSA)

La arquitectura 5G Non-Standalone (NSA) es una configuración de red que permite la coexistencia y colaboración entre las tecnologías 4G LTE y 5G NR. En esta arquitectura, la red 4G LTE actúa como la red principal para la señalización y el control, mientras que la red 5G NR se utiliza principalmente para el transporte de datos de alta velocidad. Esta coexistencia se logra mediante **Multi-Radio Dual Connectivity (MR-DC)**, descrito más adelante y que permite a los dispositivos de usuario (UE) conectarse simultáneamente a dos distintas generaciones de red de radio. Las opciones pueden ser:

- **EN-DC (E-UTRA-NR Dual Connectivity):** *Opcion 3*, donde el UE se conecta a una estación base LTE (eNodeB) como nodo maestro y a una estación base NR (gNodeB) como nodo secundario. El eNodeB

gestiona la señalización y el control, mientras que el gNodeB se utiliza principalmente para la transferencia de datos de alta velocidad.

- **NGEN-DC (NG-RAN E-UTRA-NR Dual Connectivity):** *Opcion 7*, similar a la opcion 3, pero con una integración más estrecha entre las estaciones base LTE y NR. En este modo, el eNodeB y el gNodeB están conectados a través de la interfaz Xn, lo que permite una mejor coordinación y gestión de recursos entre ambas tecnologías.
- **NE-DC (NR-E-UTRA Dual Connectivity):** *Opcion 4*, donde el UE se conecta a ng-eNB como MN y gNB como SN y ambas estaciones base están conectadas a través de la interfaz Xn. En este modo, el ng-eNB gestiona la señalización y el control, mientras que el gNB se utiliza para la transferencia de datos.
- **NR-DC (NR-NR Dual Connectivity):** *Opcion 2*, En la que UE se conecta a dos estaciones base NR (gNodeB) como nodos maestro y secundario. Ambas estaciones base están conectadas a través de la interfaz Xn, lo que permite una coordinación y gestión de recursos más eficiente entre ellas.

La arquitectura NSA permite a los proveedores de servicios móviles activar capacidades 5G de forma gradual sin necesidad de reemplazar completamente la infraestructura de núcleo de red existente. El despliegue de NSA comienza típicamente con la instalación de nuevas estaciones base NR que coexisten con las estaciones base LTE existentes. Los dispositivos del usuario (UE) que soportan tanto LTE como NR pueden conectarse simultáneamente a ambas redes, aprovechando la mejor señal disponible para la señalización de control a través de LTE y utilizando NR para el tráfico de datos cuando está disponible. Este enfoque dual proporciona beneficios inmediatos de rendimiento sin requerir una arquitectura de núcleo completamente nueva, lo que reduce significativamente los costos de inversión durante la transición. Desde el punto de vista del usuario final, la arquitectura NSA ofrece una experiencia mejorada en comparación con las redes LTE puras. Los usuarios pueden experimentar velocidades de descarga más altas (potencialmente en el rango de gigabits por segundo), menor latencia en aplicaciones específicas de datos, y mejor eficiencia general de la red. Sin embargo, las ventajas están limitadas principalmente al tráfico de datos, ya que los servicios de señalización y control siguen estando limitados por las especificaciones de LTE. Para casos de uso que requieren baja latencia extrema o requisitos ultra confiables (como comunicaciones críticas), la arquitectura NSA puede no ser suficiente, lo que subraya la necesidad eventual de migrar a 5G SA para alcanzar todas las capacidades de 5G.

## 2.4 Arquitectura 5G Standalone (SA)

### 2.4.1 Visión general de la arquitectura 5G SA

La arquitectura 5G Standalone (SA) representa una evolución significativa en comparación con las generaciones anteriores de redes móviles. A diferencia de las implementaciones Non-Standalone (NSA), que dependen de la infraestructura existente de 4G LTE, la arquitectura SA está diseñada desde cero para aprovechar al máximo las capacidades de la tecnología 5G. Esta arquitectura se basa en una serie de principios clave que permiten una mayor flexibilidad, eficiencia y capacidad para soportar una amplia gama de servicios y aplicaciones. Entre los aspectos más destacados de la arquitectura 5G SA se encuentran:

- **Red basada en servicios:** La arquitectura 5G SA adopta un enfoque basado en servicios, donde las funciones de red se implementan como servicios independientes que pueden ser orquestados y gestionados de manera flexible.
- **Virtualización y desagregación:** La arquitectura permite la virtualización de funciones de red (NFV) y la desagregación de hardware y software, lo que facilita la implementación en entornos de nube y mejora la escalabilidad.
- **Separación del plano de control y plano de usuario:** Esta separación permite una gestión más eficiente del tráfico y una mejor calidad de servicio (QoS) para diferentes tipos de aplicaciones.
- **Soporte para nuevas tecnologías:** La arquitectura 5G SA está diseñada para integrar tecnologías emergentes como la inteligencia artificial (IA), el edge computing y la Internet de las cosas (IoT).
- **Categorías de servicio: eMBB, URLLC y mMTC:** 5G define tres clases de servicio principales —**eMBB** (enhanced Mobile Broadband) para altas tasas de datos y capacidad; **URLLC** (Ultra-Reliable Low-Latency Communications) para comunicaciones con latencia muy baja y alta fiabilidad; y **mMTC** (massive Machine Type Communications) para conectividad masiva de dispositivos IoT de baja potencia y baja tasa. La arquitectura SA y el 5GC están pensados para soportar simultáneamente estas demandas diferenciadas /cite3gpp.ts.22.261.
- **Network Slicing:** Permite crear múltiples redes virtuales independientes (slices) sobre la misma infraestructura física, cada una con sus

propios requisitos de rendimiento, seguridad y gestión (por ejemplo, un slice para eMBB y otro para URLLC). Network Slicing se apoya en SBA, NFV y orquestación para instanciar, aislar y escalar slices según demanda.

### 2.4.2 Arquitectura basada en servicios (SBA)

La diferencia más destacada en 5G frente a las arquitecturas 3GPP anteriores es la adopción del concepto de interfaces basadas en servicios (**SBA**). Esto implica que las funciones de red que contienen la lógica y los mecanismos para procesar los flujos de señalización ya no se conectan mediante interfaces punto a punto, sino que **exponen sus capacidades como servicios** accesibles para otras funciones de red. En cada intercambio, una función actúa como **consumidora de servicios** y la otra como **proveedora de dichos servicios**. 2.2 muestra la arquitectura 5GC basada en SBA.

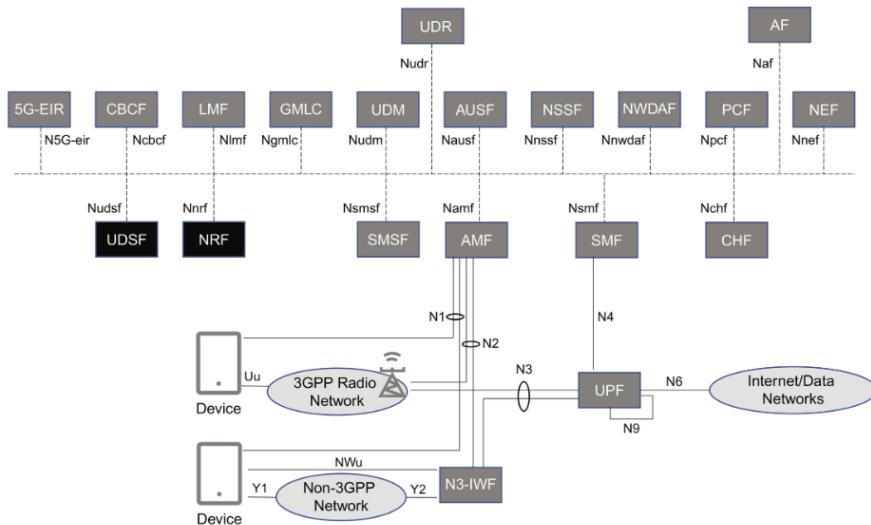


Figure 2.2: Arquitectura 5GC basada en interfaces basadas en servicios [4].

Las funciones de red en 5GC se comunican entre sí a través de una interfaz común llamada **Service-Based Interface (SBI)**. Esta interfaz utiliza protocolos estándar como HTTP/2 y RESTful APIs para facilitar la comunicación entre las funciones de red. La adopción de SBA permite una mayor flexibilidad y escalabilidad en la arquitectura de la red, ya que las funciones pueden ser desarrolladas, desplegadas y actualizadas de manera independiente. Además, esta arquitectura facilita la integración con tecnologías emergentes como la virtualización de funciones de red (NFV) y la computación

en la nube, permitiendo a los operadores de red adaptarse rápidamente a las demandas cambiantes del mercado y ofrecer nuevos servicios de manera más eficiente.

La arquitectura SBA también se puede representar como punto a punto, donde cada función de red se conecta directamente con las demás funciones que requieren sus servicios, utilizando la interfaz SBI para la comunicación. Como se muestra en la figura 2.3.

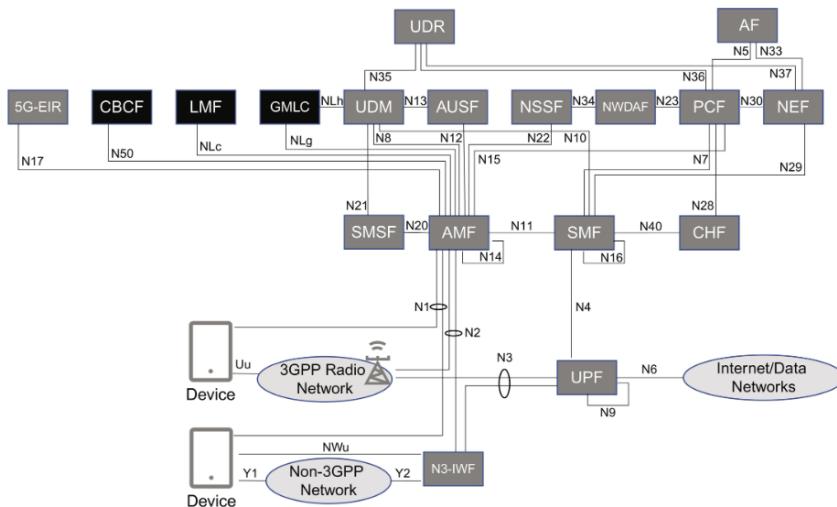


Figure 2.3: Arquitectura 5GC con interfaces punto a punto. [4].

### 2.4.3 Interfaces HTTP REST

En la arquitectura 5G Core (5GC), las funciones de red se comunican entre sí utilizando una interfaz basada en servicios conocida como Service-Based Interface (SBI). Esta interfaz utiliza el protocolo HTTP/2 junto con RESTful APIs para facilitar la comunicación y el intercambio de información entre las diferentes funciones de red. A continuación, se describen los aspectos clave de las interfaces HTTP REST en 5GC:

- **Protocolo HTTP/2:** 5GC utiliza HTTP/2 como el protocolo de transporte para las comunicaciones entre funciones de red. HTTP/2 ofrece varias ventajas sobre su predecesor, HTTP/1.1, incluyendo una mayor eficiencia en la multiplexación de solicitudes, compresión de encabezados y reducción de la latencia, lo que es crucial para las aplicaciones de baja latencia en 5G.
- **RESTful APIs:** Las funciones de red en 5GC exponen sus capacidades a través de RESTful APIs, que son interfaces basadas en prin-

cipios REST (Representational State Transfer). Estas APIs permiten a las funciones de red interactuar de manera sencilla y estandarizada, utilizando métodos HTTP como GET, POST, PUT y DELETE para realizar operaciones sobre los recursos.

- **Formato de datos JSON:** La comunicación entre funciones de red a través de las RESTful APIs generalmente utiliza JSON (JavaScript Object Notation) como formato de datos para el intercambio de información. JSON es ligero y fácil de leer, lo que facilita la integración entre diferentes sistemas y tecnologías.
- **Seguridad:** La seguridad es un aspecto crítico en las comunicaciones entre funciones de red. En 5GC, se implementan mecanismos de autenticación y autorización para garantizar que solo las funciones autorizadas puedan acceder a los servicios expuestos a través de las RESTful APIs. Además, se utilizan protocolos seguros como TLS (Transport Layer Security) para proteger la integridad y confidencialidad de los datos transmitidos.
- **Escalabilidad y flexibilidad:** La adopción de interfaces HTTP REST permite una mayor escalabilidad y flexibilidad en la arquitectura 5GC. Las funciones de red pueden ser desarrolladas, desplegadas y actualizadas de manera independiente, lo que facilita la adaptación a las demandas cambiantes del mercado y la incorporación de nuevas tecnologías.

#### 2.4.4 Componentes de 5G Core (5GC)

En 5G Core (5GC), los componentes tienen funciones específicas a diferencia de las arquitecturas anteriores, las cuales combinaban algunas funciones como por ejemplo, el MME y el SGW en una sola entidad llamada AMF.

#### 2.4.5 Componentes principales de 5GC

**NRF** El Network Repository Function es un componente clave en la arquitectura 5G Core (5GC) que actúa como un repositorio centralizado para la gestión y descubrimiento de servicios de red. Su función principal es mantener un registro actualizado de todas las funciones de red disponibles en la red 5G, permitiendo que otras funciones de red puedan descubrir y comunicarse con ellas de manera eficiente. El NRF mantiene un catálogo dinamizado de instancias de funciones de red (NF) con sus perfiles, incluyendo identidad (NF Instance ID), tipo de NF, direcciones de servicio (Service URIs) y

versiones de API soportadas, con soporte para mecanismos de heartbeat y actualización periódica del estado operacional [1].

**AMF** El Access and Mobility Management Function es una de las funciones clave en la arquitectura 5G Core (5GC) y se encarga de gestionar el acceso y la movilidad de los dispositivos de usuario (UE) en la red 5G. El AMF desempeña un papel fundamental en la gestión de la conexión del UE, la autenticación, la autorización y el control de movilidad, asegurando que los dispositivos puedan acceder a los servicios de red de manera eficiente y segura [1].

**SMF** El Session Management Function se encarga de gestionar las sesiones de datos del usuario (UE) en la red 5G. El SMF desempeña un papel fundamental en la configuración, mantenimiento y liberación de las sesiones de datos, asegurando que los dispositivos puedan acceder a los servicios de red de manera eficiente y segura [1].

**UPF** El User Plane Function es responsable de la gestión del plano de usuario (User Plane) en la arquitectura 5GC, actuando como el punto de anclaje para el procesamiento y enrutamiento de tráfico de datos. El UPF desempeña un papel crítico en la cadena de procesamiento de paquetes, implementando funciones de forwarding, encapsulación y aplicación de políticas a nivel de flujo [1].

**AUSF** El Authentication Server Function Se encarga de gestionar la autenticación de los dispositivos de usuario (UE) en la red 5G. El AUSF desempeña un papel fundamental en la seguridad de la red, asegurando que solo los dispositivos autorizados puedan acceder a los servicios de red [1].

**UDM** El Unified Data Management Es responsable de gestionar los datos de suscripción y la información del usuario en la red 5G. El UDM desempeña un papel fundamental en la gestión de la identidad del usuario, las políticas de acceso y las configuraciones de servicio, asegurando que los dispositivos puedan acceder a los servicios de red de manera eficiente y segura [1].

**PCF** El Policy Control Function Gestiona las políticas de control y calidad de servicio (QoS) en la red 5G. El PCF desempeña un papel fundamental en la aplicación de políticas de acceso, control de tráfico y gestión de recursos, asegurando que los dispositivos puedan acceder a los servicios de red de manera eficiente y segura.

**NSSF** El Network Slice Selection Function Se encarga de gestionar la selección y asignación de redes de corte (slices) para los dispositivos de usuario (UE) en la red 5G. El NSSF desempeña un papel fundamental en la implementación de redes de corte, que permiten a los operadores ofrecer servicios personalizados y optimizados para diferentes tipos de aplicaciones y usuarios.

**NEF** El Network Exposure Function Expone las capacidades y servicios de la red 5G a aplicaciones externas y terceros. El NEF desempeña un

papel fundamental en la facilitación de la interoperabilidad entre la red 5G y aplicaciones de terceros, permitiendo a los desarrolladores crear aplicaciones innovadoras que aprovechen las capacidades avanzadas de la red 5G.

**AF** El Application Function Gestiona las aplicaciones que interactúan con la red 5G. El AF desempeña un papel fundamental en la facilitación de la comunicación entre las aplicaciones y las funciones de red, permitiendo a los desarrolladores crear aplicaciones innovadoras que aprovechen las capacidades avanzadas de la red 5G.

#### 2.4.6 Pila de protocolos del plano de control y plano de usuario

Uno de los aspectos diferenciadores de la arquitectura 5G Core (5GC) es la separación clara entre el plano de control (Control Plane - CP) y el plano de usuario (User Plane - UP) mediante CUPS (Control and User Plane Separation). Esta separación permite una gestión más eficiente del tráfico y una mejor calidad de servicio (QoS) para diferentes tipos de aplicaciones. A continuación, se describen las pilas de protocolos utilizadas en ambos planos.

##### 2.4.6.1 Plano de Control

En el plano de control (Control Plane - CP) se gestionan las señales y la lógica necesarias para establecer, mantener y liberar las conexiones entre los dispositivos de usuario (UE) y la red, incluyendo protocolos como SCTP, NGAP, NAS, HTTP/2 y RESTful APIs. A continuación se muestra la pila de protocolos del plano de control de forma detallada 2.4.

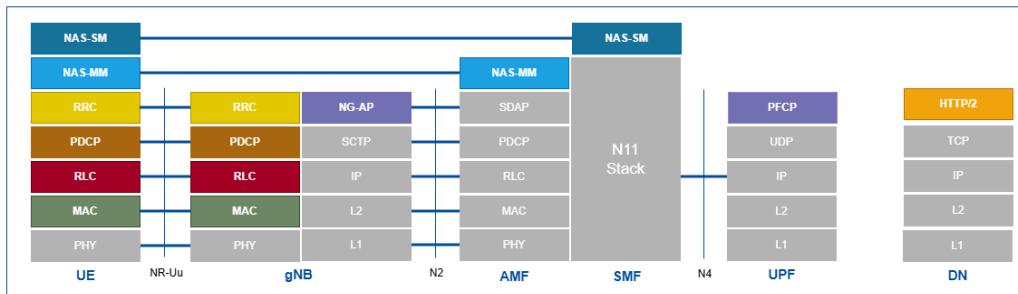


Figure 2.4: Pila de protocolos del plano de control en 5GC [1].

Para mas detalles de los acrónimos y protocolos ver la tabla ?? y ?? en el apéndice ??.

### 2.4.6.2 Plano de Usuario

En este plano se maneja el tráfico de datos real que fluye entre los dispositivos de usuario (UE) y la red. El plano de usuario se encarga de la transmisión eficiente y segura de los datos, asegurando que se cumplan los requisitos de calidad de servicio (QoS) y latencia para diferentes tipos de aplicaciones. A continuación se muestra la pila de protocolos del plano de usuario de forma detallada 2.5.

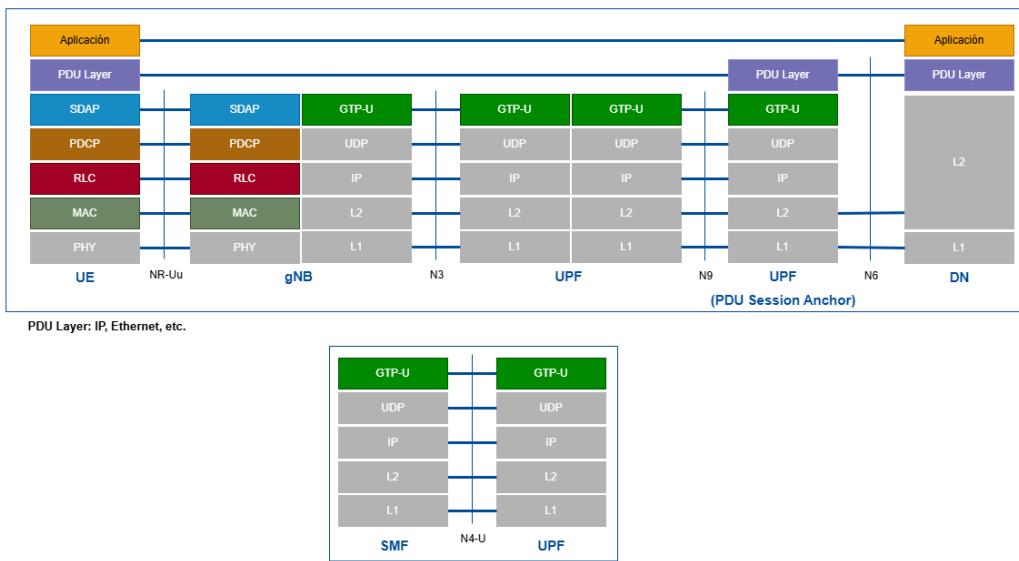


Figure 2.5: Pila de protocolos del plano de usuario en 5GC. [1, 2].

Al igual que en el plano de control, para mas detalles de los acrónimos y protocolos ver la tabla ?? y ?? en el apéndice ??.

### 2.4.7 Interfaces clave en 5GC

Las interfaces N1, N2, N3, N4, N6 y N9 son fundamentales en la arquitectura de las redes 5G, cada una desempeñando un papel crucial en la interconexión de dispositivos y la gestión de datos. A continuación, se describen cada una de estas interfaces, los protocolos asociados y los dispositivos que conectan.

- **N1:** Esta interfaz conecta el dispositivo de usuario (UE) con el Access and Mobility Management Function (AMF) en el plano de control. Utiliza el protocolo NAS (Non-Access Stratum) para la señalización y gestión de la conexión del UE.

- **N2:** Conecta la estación base 5G (gNodeB) con el AMF en el plano de control. Utiliza el protocolo NGAP (Next Generation Application Protocol) para la señalización y gestión de la conexión entre el gNodeB y el AMF.
- **N3:** Esta interfaz conecta la estación base 5G (gNodeB) con el User Plane Function (UPF) en el plano de usuario. Utiliza el protocolo GTP-U (GPRS Tunneling Protocol - User Plane) para la transmisión de datos entre el gNodeB y el UPF.
- **N4:** Conecta el SMF (Session Management Function) con el UPF en el plano de control. Utiliza el protocolo PFCP (Packet Forwarding Control Protocol) para la gestión y configuración del plano de usuario en el UPF.
- **N6:** Esta interfaz conecta el UPF con la red externa, como Internet o una red privada. Utiliza protocolos estándar como IP para la transmisión de datos entre el UPF y la red externa.
- **N9:** Conecta diferentes instancias de UPF entre sí en el plano de usuario. Utiliza el protocolo GTP-U para la transmisión de datos entre las instancias de UPF, permitiendo la movilidad y continuidad del servicio.
- **N11:** Conecta el AMF con el SMF en el plano de control. Utiliza interfaces basadas en servicios (SBI) y protocolos HTTP/2 y RESTful APIs para la comunicación entre el AMF y el SMF.

#### 2.4.8 Tendencias emergentes en redes beyond 5G

Las tendencias emergentes en redes beyond 5G incluyen:

- **Comunicaciones Terahertz (THz):** Utilización de frecuencias en el rango de terahercios para ofrecer velocidades de datos ultra altas y baja latencia.
- **Redes auto-organizadas (Self-Organizing Networks - SON):** Implementación de algoritmos de inteligencia artificial para la gestión automática y optimización de la red.
- **Computación en el borde (Edge Computing):** Despliegue de capacidades de procesamiento cerca del usuario final para reducir la latencia y mejorar la eficiencia.

- **Integración de inteligencia artificial (IA):** Uso de IA para la gestión de red, optimización del rendimiento y personalización de servicios.
- **Redes holográficas y realidad extendida (XR):** Soporte para aplicaciones avanzadas que requieren alta capacidad y baja latencia, como hologramas y experiencias inmersivas.
- **Sostenibilidad y eficiencia energética:** Desarrollo de tecnologías y prácticas que reduzcan el consumo energético y el impacto ambiental de las redes móviles.
- **Seguridad y privacidad mejoradas:** Implementación de mecanismos avanzados para proteger los datos y la privacidad de los usuarios en un entorno de red cada vez más complejo.
- **Redes cuánticas:** Investigación en la integración de tecnologías de comunicación cuántica para mejorar la seguridad y la capacidad de las redes móviles.
- **Conectividad masiva y ubicua:** Desarrollo de infraestructuras que permitan una conectividad continua y confiable en cualquier lugar y momento, soportando una amplia gama de dispositivos y aplicaciones.

Estas tendencias reflejan el compromiso de la industria de las telecomunicaciones para avanzar hacia redes móviles más inteligentes, eficientes y capaces de satisfacer las necesidades futuras de los usuarios y las aplicaciones.

# **Chapter 3**

## **Telemetría y QoS en 5G**

En 5GC, la telemetría juega un papel crucial en la monitorización y gestión del rendimiento de la red, especialmente en el contexto de las demandas crecientes de servicios como URLLC (Ultra-Reliable Low-Latency Communications), eMBB (enhanced Mobile Broadband) y mMTC (massive Machine Type Communications). La telemetría en banda (In-Band Network Telemetry, INT) permite la recopilación de datos en tiempo real directamente desde los paquetes que atraviesan la red, proporcionando una visibilidad detallada del estado de la red y facilitando la detección y resolución de problemas.

### 3.1 Desafíos de Telemetría en Redes 5G

- 3.1.1 Limitaciones de mecanismos tradicionales (SNMP, NetFlow, sFlow)
- 3.1.2 Requisitos de visibilidad en URLLC, eMBB y mMTC

### 3.2 In-Band Network Telemetry (INT)

- 3.2.1 Concepto y motivación
- 3.2.2 Arquitectura INT: source, transit y sink
- 3.2.3 Metadatos INT-MD y su estructura
- 3.2.4 INT vs telemetría out-of-band

### 3.3 Telemetría aplicada en 5G

- 3.3.1 Relevancia de INT en el plano de control N2
- 3.3.2 Relevancia de INT en el plano de usuario N3
- 3.3.3 Métricas clave para tráfico GTP-U
- 3.3.4 Desafíos y limitaciones en redes móviles

### 3.4 Calidad de Servicio en 5G

### 3.5 QoS basado en 5G Flujo

- 3.5.1 Definición de flujo en 5G
- 3.5.2 Identificación y clasificación de flujos
- 3.5.3 Mecanismos de gestión de QoS por flujo

### 3.6 Señalización de QoS en 5G

### 3.7 Características de QoS en 5G

# Chapter 4

## Fundamentos de P4 y Programación del Plano de Datos

### 4.1 Introducción a P4

- 4.1.1 Historia y evolución
- 4.1.2 Modelo de arquitectura PISA
- 4.1.3 P4\_14 vs P4\_16

### 4.2 Herramientas y ecosistema P4

- 4.2.1 BMv2 como switch software
- 4.2.2 P4C: compilador
- 4.2.3 P4Runtime: control del data plane

### 4.3 Implementación de INT en P4

- 4.3.1 Definición de cabeceras INT
- 4.3.2 Parsing y deparsing en BMv2
- 4.3.3 Inyección hop-by-hop de metadatos
- 4.3.4 Limitaciones del pipeline P4

# Chapter 5

## Entorno de Desarrollo Implementado

### 5.1 Arquitectura general del sistema

A diferencia de las tecnologías pasadas, 5G está diseñada para ser fácilmente desplegada en entornos virtualizados y basados en contenedores, lo que permite una mayor flexibilidad y escalabilidad. En este proyecto, se ha implementado un entorno de desarrollo que emula una red 5G Standalone (SA), virtualizando sus componentes principales mediante Docker y orquestando el Core con Docker Compose. El entorno de desarrollo se ha desplegado utilizando GNS3, GNS3 VM y VMware Workstation para crear una infraestructura base o underlay sobre la cual se han implementado otras máquinas virtuales que alojan los contenedores Docker. Esta infraestructura virtual proporciona la conectividad necesaria entre los componentes y permite emular una red de un MNO (Mobile Network Operator) real, con componentes tanto de red y VFs (Virtualized Functions) como de 5G. También cabe destacar que para los componentes de 5G se empleó el proyecto open source free5gc [3], el cual permite desplegar un core 5G SA completo en un entorno virtualizado con Docker. La siguiente figura muestra la arquitectura general del entorno de desarrollo implementado, destacando la integración entre GNS3, Docker y la máquina virtual en vmware 5.1.

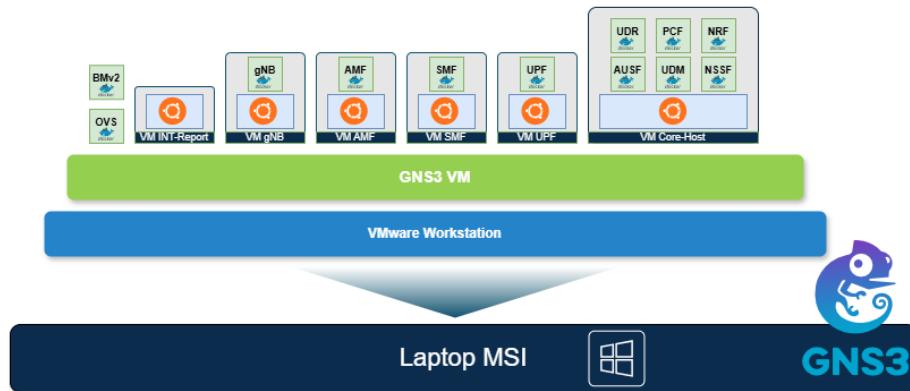


Figure 5.1: Arquitectura general

### 5.1.1 Diseño de la red 5G SA

Con respecto a la red de transporte, se ha implementado utilizando switches programables P4 para insertar metadatos de telemetría en banda (INT-MD) en el tráfico N2 (SCTP) y N3 (GTP-U), estos switches estan construidos con P4 para el reenvío de tráfico a nivel de L3. ademas de estos switches P4, tanto para el tráfico entre los Nodos en el Core como en la comunicación entre el NGRAN y R1, se ha empleado switches virtuales Open vSwitch (OVS) para gestionar y gestionar los paquetes ARP. La siguiente figura es un diseño de alto nivel (HLD) que muestra la topología de red implementada en GNS3 5.2.

### 5.1.1.1 Diseño de alto nivel (HLD)

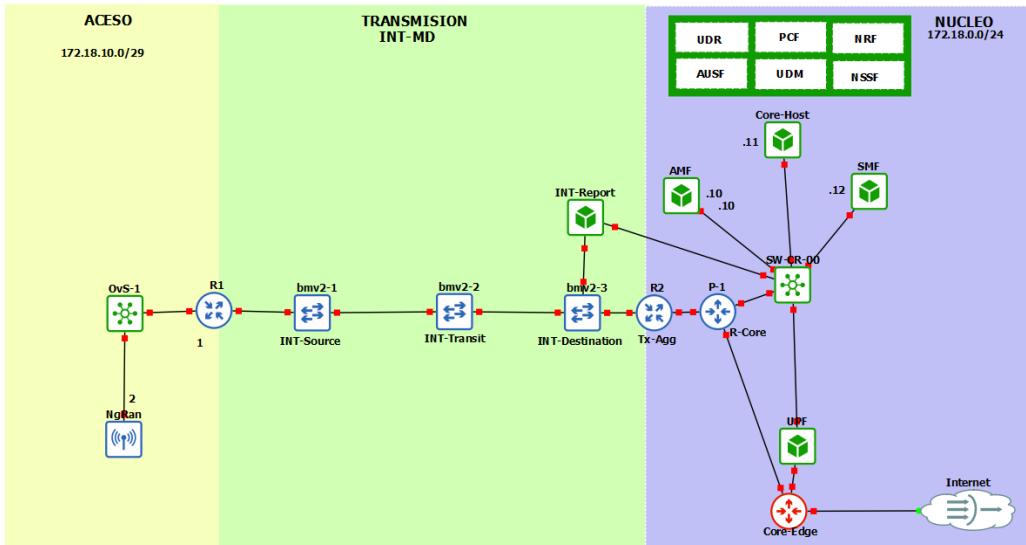


Figure 5.2: Topología de red implementada en GNS3

### 5.1.1.2 Diseño de bajo nivel (LLD)

La anterior tabla 5.1 detalla la configuración de red de cada contenedor Docker, direcciones IP y MAC asignadas, el tipo de red utilizado (macvlan), etc. Con esta configuración, se asegura una correcta interconexión y funcionamiento del Core 5G SA dentro del entorno virtualizado.

NF	Direccion IP	Mascara	Direccion MAC	Tipo de Red	Alias	Puertos	BD	Deps.
db	172.18.0.50	/24	22:e5:63:00:49:85	macvlan	db	27017	-	-
NRF	172.18.0.51	/24	3a:e5:07:18:07:fb	macvlan	nrf.free5gc.org	8000	db	db
AUSF	172.18.0.52	/24	aa:61:e3:b4:07:2e	macvlan	ausf.free5gc.org	8000	-	nrf
NSSF	172.18.0.53	/24	3e:74:95:c5:bc:4e	macvlan	nssf.free5gc.org	8000	-	nrf
PCF	172.18.0.54	/24	72:b2:53:59:fc:ce	macvlan	pcf.free5gc.org	8000	-	nrf
UDM	172.18.0.55	/24	9a:a1:ed:b3:60:8c	macvlan	udm.free5gc.org	8000	-	db, nrf
UDR	172.18.0.56	/24	ea:b1:9d:90:a4:d1	macvlan	udr.free5gc.org	8000	db	db, nrf
CHF	172.18.0.57	/24	ee:e6:15:22:6f:ad	macvlan	chf.free5gc.org	8000, 2122	db	db, nrf, webui
NEF	172.18.0.58	/24	e2:66:f3:0f:e0:e2	macvlan	nef.free5gc.org	8000	db	db, nrf
WebUI	172.18.0.59	/24	2a:5c:48:5c:aa:a0	macvlan	webui	2121 (ext: 5000)	-	db, nrf
AMF	172.18.0.20	/24	f2:ff:cb:34:07:b5	macvlan	amf.free5gc.org	38412 (SCTP), 8000	-	-
SMF	172.18.0.22	/24	ca:76:58:9c:8e:ec	macvlan	smf.free5gc.org	8000	-	-
UPF	172.18.0.24	/24	be:44:98:a4:e5:c4	macvlan	upf.free5gc.org	8000, 38412 (SCTP)	-	-
ueransim	172.18.10.3	/29	f6:b8:eb:4d:0f:3c	macvlan	ueransim.free5gc.org	38412 (SCTP)	-	-

Table 5.1: Diseño de bajo nivel (LLD)

## 5.2 Despliegue del Core 5G SA

Como se pudo apreciar en la figura 5.1, el Core 5G SA se ha desplegado utilizando contenedores Docker orquestados con Docker Compose dentro de maquinas virtuales alojadas en una maquina virtual principal (GNS3 VM). los componentes que interactuan mediante HTTP/2 se han desplegado en una VM con Ubuntu 20.04, mientras que el resto de componentes; AMF, SMF y UPF se han desplegado en maquinas virtuales separadas con la misma version de Ubuntu. Juntar los componentes que interactuan mediante HTTP/2 en una sola VM permite reducir la latencia y mejorar el rendimiento de las comunicaciones entre ellos.

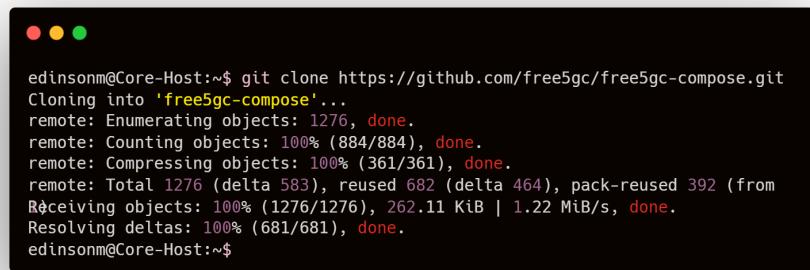
La configuracion de red de cada contenedor Docker se realizó con la modalidad de red **macvlan**, detallada en la tabla 5.1, que permite asignar una direccion IP en el mismo ranfo que la red fisica de la VM anfitriona. Esto facilita la comunicacion entre los contenedores y hace posible el establecimiento del enlace **N2** (SCTP) entre el NGRAN y el AMF.

### 5.2.1 Configuracion de Core-Host

Los detalles sobre la instalacion de Docker y Docker Compose no estan incluidos en este documento, pero se asume que el lector tiene conocimientos basicos sobre estas tecnologias. En caso contrario, puede consultar la documentacion oficial [?].

#### 5.2.1.1 Obtención del código fuente 5GC

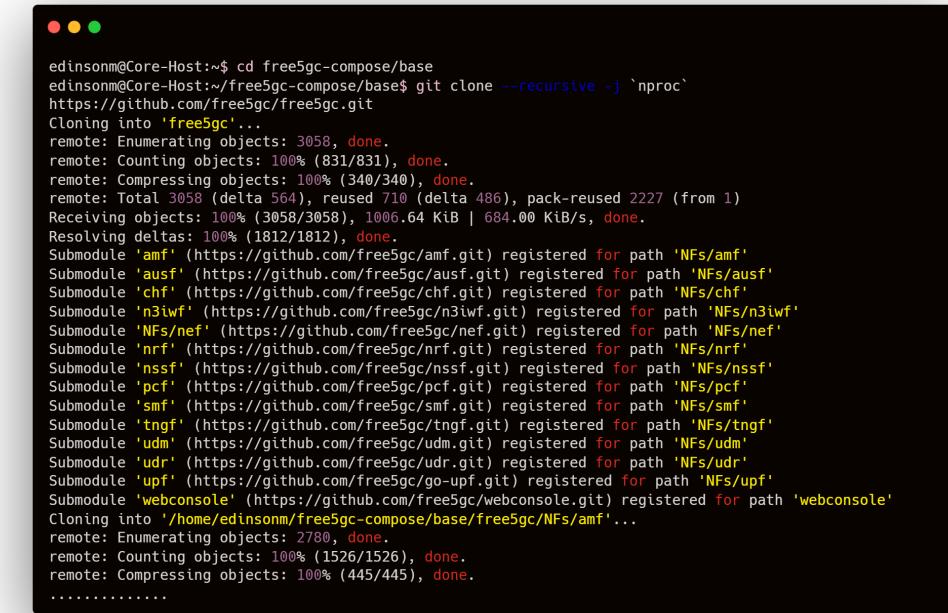
El primer paso para desplegar el Core 5G SA es clonar el repositorio oficial de free5gc desde GitHub como se muestra en la figura 5.3.



```
edinsonm@Core-Host:~$ git clone https://github.com/free5gc/free5gc-compose.git
Cloning into 'free5gc-compose'...
remote: Enumerating objects: 1276, done.
remote: Counting objects: 100% (884/884), done.
remote: Compressing objects: 100% (361/361), done.
remote: Total 1276 (delta 583), reused 682 (delta 464), pack-reused 392 (from
Receiving objects: 100% (1276/1276), 262.11 KiB | 1.22 MiB/s, done.
Resolving deltas: 100% (681/681), done.
edinsonm@Core-Host:~$
```

Figure 5.3: Clonación del repositorio de free5gc

Luego de clonar el repositorio, se procede a clonar las bases de las Network Functions (NFs) adicionales que no vienen incluidas en el repositorio principal de free5gc, como se muestra en la figura 5.4.



```
edinsonm@Core-Host:~$ cd free5gc-compose/base
edinsonm@Core-Host:~/free5gc-compose/base$ git clone --recursive -j `nproc` https://github.com/free5gc/free5gc.git
Cloning into 'free5gc'...
remote: Enumerating objects: 3058, done.
remote: Counting objects: 100% (831/831), done.
remote: Compressing objects: 100% (340/340), done.
remote: Total 3058 (delta 564), reused 710 (delta 486), pack-reused 2227 (from 1)
Receiving objects: 100% (3058/3058), 1,006.64 KiB | 684.00 KiB/s, done.
Resolving deltas: 100% (1812/1812), done.
Submodule 'amf' (https://github.com/free5gc/amf.git) registered for path 'NFs/amf'
Submodule 'ausf' (https://github.com/free5gc/ausf.git) registered for path 'NFs/ausf'
Submodule 'chf' (https://github.com/free5gc/chf.git) registered for path 'NFs/chf'
Submodule 'n3iwf' (https://github.com/free5gc/n3iwf.git) registered for path 'NFs/n3iwf'
Submodule 'NFS/nef' (https://github.com/free5gc/nef.git) registered for path 'NFS/nef'
Submodule 'nrf' (https://github.com/free5gc/nrf.git) registered for path 'NFs/nrf'
Submodule 'nssf' (https://github.com/free5gc/nssf.git) registered for path 'NFs/nssf'
Submodule 'pcf' (https://github.com/free5gc/pcf.git) registered for path 'NFs/pcf'
Submodule 'smf' (https://github.com/free5gc/smf.git) registered for path 'NFs/smf'
Submodule 'tngf' (https://github.com/free5gc/tngf.git) registered for path 'NFs/tngf'
Submodule 'udm' (https://github.com/free5gc/udm.git) registered for path 'NFs/udm'
Submodule 'udr' (https://github.com/free5gc/udr.git) registered for path 'NFs/udr'
Submodule 'upf' (https://github.com/free5gc/go-upf.git) registered for path 'NFs/upf'
Submodule 'webconsole' (https://github.com/free5gc/webconsole.git) registered for path 'webconsole'
Cloning into '/home/edinsonm/free5gc-compose/base/free5gc/NFs/amf'...
remote: Enumerating objects: 2780, done.
remote: Counting objects: 100% (1526/1526), done.
remote: Compressing objects: 100% (445/445), done.
.....
```

Figure 5.4: Clonación de base de NFs adicionales

Una vez clonado el repositorio principal y las bases de las NFs adicionales, se procede a complilar las NFs que en este caso, como es en el Core-Host, los compilamos todos como se puede ver en la figura 5.5.

```

edinsonm@Core-Host:~/free5gc-compose$ sudo make all
[sudo] password for edinsonm:
docker build -t 'free5gc' --base='latest' ./base
[+] Building 488.5s (7/7) FINISHED
=> [internal] load build definition from Dockerfile                               docker:default
=> [internal] load build context                                                 2.0s
=> => transferring dockerfile: 479B                                              0.4s
=> [internal] load metadata for docker.io/library/golang:1.21.8-bullseye          13.4s
=> [internal] load .dockerrcignore                                               0.1s
=> => transferring context: 2B                                                 0.0s
=> [1/3] FROM docker.io/library/golang:1.21.8-bullseye@sha256:81d98548f08e22a59c5c0be814acc099 331.2s
=> => resolve docker.io/library/golang:1.21.8-bullseye@sha256:81d98548f08e22a59c5c0be814acc0997f 0.2s
=> sha256:81d98548f08e22a59c5c0be814acc0997f3df3d313487adcfe1d3d962af3a43 1.64kB / 1.64kB 0.0s
=> sha256:2b64e533430ee61e12544f2155ff8b93557bdeee898e068b91045f279c9fcf0c 1.79kB / 1.79kB 0.0s
=> sha256:2fdf29b343a8e184d06dfa178ff3939bf79d25b9e97f40ac0e73d3bed33de0 2.88kB / 2.88kB 0.0s
=> sha256:ec335f17d0c74f7a270925cb1bbd29acc72ae904c6f4570f9ae369e3eebb64e 55.08MB / 55.08MB 192.9s
=> sha256:d2b4675e1918dcba5b8634d2459306d1f3b91f08c7293380f10585 15.76MB / 15.76MB 39.9s
=> sha256:f67b1746a83d257a6398cf8eec47bfa1f854670097ea4234f12857fcfd593 54.59MB / 54.59MB 143.8s
=> sha256:fb73e3517bc8fa1e9863e448b6ad3b54c5482345b15ce6ad28439cf1f01752 86.11MB / 86.11MB 259.4s
=> sha256:0635831cef590cc18d45a50a86211bcdcb7e2dc2aeac5a3d19671604e6b7e3d 67.01MB / 67.01MB 275.3s
=> sha256:579c95fc9df094db635a801a036ce19e0793e293b61b4621bac859996c4bbd47 173B / 173B 197.8s
=> sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cd577484a6d75e68dc38e8acc1 32B / 32B 199.8s
=> => extracting sha256:ec335f17d0c74f7a270925cb1bbd29acc72ae904c6f4570f9ae369e3eebb64ed 25.2s
=> => extracting sha256:d2b4675e1918dcba5b8634d2459306d1f3b91f08c7293380f10585 3.7s
=> => extracting sha256:7f67b1746a83d257a6398cf8eec47bfa1f854670097ea4234f12857fcfd5932 28.4s
=> => extracting sha256:fb73e3517bc8fa1e9863e448b6ad3b54c5482345b15ce6ad28439cf1f017523 23.0s
=> => extracting sha256:0635831cef590cc18d45a50a86211bcdcb7e2dc2aeac5a3d19671604e6b7e3d9 38.8s
=> => extracting sha256:579c95fc9df094db635a801a036ce19e0793e293b61b4621bac859996c4bbd47 0.0s
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cd577484a6d75e68dc38e8acc1 0.0s
=> [2/3] RUN apt-get update && apt-get -y install gcc cmake autoconf libtool pkg-config lib 94.4s
=> [3/3] RUN apt-get clean 6.3s
=> => exporting to image 33.6s
=> => exporting layers 32.5s
=> => writing image sha256:8e04e0e085a5583cc8ac3d2d3556b962ed01d02c2abe69d33a6579da1a75fea4 0.3s
=> => naming to docker.io/free5gc/base:latest 0.4s
.....
```

Figure 5.5: Compilación de NFs adicionales

### 5.2.1.2 Configuracion de Docker Compose

Luego de haberlos compilado, se procede a configurar cada NF de acuerdo con el diseño de bajo nivel (LLD) mostrado en la tabla 5.1. Como la VM Core-Host corre los contenedores UDR, UDM, PCF, NRF, AUSF, UDM y NSSF, DB, se despliegan creando un único archivo Docker Compose llamado *docker-compose-build-core-host.yaml*, el cual se mostrará contenedor por contenedor a continuación:

#### 5.2.1.2.1 DB

```

1 services:
2   db:
3     container_name: mongodb
4     image: mongo:3.6.8
5     command: mongod --port 27017
6     expose:
```

```

7      - "27017"
8  volumes:
9    - dbdata:/data/db
10 networks:
11   macvlan_net:
12     ipv4_address: 172.18.0.50
13     mac_address: "22:e5:63:00:49:85"
14     aliases:
15       - db

```

Código 5.1: Configuración del MongoDB en Docker Compose

### 5.2.1.2.2 NRF

```

1 free5gc-nrf:
2   container_name: nrf
3   build:
4     context: ./nf_nrf
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./nrf -c ./config/nrfcfg.yaml
8   expose:
9     - "8000"
10  volumes:
11    - ./config/nrfcfg.yaml:/free5gc/config/nrfcfg.yaml
12    - ./cert:/free5gc/cert
13  environment:
14    DB_URI: mongodb://db/free5gc
15    GIN_MODE: release
16  networks:
17    macvlan_net:
18      ipv4_address: 172.18.0.51
19      mac_address: "3a:e5:07:18:07:fb"
20      aliases:
21        - nrf.free5gc.org
22  extra_hosts:
23    - "amf.free5gc.org:172.18.0.20"
24    - "smf.free5gc.org:172.18.0.22"
25    - "upf.free5gc.org:172.18.0.23"
26  ports:
27    - "8000"
28
29  depends_on:
30    - db

```

Código 5.2: Configuración del NRF en Docker Compose

### 5.2.1.2.3 AUSF

```

1 free5gc-ausf:
2   container_name: ausf
3   build:
4     context: ./nf_ausf
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./ausf -c ./config/ausfcfg.yaml
8   expose:
9     - "8000"
10  volumes:
11    - ./config/ausfcfg.yaml:/free5gc/config/ausfcfg.yaml
12    - ./cert:/free5gc/cert
13  environment:
14    GIN_MODE: release
15  networks:
16    macvlan_net:
17      ipv4_address: 172.18.0.52
18      mac_address: "aa:61:e3:b4:07:2e"
19      aliases:
20        - ausf.free5gc.org
21  extra_hosts:
22    - "amf.free5gc.org:172.18.0.20"
23    - "smf.free5gc.org:172.18.0.22"
24    - "upf.free5gc.org:172.18.0.23"
25
26 depends_on:
27   - free5gc-nrf

```

Código 5.3: Configuración del AUSF en Docker Compose

### 5.2.1.2.4 NSSF

```

1 free5gc-nssf:
2   container_name: nssf
3   build:
4     context: ./nf_nssf
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./nssf -c ./config/nssfcfg.yaml
8   expose:
9     - "8000"
10  volumes:
11    - ./config/nssfcfg.yaml:/free5gc/config/nssfcfg.yaml
12    - ./cert:/free5gc/cert
13  environment:
14    GIN_MODE: release

```

```

15 networks:
16   macvlan_net:
17     ipv4_address: 172.18.0.53
18     mac_address: "3e:74:95:c5:bc:4e"
19     aliases:
20       - nssf.free5gc.org
21   extra_hosts:
22     - "amf.free5gc.org:172.18.0.20"
23     - "smf.free5gc.org:172.18.0.22"
24     - "upf.free5gc.org:172.18.0.23"
25   depends_on:
26     - free5gc-nrf

```

Código 5.4: Configuración del NSSF en Docker Compose

### 5.2.1.2.5 PCF

```

1 free5gc-pcf:
2   container_name: pcf
3   build:
4     context: ./nf_pcf
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./pcf -c ./config/pcfcfg.yaml
8   expose:
9     - "8000"
10  volumes:
11    - ./config/pcfcfg.yaml:/free5gc/config/pcfcfg.yaml
12    - ./cert:/free5gc/cert
13  environment:
14    GIN_MODE: release
15  networks:
16    macvlan_net:
17      ipv4_address: 172.18.0.54
18      mac_address: "72:b2:53:59:fc:ce"
19      aliases:
20        - pcf.free5gc.org
21   extra_hosts:
22     - "amf.free5gc.org:172.18.0.20"
23     - "smf.free5gc.org:172.18.0.22"
24     - "upf.free5gc.org:172.18.0.23"
25   depends_on:
26     - free5gc-nrf

```

Código 5.5: Configuración del PCF en Docker Compose

### 5.2.1.2.6 UDM

```

1 free5gc-udm:
2   container_name: udm
3   build:
4     context: ./nf_udm
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./udm -c ./config/udmcfg.yaml
8   expose:
9     - "8000"
10  volumes:
11    - ./config/udmcfg.yaml:/free5gc/config/udmcfg.yaml
12    - ./cert:/free5gc/cert
13  environment:
14    GIN_MODE: release
15  networks:
16    macvlan_net:
17      ipv4_address: 172.18.0.55
18      mac_address: "9a:a1:ed:b3:60:8c"
19      aliases:
20        - udm.free5gc.org
21  extra_hosts:
22    - "amf.free5gc.org:172.18.0.20"
23    - "smf.free5gc.org:172.18.0.22"
24    - "upf.free5gc.org:172.18.0.23"
25  depends_on:
26    - db
27    - free5gc-nrf

```

Código 5.6: Configuración del UDM en Docker Compose

### 5.2.1.2.7 UDR

```

1 free5gc-udr:
2   container_name: udr
3   build:
4     context: ./nf_udr
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./udr -c ./config/udrcfg.yaml
8   expose:
9     - "8000"
10  volumes:
11    - ./config/udrcfg.yaml:/free5gc/config/udrcfg.yaml
12    - ./cert:/free5gc/cert
13  environment:
14    DB_URI: mongodb://db/free5gc
15    GIN_MODE: release

```

```

16 networks:
17   macvlan_net:
18     ipv4_address: 172.18.0.56
19     mac_address: "ea:b1:9d:90:a4:d1"
20     aliases:
21       - udr.free5gc.org
22 extra_hosts:
23   - "amf.free5gc.org:172.18.0.20"
24   - "smf.free5gc.org:172.18.0.22"
25   - "upf.free5gc.org:172.18.0.23"
26 depends_on:
27   - db
28   - free5gc-nrf

```

Código 5.7: Configuración del UDR en Docker Compose

### 5.2.1.2.8 Despliegue del CHF

```

1 free5gc-chf:
2   container_name: chf
3   build:
4     context: ./nf_chf
5     args:
6       DEBUG_TOOLS: "false"
7   command: ./chf -c ./config/chfcfg.yaml
8   expose:
9     - "8000"
10    - "2122"
11   volumes:
12     - ./config/chfcfg.yaml:/free5gc/config/chfcfg.yaml
13     - ./cert:/free5gc/cert
14   environment:
15     DB_URI: mongodb://db/free5gc
16     GIN_MODE: release
17   networks:
18     macvlan_net:
19       ipv4_address: 172.18.0.57
20       mac_address: "ee:e6:15:22:6f:ad"
21       aliases:
22         - chf.free5gc.org
23 extra_hosts:
24   - "amf.free5gc.org:172.18.0.20"
25   - "smf.free5gc.org:172.18.0.22"
26   - "upf.free5gc.org:172.18.0.23"
27 depends_on:
28   - db
29   - free5gc-nrf
30   - free5gc-webui

```

Código 5.8: Configuración del CHF en Docker Compose

### 5.2.1.2.9 Despliegue del NEF

```

1 free5gc-nef:
2   container_name: nef
3   build:
4     context: ./nf_nef
5     args:
6       DEBUG_TOOLS: "false"
7     command: ./nef -c ./config/nefcfg.yaml
8     expose:
9       - "8000"
10    volumes:
11      - ./config/nefcfg.yaml:/free5gc/config/nefcfg.yaml
12      - ./cert:/free5gc/cert
13    environment:
14      GIN_MODE: release
15    networks:
16      macvlan_net:
17        ipv4_address: 172.18.0.58
18        mac_address: "e2:66:f3:0f:e0:e2"
19        aliases:
20          - nef.free5gc.org
21    extra_hosts:
22      - "amf.free5gc.org:172.18.0.20"
23      - "smf.free5gc.org:172.18.0.22"
24      - "upf.free5gc.org:172.18.0.23"
25    depends_on:
26      - db
27      - free5gc-nrf

```

Código 5.9: Configuración del NEF en Docker Compose

### 5.2.1.2.10 Despliegue del WebUI

A pesar de que el contenedor WebUI no es un Network Function (NF) del Core 5G SA, se ha incluido en el mismo archivo Docker Compose para facilitar su despliegue y gestión. El WebUI proporciona una interfaz gráfica para monitorizar y gestionar el Core 5G, facilitando la administración de la red.

```

1 free5gc-webui:
2   container_name: webui
3   build:
4     context: ./webui

```

```

5   args:
6     DEBUG_TOOLS: "false"
7   command: ./webui -c ./config/webuicfg.yaml
8   expose:
9     - "2121"
10  volumes:
11    - ./config/webuicfg.yaml:/free5gc/config/webuicfg.yaml
12  environment:
13    - GIN_MODE=release
14  ports:
15    - "5000:5000"
16  networks:
17    macvlan_net:
18      ipv4_address: 172.18.0.59
19      mac_address: "2a:5c:48:5c:aa:a0"
20      aliases:
21        - webui
22  extra_hosts:
23    - "amf.free5gc.org:172.18.0.20"
24    - "smf.free5gc.org:172.18.0.22"
25    - "upf.free5gc.org:172.18.0.23"
26  depends_on:
27    - db
28    - free5gc-nrf

```

Código 5.10: Configuración del WebUI en Docker Compose

### 5.2.1.2.11 Configuracion de Red y Volumenes

```

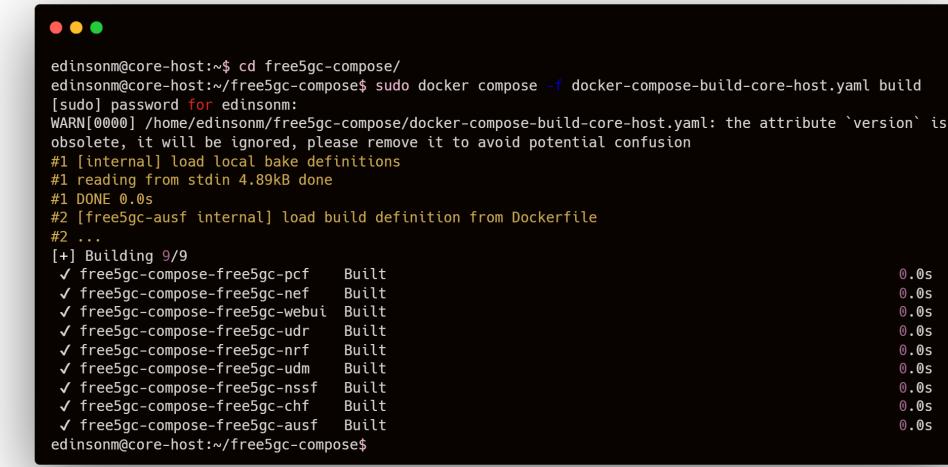
1 networks:
2   macvlan_net:
3     external: true
4
5 volumes:
6   dbdata:

```

Código 5.11: Configuración de red y volúmenes en Docker Compose

### 5.2.1.3 Construccion y despliegue

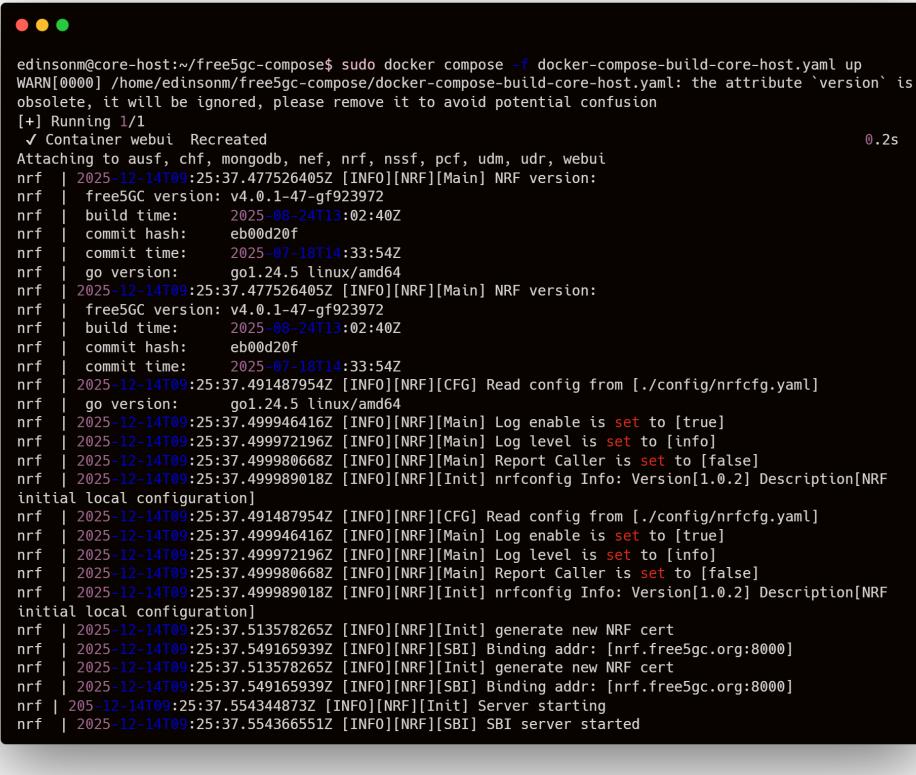
Una vez definido el archivo Docker Compose con la configuración de todas las NFs de Core-Host, se procede a construir las imágenes de acuerdo con el archivo Docher Compose como se puede apreciar en la figura 5.6.

A screenshot of a terminal window on a Mac OS X desktop. The window has a dark background with red, yellow, and green title bar buttons. The terminal text is white. The command run is 'sudo docker compose -f docker-compose-build-core-host.yaml build'. It shows a warning about the 'version' attribute being obsolete. Then it lists nine services being built, each with a checkmark and a duration of 0.0s. The services are: free5gc-pcf, free5gc-nef, free5gc-webui, free5gc-udr, free5gc-nrf, free5gc-free5gc-udm, free5gc-nssf, free5gc-chf, and free5gc-ausf.

```
edinsonm@core-host:~/free5gc-compose/
edinsonm@core-host:~/free5gc-compose$ sudo docker compose -f docker-compose-build-core-host.yaml build
[sudo] password for edinsonm:
WARN[0000] /home/edinsonm/free5gc-compose/docker-compose-build-core-host.yaml: the attribute `version` is
obsolete, it will be ignored, please remove it to avoid potential confusion
#1 [internal] load local bake definitions
#1 reading from stdin 4.89kB done
#1 DONE 0.0s
#2 [free5gc-ausf internal] load build definition from Dockerfile
#2 ...
[+] Building 9/9
  ✓ free5gc-compose-free5gc-pcf    Built          0.0s
  ✓ free5gc-compose-free5gc-nef   Built          0.0s
  ✓ free5gc-compose-free5gc-webui Built          0.0s
  ✓ free5gc-compose-free5gc-udr   Built          0.0s
  ✓ free5gc-compose-free5gc-nrf   Built          0.0s
  ✓ free5gc-compose-free5gc-udm   Built          0.0s
  ✓ free5gc-compose-free5gc-nssf  Built          0.0s
  ✓ free5gc-compose-free5gc-chf   Built          0.0s
  ✓ free5gc-compose-free5gc-ausf  Built          0.0s
edinsonm@core-host:~/free5gc-compose$
```

Figure 5.6: Construcción de imágenes en Core-Host

Luego de construir las imágenes y como se puede apreciar en la figura 5.7, se procede a desplegar los contenedores de los cuales se destaca el log del NRF, quien orquesta todo el Core 5G basado en la arquitectura SBI descrita en la figura 2.3. El resto de logs de los demás contenedores no se muestran por cuestiones de espacio.



```

edinsonm@core-host:~/free5gc-compose$ sudo docker compose -f docker-compose-build-core-host.yaml up
WARN[0000] /home/edinsonm/free5gc-compose/docker-compose-build-core-host.yaml: the attribute `version` is
obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 1/1
  ✓ Container webui Recreated
Attaching to ausf, chf, mongodb, nef, nssf, pcf, udm, udr, webui
nrf | 2025-12-14T09:25:37.477526405Z [INFO][NRF][Main] NRF version: 0.2s
nrf | free5GC version: v4.0.1-47-gf923972
nrf | build time: 2025-08-24T13:02:40Z
nrf | commit hash: eb00d20f
nrf | commit time: 2025-07-18T14:33:54Z
nrf | go version: go1.24.5 linux/amd64
nrf | 2025-12-14T09:25:37.477526405Z [INFO][NRF][Main] NRF version:
nrf | free5GC version: v4.0.1-47-gf923972
nrf | build time: 2025-08-24T13:02:40Z
nrf | commit hash: eb00d20f
nrf | commit time: 2025-07-18T14:33:54Z
nrf | 2025-12-14T09:25:37.491487954Z [INFO][NRF][CFG] Read config from ./config/nrfcfg.yaml
nrf | go version: go1.24.5 linux/amd64
nrf | 2025-12-14T09:25:37.499946416Z [INFO][NRF][Main] Log enable is set to [true]
nrf | 2025-12-14T09:25:37.499972196Z [INFO][NRF][Main] Log level is set to [info]
nrf | 2025-12-14T09:25:37.499980668Z [INFO][NRF][Main] Report Caller is set to [false]
nrf | 2025-12-14T09:25:37.499989018Z [INFO][NRF][Init] nrfconfig Info: Version[1.0.2] Description[NRF
initial local configuration]
nrf | 2025-12-14T09:25:37.491487954Z [INFO][NRF][CFG] Read config from ./config/nrfcfg.yaml
nrf | 2025-12-14T09:25:37.499946416Z [INFO][NRF][Main] Log enable is set to [true]
nrf | 2025-12-14T09:25:37.499972196Z [INFO][NRF][Main] Log level is set to [info]
nrf | 2025-12-14T09:25:37.499980668Z [INFO][NRF][Main] Report Caller is set to [false]
nrf | 2025-12-14T09:25:37.499989018Z [INFO][NRF][Init] nrfconfig Info: Version[1.0.2] Description[NRF
initial local configuration]
nrf | 2025-12-14T09:25:37.513578265Z [INFO][NRF][Init] generate new NRF cert
nrf | 2025-12-14T09:25:37.549165939Z [INFO][NRF][SBI] Binding addr: [nrf.free5gc.org:8000]
nrf | 2025-12-14T09:25:37.513578265Z [INFO][NRF][Init] generate new NRF cert
nrf | 2025-12-14T09:25:37.549165939Z [INFO][NRF][SBI] Binding addr: [nrf.free5gc.org:8000]
nrf | 2025-12-14T09:25:37.554344873Z [INFO][NRF][Init] Server starting
nrf | 2025-12-14T09:25:37.554366551Z [INFO][NRF][SBI] SBI server started

```

Figure 5.7: Despliegue de contenedores en Core-Host

### 5.2.2 Configuración de AMF

El proceso de obtención, compilación y despliegue del AMF es similar al realizado para el Core-Host. Para fines de ahorrar espacio, no se mostrará el proceso de clonación. En cambio se mostrará la construcción de la imagen Docker del AMF y su posterior despliegue.

### 5.2.2.1 Construcción de la imagen Docker del AMF

```
edinsonm@amf:~/free5gc-compose$ sudo make amf
docker build -t 'free5gc'/'base':'latest' ./base
[+] Building 2.4s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 479B
=> [internal] load metadata for docker.io/library/golang:1.21.8-bullseye
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/3] FROM docker.io/library/golang:1.21.8-bullseye@sha256:81d98548f08e22a59c5c0be814acc0997f
=> CACHED [2/3] RUN apt-get update && apt-get -y install gcc cmake autoconf libtool pkg-conf
=> CACHED [3/3] RUN apt-get clean
=> exporting to image
=> => exporting layers
=> => writing image sha256:8e04e0e085a5583cc8ac3d2d3556b962ed01d02c2abe69d33a6579da1a75fea4
=> => naming to docker.io/free5gc/base:latest

1 warning found (use docker --debug to expand):
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line
  docker image ls 'free5gc'/'base':'latest'
REPOSITORY        TAG           IMAGE ID      CREATED          SIZE
free5gc/base      latest        8e04e0e085a5   22 hours ago   849MB
docker build --build-arg F5GC_MODULE=amf -t 'free5gc'/'amf-base:'latest' -f ./base/Dockerfile.nf ./base
[+] Building 387.8s (15/15) FINISHED
=> [internal] load build definition from Dockerfile.nf
=> => transferring dockerfile: 767B
=> WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 5)
=> [internal] load metadata for docker.io/library/alpine:3.15
=> [internal] load metadata for docker.io/free5gc/base:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [stage-1 1/6] FROM docker.io/library/alpine:3.15@sha256:19b4bcc4f60e99dd5ebdca0cbce22c503bbcf
=> [my-base 1/3] FROM docker.io/free5gc/base:latest
=> [internal] load build context
=> => transferring context: 142.30kB
=> CACHED [stage-1 2/6] WORKDIR /free5gc
=> CACHED [stage-1 3/6] RUN mkdir -p cert/ public
=> CACHED [my-base 2/3] COPY free5gc/ /go/src/free5gc/
=> [my-base 3/3] RUN cd /go/src/free5gc && make amf
=> [stage-1 4/6] COPY --from=my-base /go/src/free5gc/bin/ ../
=> [stage-1 5/6] COPY --from=my-base /go/src/free5gc/config/* ./config/
=> [stage-1 6/6] COPY --from=my-base /go/src/free5gc/cert/* ./cert/
=> exporting to image
=> => exporting layers
=> => writing image sha256:acd76169aa75d716f3875b30dba269789c10595cc38315181c9374d6a35dee
=> => naming to docker.io/free5gc/amf-base:latest

3 warnings found (use docker --debug to expand):
- UndefinedVar: Usage of undefined variable '$F5GC_MODULE' (line 23)
- FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 5)
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line
edinsonm@amf:~/free5gc-compose$
```

Figure 5.8: Construcción de la imagen Docker del AMF

### 5.2.2.2 Despliegue del contenedor AMF

Para el despliegue del contenedor AMF, se crea un archivo Docker Compose llamado *docker-compose-build-amf.yaml* con la configuración del AMF mostrado en el siguiente código 5.12.

## 1 services:

```

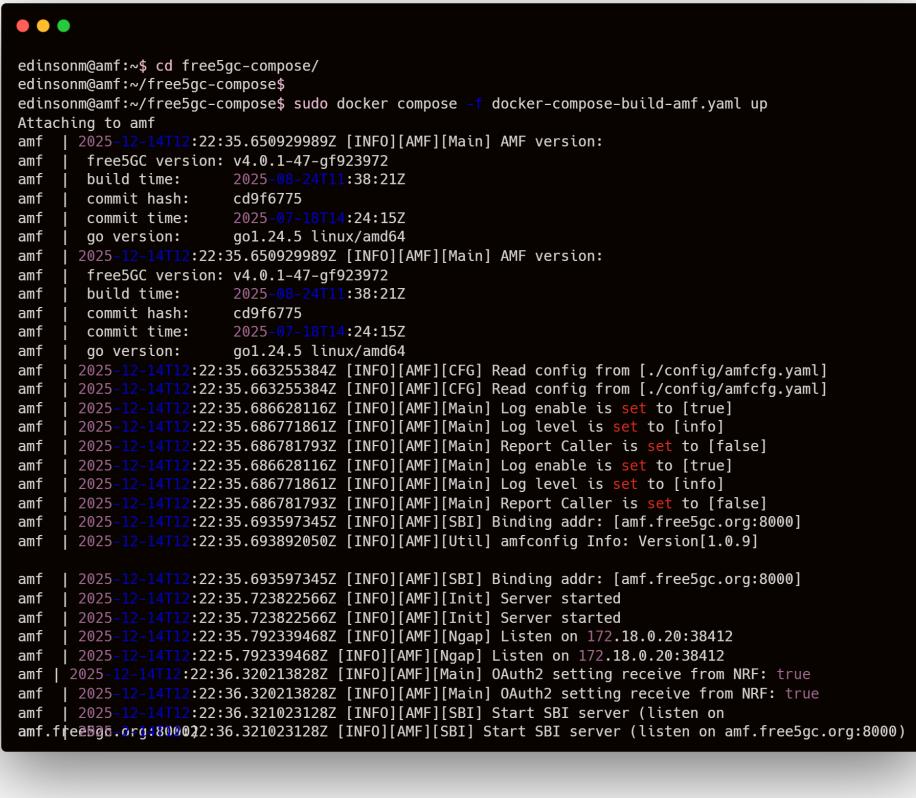
2   free5gc-amf:
3     container_name: amf
4     build:
5       context: ./nf_amf
6       args:
7         DEBUG_TOOLS: "false"
8       command: ./amf -c ./config/amfcfg.yaml
9     expose:
10      - "8000"
11      - "38412"
12     volumes:
13       - ./config/amfcfg.yaml:/free5gc/config/amfcfg.yaml
14       - ./cert:/free5gc/cert
15     environment:
16       GIN_MODE: release
17     ports:
18       - "38412:38412"
19       - "8000:8000"
20     networks:
21       macvlan_net:
22         ipv4_address: 172.18.0.20
23         mac_address: f2:ff:cb:34:07:b5
24         aliases:
25           - amf.free5gc.org
26     extra_hosts:
27       - "nrf.free5gc.org:172.18.0.51"
28       - "udr.free5gc.org:172.18.0.56"
29       - "pcf.free5gc.org:172.18.0.54"
30       - "ausf.free5gc.org:172.18.0.52"
31       - "udm.free5gc.org:172.18.0.55"
32       - "nssf.free5gc.org:172.18.0.53"
33       - "smf.free5gc.org:172.18.0.22"
34       - "upf.free5gc.org:172.18.0.24"
35       - "gnb.free5gc.org:172.18.10.3"
36   networks:
37     macvlan_net:
38       external: true

```

Código 5.12: Configuración del Docker Compose AMF

### 5.2.2.3 Registro del AMF en el NRF

Una vez desplegado el contenedor AMF, se verifica en los logs del NRF que el AMF se haya registrado correctamente en el NRF mediante la interfaz SBI de acuerdo con la arquitectura mostrada en la figura 2.3. Ademas, que empiece a escuchar en el puerto SCTP 38412 y que este listo para recibir conexiones desde el NGRAN, como se muestra en la figura 5.9.



```

edinsonm@amf:~$ cd free5gc-compose/
edinsonm@amf:~/free5gc-compose$
edinsonm@amf:~/free5gc-compose$ sudo docker compose -f docker-compose-build-amf.yaml up
Attaching to amf
amf | 2025-12-14T12:22:35.650929989Z [INFO][AMF][Main] AMF version:
amf |   free5GC version: v4.0.1-47-gf923972
amf |   build time:    2025-08-24T11:38:21Z
amf |   commit hash:   cd9f6775
amf |   commit time:   2025-07-18T14:24:15Z
amf |   go version:    go1.24.5 linux/amd64
amf | 2025-12-14T12:22:35.650929989Z [INFO][AMF][Main] AMF version:
amf |   free5GC version: v4.0.1-47-gf923972
amf |   build time:    2025-08-24T11:38:21Z
amf |   commit hash:   cd9f6775
amf |   commit time:   2025-07-18T14:24:15Z
amf |   go version:    go1.24.5 linux/amd64
amf | 2025-12-14T12:22:35.663255384Z [INFO][CFG] Read config from [./config/amfcfg.yaml]
amf | 2025-12-14T12:22:35.663255384Z [INFO][AMF][CFG] Read config from [./config/amfcfg.yaml]
amf | 2025-12-14T12:22:35.686628116Z [INFO][AMF][Main] Log enable is set to [true]
amf | 2025-12-14T12:22:35.686771861Z [INFO][AMF][Main] Log level is set to [info]
amf | 2025-12-14T12:22:35.686781793Z [INFO][AMF][Main] Report Caller is set to [false]
amf | 2025-12-14T12:22:35.686628116Z [INFO][AMF][Main] Log enable is set to [true]
amf | 2025-12-14T12:22:35.686771861Z [INFO][AMF][Main] Log level is set to [info]
amf | 2025-12-14T12:22:35.686781793Z [INFO][AMF][Main] Report Caller is set to [false]
amf | 2025-12-14T12:22:35.693597345Z [INFO][AMF][SBI] Binding addr: [amf.free5gc.org:8000]
amf | 2025-12-14T12:22:35.693892050Z [INFO][Util] amfconfig Info: Version[1.0.9]

amf | 2025-12-14T12:22:35.693597345Z [INFO][AMF][SBI] Binding addr: [amf.free5gc.org:8000]
amf | 2025-12-14T12:22:35.723822566Z [INFO][AMF][Init] Server started
amf | 2025-12-14T12:22:35.723822566Z [INFO][AMF][Init] Server started
amf | 2025-12-14T12:22:35.792339468Z [INFO][AMF][Ngap] Listen on 172.18.0.20:38412
amf | 2025-12-14T12:22:35.792339468Z [INFO][AMF][Ngap] Listen on 172.18.0.20:38412
amf | 2025-12-14T12:22:36.320213828Z [INFO][AMF][Main] OAuth2 setting receive from NRF: true
amf | 2025-12-14T12:22:36.320213828Z [INFO][AMF][Main] OAuth2 setting receive from NRF: true
amf | 2025-12-14T12:22:36.321023128Z [INFO][AMF][SBI] Start SBI server (listen on
amf.ffeebg6.org:8000)
amf | 2025-12-14T12:22:36.321023128Z [INFO][AMF][SBI] Start SBI server (listen on amf.free5gc.org:8000)

```

Figure 5.9: Registro del AMF en el NRF

En la figura 5.10 se muestran los logs del NRF que confirman que el AMF se ha registrado correctamente y que esta escuchando en el puerto SCTP 38412, listo para recibir conexiones del NGRAN.

```

nrf    | 2025-12-14T12:22:35.855912112Z [INFO][NRF][NFM] Handle NFRegisterRequest
nrf    | 2025-12-14T12:22:35.855912112Z [INFO][NRF][NFM] Handle NFRegisterRequest
mongodba | 2025-12-14T12:22:36.0944+0000 I COMMAND [conn9] command free5gc.urillist command: find { find: "urillist", filter: { nftype: "AMF" }, limit: 1, singleBatch: true, lstd: { id: UUID("25e28981-4d12-4b52-a165-f27a49db29bd") }, $dc: "free5gc" } planSummary: COLLSCAN keysExamined:10 docsExamined:10 cursorExhausted:1 numYields:1 nreturned:1 reslen:1668 locks:{ Global: { acquireCount: { r: 4 } }, Database: { acquireCount: { r: 2 } } }, Collection: { acquireCount: { r: 2 } } protocol:op_msg 123ms
mongodba | 2025-12-14T12:22:36.0944+0000 I COMMAND [conn9] command free5gc.urillist command: find { find: "urillist", filter: { nftype: "AMF" }, limit: 1, singleBatch: true, lstd: { id: UUID("25e28981-4d12-4b52-a165-f27a49db29bd") }, $dc: "free5gc" } planSummary: COLLSCAN keysExamined:8 docsExamined:10 cursorExhausted:1 numYields:1 nreturned:1 reslen:1668 locks:{ Global: { acquireCount: { r: 4 } }, Database: { acquireCount: { r: 2 } } }, Collection: { acquireCount: { r: 2 } } protocol:op_msg 123ms
nrf    | 2025-12-14T12:22:36.1842835502 [INFO][NRF][urillist update
nrf    | 2025-12-14T12:22:36.1842835502 [INFO][NRF][urillist update
nrf    | 2025-12-14T12:22:36.2206421182 [INFO][NRF][NFM] Create NF Profile: 8773b008-7c8e-4476-aef0-5ba67af6c6d2
nrf    | 2025-12-14T12:22:36.2206421182 [INFO][NRF][NFM] Create NF Profile: 8773b008-7c8e-4476-aef0-5ba67af6c6d2
nrf    | 2025-12-14T12:22:36.2413411762 [INFO][NRF][NFM] Use NF certPath: cert/amf.pem
nrf    | 2025-12-14T12:22:36.2413411762 [INFO][NRF][NFM] Use NF certPath: cert/amf.pem
nrf    | 2025-12-14T12:22:36.2998327272 [INFO][NRF][GIN] | 201 |      172.18.0.20 | PUT | /nnrf-nfm/v1/nf-instances/8773b008-7c8e-4476-aef0-5ba67af6c6d2 |
nrf    | 2025-12-14T12:22:36.2998327272 [INFO][NRF][GIN] | 201 |      172.18.0.20 | PUT | /nnrf-nfm/v1/nf-instances/8773b008-7c8e-4476-aef0-5ba67af6c6d2 |

```

Figure 5.10: Verificación del registro del AMF en el NRF

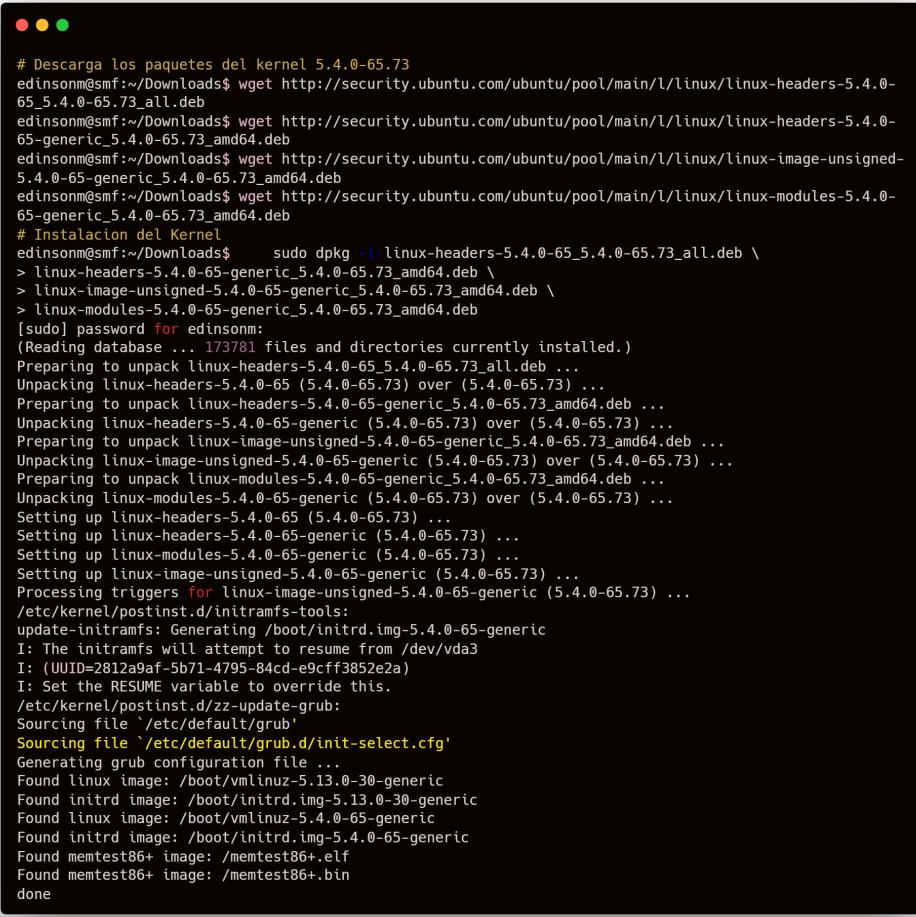
Las configuraciones del AMF, como las direcciones IP y puertos de escucha, se realizan en el archivo de configuración *amfcfg.yaml* ubicado en el directorio *./config/* del repositorio del AMF, las cuales no se muestran en este documento por cuestiones de espacio.

### 5.2.3 Configuración de SMF

A diferencia de los NFs desplegados en el Core-Host y SMF, tanto gNB (UER-ANSIM), SMF como UPF requieren de la version kernel 5.0.0-23-generic o superior a la 5.4 los cuales son compatibles con el modulo de kernel GTP para 5G.

#### 5.2.3.1 Instalacion del modulo GTP

Como primer paso antes de instalar el modulo GTP, se instala la version de kernel requerida como se muestra en la figura 5.11.



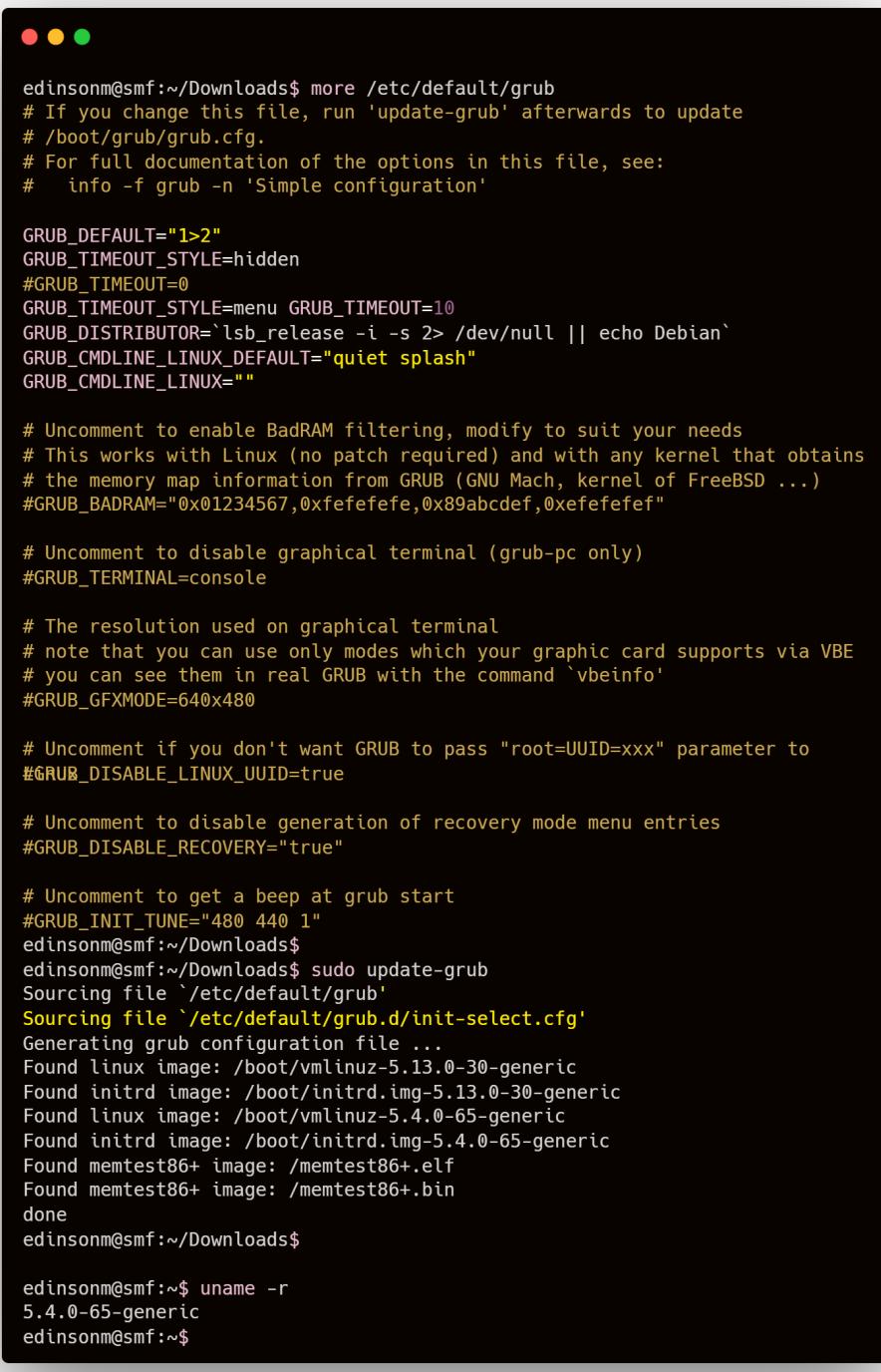
```

# Descarga los paquetes del kernel 5.4.0-65.73
edinsonm@smf:~/Downloads$ wget http://security.ubuntu.com/ubuntu/pool/main/l/linux/linux-headers-5.4.0-65_5.4.0-65.73_all.deb
edinsonm@smf:~/Downloads$ wget http://security.ubuntu.com/ubuntu/pool/main/l/linux/linux-headers-5.4.0-65-generic_5.4.0-65.73_amd64.deb
edinsonm@smf:~/Downloads$ wget http://security.ubuntu.com/ubuntu/pool/main/l/linux/linux-image-unsigned-5.4.0-65-generic_5.4.0-65.73_amd64.deb
edinsonm@smf:~/Downloads$ wget http://security.ubuntu.com/ubuntu/pool/main/l/linux/linux-modules-5.4.0-65-generic_5.4.0-65.73_amd64.deb
# Instalacion del Kernel
edinsonm@smf:~/Downloads$ sudo dpkg -i linux-headers-5.4.0-65_5.4.0-65.73_all.deb \
> linux-headers-5.4.0-65-generic_5.4.0-65.73_amd64.deb \
> linux-image-unsigned-5.4.0-65-generic_5.4.0-65.73_amd64.deb \
> linux-modules-5.4.0-65-generic_5.4.0-65.73_amd64.deb
[sudo] password for edinsonm:
(Reading database ... 173781 files and directories currently installed.)
Preparing to unpack linux-headers-5.4.0-65_5.4.0-65.73_all.deb ...
Unpacking linux-headers-5.4.0-65 (5.4.0-65.73) over (5.4.0-65.73) ...
Preparing to unpack linux-headers-5.4.0-65-generic_5.4.0-65.73_amd64.deb ...
Unpacking linux-headers-5.4.0-65-generic (5.4.0-65.73) over (5.4.0-65.73) ...
Preparing to unpack linux-image-unsigned-5.4.0-65-generic_5.4.0-65.73_amd64.deb ...
Unpacking linux-image-unsigned-5.4.0-65-generic (5.4.0-65.73) over (5.4.0-65.73) ...
Preparing to unpack linux-modules-5.4.0-65-generic_5.4.0-65.73_amd64.deb ...
Unpacking linux-modules-5.4.0-65-generic (5.4.0-65.73) over (5.4.0-65.73) ...
Setting up linux-headers-5.4.0-65 (5.4.0-65.73) ...
Setting up linux-headers-5.4.0-65-generic (5.4.0-65.73) ...
Setting up linux-modules-5.4.0-65-generic (5.4.0-65.73) ...
Setting up linux-image-unsigned-5.4.0-65-generic (5.4.0-65.73) ...
Processing triggers for linux-image-unsigned-5.4.0-65-generic (5.4.0-65.73) ...
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-5.4.0-65-generic
I: The initramfs will attempt to resume from /dev/vda3
I: (UUID=2812a9af-5b71-4795-84cd-e9cff3852e2a)
I: Set the RESUME variable to override this.
/etc/kernel/postinst.d/zz-update-grub:
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.13.0-30-generic
Found initrd image: /boot/initrd.img-5.13.0-30-generic
Found linux image: /boot/vmlinuz-5.4.0-65-generic
Found initrd image: /boot/initrd.img-5.4.0-65-generic
Found memtest86+ image: /memtest86+.elf
Found memtest86+ image: /memtest86+.bin
done

```

Figure 5.11: Instalación de kernel compatible con GTP

Luego de instalar el kernel compatible, se edita y actualiza el archivo */etc/default/grub* para establecer la nueva versión de kernel como predeterminada al iniciar el sistema, como se muestra en la figura 5.12.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three small colored circles (red, yellow, green) representing window control buttons. The terminal session starts with:

```
edinsonm@smf:~/Downloads$ more /etc/default/grub
```

Then it displays the contents of the /etc/default/grub file, which includes various GRUB configuration options like GRUB\_TIMEOUT, GRUB\_CMDLINE\_LINUX\_DEFAULT, and GRUB\_TERMINAL. The file ends with:

```
#GRUB_DISABLE_LINUX_UUID=true
```

Following this, the user runs the command:

```
edinsonm@smf:~/Downloads$ sudo update-grub
```

The terminal then shows the output of the Sourcing file process, followed by a list of found Linux and initrd images:

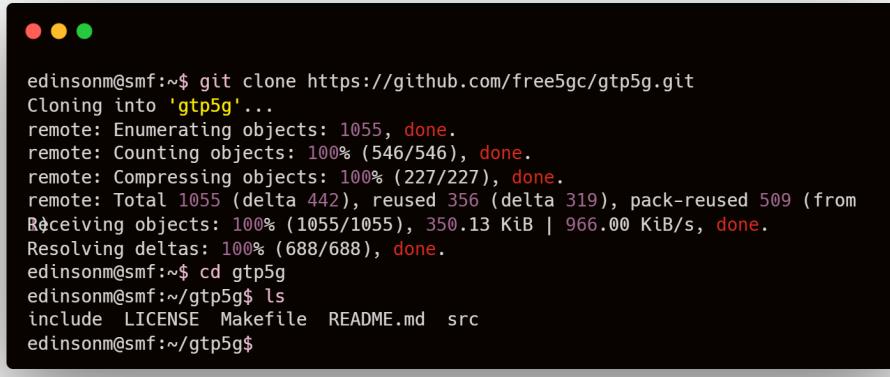
```
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.13.0-30-generic
Found initrd image: /boot/initrd.img-5.13.0-30-generic
Found linux image: /boot/vmlinuz-5.4.0-65-generic
Found initrd image: /boot/initrd.img-5.4.0-65-generic
Found memtest86+ image: /memtest86+.elf
Found memtest86+ image: /memtest86+.bin
done
```

Finally, the user checks the system version:

```
edinsonm@smf:~/Downloads$ uname -r
5.4.0-65-generic
edinsonm@smf:~$
```

Figure 5.12: Edición del archivo y actualización de /etc/default/grub

Una vez reiniciado el sistema con la nueva versión de kernel, se procede con la instalación del módulo GTP. Primero se clona el repositorio oficial desde GitHub como se muestra en la figura 5.13.



```
edinsonm@smf:~$ git clone https://github.com/free5gc/gtp5g.git
Cloning into 'gtp5g'...
remote: Enumerating objects: 1055, done.
remote: Counting objects: 100% (546/546), done.
remote: Compressing objects: 100% (227/227), done.
remote: Total 1055 (delta 442), reused 356 (delta 319), pack-reused 509 (from
Receiving objects: 100% (1055/1055), 350.13 KiB | 966.00 KiB/s, done.
Resolving deltas: 100% (688/688), done.
edinsonm@smf:~$ cd gtp5g
edinsonm@smf:~/gtp5g$ ls
include LICENSE Makefile README.md src
edinsonm@smf:~/gtp5g$
```

Figure 5.13: Clonación del repositorio del módulo GTP

Luego de clonar el repositorio, se limpia, compila e instala el módulo GTP en el kernel como se muestra en la figura 5.14.

```
edinsonm@smf:~/gtp5g$ make clean
make -C /lib/modules/5.4.0-65-generic/build M=/home/edinsonm/gtp5g clean
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-65-generic'
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-65-generic'
edinsonm@smf:~/gtp5g$ make
make -C /lib/modules/5.4.0-65-generic/build M=/home/edinsonm/gtp5g
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-65-generic'
  CC [M] /home/edinsonm/gtp5g/src/gtp5g.o
  CC [M] /home/edinsonm/gtp5g/src/log.o
  CC [M] /home/edinsonm/gtp5g/src/util.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/dev.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/encap.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/hash.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/link.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/net.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/pktinfo.o
  CC [M] /home/edinsonm/gtp5g/src/gtpu/trTCM.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_version.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_pdr.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_far.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_qer.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_ur.r.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_report.o
  CC [M] /home/edinsonm/gtp5g/src/genl/genl_bar.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/api_version.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/pdr.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/far.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/qer.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/ur.r.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/bar.o
  CC [M] /home/edinsonm/gtp5g/src/pfcp/seid.o
  CC [M] /home/edinsonm/gtp5g/src/proc.o
  LD [M] /home/edinsonm/gtp5g/gtp5g.o
Building modules, stage 2.
MODPOST 1 modules
  CC [M] /home/edinsonm/gtp5g/gtp5g.mod.o
  LD [M] /home/edinsonm/gtp5g/gtp5g.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-65-generic'
edinsonm@smf:~/gtp5g$ sudo make install
cp gtp5g.ko /lib/modules/5.4.0-65-generic/kernel/drivers/net
modprobe udp_tunnel
/sbin/depmod -a
modprobe gtp5g
echo "udp_tunnel" > /etc/modules-load.d/gtp5g.conf
echo "gtp5g" >> /etc/modules-load.d/gtp5g.conf
edinsonm@smf:~/gtp5g$
```

Figure 5.14: Compilación e instalación del módulo GTP

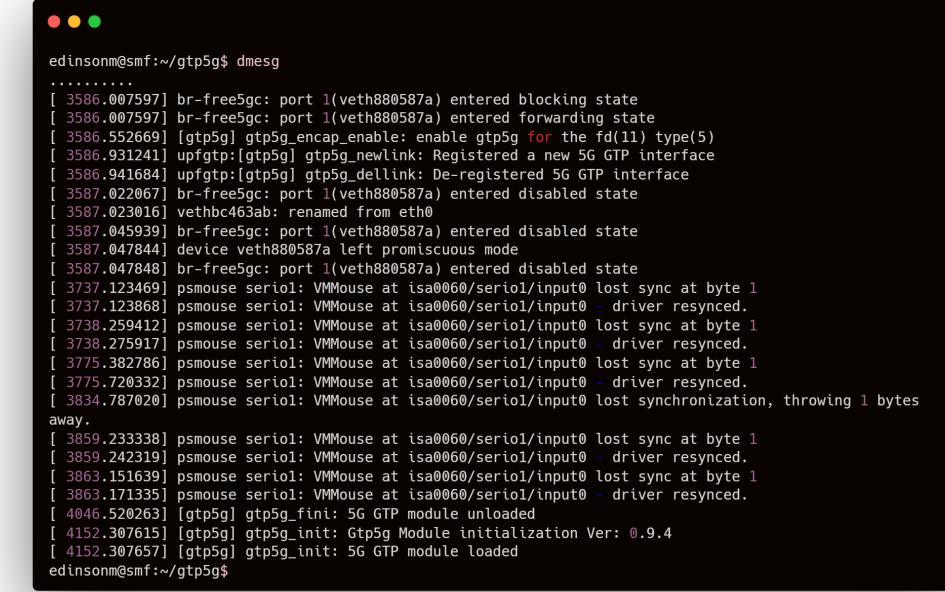
Luego de instalado el modulo GTP, se carga el modulo gtp5g en el kernel y luego se listan los modulos cargados para verificar que el modulo GTP se haya cargado correctamente, como se muestra en la figura 5.15.



```
edinsonm@smf:~/gtp5g$  
edinsonm@smf:~/gtp5g$ sudo modprobe  
gtp5g  
edinsonm@smf:~/gtp5g$ lsmod | grep gtp5g  
gtp5g                  143360  0  
udp_tunnel              16384   1 gtp5g  
edinsonm@smf:~/gtp5g$
```

Figure 5.15: Carga y verificación del módulo GTP

Finalmente, se revisa en los logs del sistema que el modulo GTP se haya cargado correctamente al iniciar el sistema, como se muestra en la figura 5.16.



```

edinsonm@smf:~/gtp5g$ dmesg
.....
[ 3586.007597] br-free5gc: port 1(veth880587a) entered blocking state
[ 3586.007597] br-free5gc: port 1(veth880587a) entered forwarding state
[ 3586.552669] [gtp5g] gtp5g_encap_enable: enable gtp5g for the fd(11) type(5)
[ 3586.931241] upfgtp:[gtp5g] gtp5g_newlink: Registered a new 5G GTP interface
[ 3586.941684] upfgtp:[gtp5g] gtp5g_dellink: De-registered 5G GTP interface
[ 3587.022067] br-free5gc: port 1(veth880587a) entered disabled state
[ 3587.023016] vethbc463ab: renamed from eth0
[ 3587.045939] br-free5gc: port 1(veth880587a) entered disabled state
[ 3587.047844] device veth880587a left promiscuous mode
[ 3587.047848] br-free5gc: port 1(veth880587a) entered disabled state
[ 3737.123469] psmouse seriol: VMMouse at isa0060/seriol/input0 lost sync at byte 1
[ 3737.123868] psmouse seriol: VMMouse at isa0060/seriol/input0 - driver resynced.
[ 3738.259412] psmouse seriol: VMMouse at isa0060/seriol/input0 lost sync at byte 1
[ 3738.275917] psmouse seriol: VMMouse at isa0060/seriol/input0 - driver resynced.
[ 3775.382786] psmouse seriol: VMMouse at isa0060/seriol/input0 lost sync at byte 1
[ 3775.720332] psmouse seriol: VMMouse at isa0060/seriol/input0 - driver resynced.
[ 3834.787020] psmouse seriol: VMMouse at isa0060/seriol/input0 lost synchronization, throwing 1 bytes
away.
[ 3859.233338] psmouse seriol: VMMouse at isa0060/seriol/input0 lost sync at byte 1
[ 3859.242319] psmouse seriol: VMMouse at isa0060/seriol/input0 - driver resynced.
[ 3863.151639] psmouse seriol: VMMouse at isa0060/seriol/input0 lost sync at byte 1
[ 3863.171335] psmouse seriol: VMMouse at isa0060/seriol/input0 - driver resynced.
[ 4046.520263] [gtp5g] gtp5g_fini: 5G GTP module unloaded
[ 4152.307615] [gtp5g] gtp5g_init: Gtp5g Module initialization Ver: 0.9.4
[ 4152.307657] [gtp5g] gtp5g_init: 5G GTP module loaded
edinsonm@smf:~/gtp5g$
```

Figure 5.16: Verificación del módulo GTP en los logs del sistema

### 5.2.3.2 Obtención e instalación del código fuente SMF

Este proceso es similar al realizado para el Core-Host, y el AMF. Para fines de ahorrar espacio, no se mostrará el proceso de clonación y construcción de la imagen Docker del SMF.

### 5.2.3.3 Configuración de Docker Compose

Al igual que AMF, para el despliegue del contenedor SMF, se crea un archivo Docker Compose llamado *docker-compose-build-smf.yaml* con la configuración del SMF mostrada en el siguiente código 5.13.

```

1 services:
2   free5gc-smf:
3     container_name: smf
4     build:
5       context: ./nf_smf
6       args:
7         DEBUG_TOOLS: "false"
8     command: ./smf -c ./config/smfcfg.yaml -u ./config/
9       uerouting.yaml
  expose:
```

```

10      - "8000"
11  volumes:
12    - ./config/smfcfg.yaml:/free5gc/config/smfcfg.yaml
13    - ./config/uerouting.yaml:/free5gc/config/uerouting.
14      yaml
15    - ./cert:/free5gc/cert
16  environment:
17    GIN_MODE: release
18  ports:
19    - "8000:8000"
20  networks:
21    macvlan_net:
22      ipv4_address: 172.18.0.22
23      mac_address: ca:76:58:9c:8e:ec
24      aliases:
25        - smf.free5gc.org
26  extra_hosts:
27    - "nrf.free5gc.org:172.18.0.51"
28    - "udr.free5gc.org:172.18.0.56"
29    - "pcf.free5gc.org:172.18.0.54"
30    - "ausf.free5gc.org:172.18.0.52"
31    - "udm.free5gc.org:172.18.0.55"
32    - "nssf.free5gc.org:172.18.0.53"
33    - "amf.free5gc.org:172.18.0.20"
34    - "upf.free5gc.org:172.18.0.24"
35    - "gnb.free5gc.org:172.18.10.3"
36
37  networks:
38    macvlan_net:
39      external: true

```

Código 5.13: Configuración del Docker Compose SMF

Al igual que en los NFs anteriores, se le especifica las direcciones IP de los demás NFs para que pueda resolver los nombre de dominio de cada NF mediante el archivo `/etc/hosts` del contenedor SMF de acuerdo con el LLD mostrado en la tabla 5.1.

#### 5.2.3.4 Registro del SMF en el NRF y conexión N4 con UPF

Una vez desplegado el contenedor SMF, se verifica que se haya registrado correctamente en el NRF y que se establezca la conexión con el UPF mediante la interfaz N4, orquestado por el NRF según la arquitectura SBI mostrada en la figura 2.3. Como se puede apreciar en la figura 5.17, el SMF se ha registrado correctamente en el NRF y esta listo para gestionar las sesiones de usuario y la conexión N4 con el UPF está establecida correctamente.

Figure 5.17: Registro del SMF en el NRF

Logs logs del establecimiento de la conexión N4 entre el SMF y el UPF se muestran a detalle en la sesión del UPF en la figura 5.19.

## 5.2.4 Configuración de UPF

El proceso de obtención, compilación, instalación de modulo gtp5g y despliegue del UPF es similar al realizado para el Core-Host, AMF y SMF. Para fines de ahorrar espacio, no se mostrará el proceso de clonación y construcción de imagen Docker del UPF.

### 5.2.4.1 Configuración de Docker Compose

Al igual que AMF y SMF, para el despliegue del contenedor UPF, se crea un archivo Docker Compose llamado *docker-compose-build-upf.yaml* con la configuración del UPF mostrado en el siguiente código 5.14.

```

1 version: "3.8"
2 services:
3   free5gc-upf:
4     container_name: upf
5     build:
6       context: ./nf_upf
7       args:
8         DEBUG_TOOLS: "false"
9       command: bash -c "./upf-iptables.sh && ./upf -c ./config/
10        upfcfg.yaml"
11     expose:
12       - "8000"
13       - "38412"
14     volumes:
15       - ./config/upfcfg.yaml:/free5gc/config/upfcfg.yaml
16       - ./config/upf-iptables.sh:/free5gc/upf-iptables.sh
17     cap_add:
18       - NET_ADMIN
19     ports:
20       - "38412:38412"
21       - "8000:8000"
22     networks:
23       macvlan_net:
24         ipv4_address: 172.18.0.24
25         mac_address: be:44:98:a4:e5:c4
26         aliases:
27           - upf.free5gc.org
28     extra_hosts:
29       - "nrf.free5gc.org:172.18.0.51"
30       - "udr.free5gc.org:172.18.0.56"
31       - "pcf.free5gc.org:172.18.0.54"
32       - "ausf.free5gc.org:172.18.0.52"
33       - "udm.free5gc.org:172.18.0.55"
34       - "nssf.free5gc.org:172.18.0.53"
35       - "amf.free5gc.org:172.18.0.20"
```

```
35      - "smf.free5gc.org:172.18.0.22"
36      - "gnb.free5gc.org:172.18.10.3"
37 networks:
38   macvlan_net:
39     external: true
```

Código 5.14: Configuración del Docker Compose UPF

#### 5.2.4.2 Registro del UPF en el NRF

Una vez desplegado el contenedor UPF, en la figura ?? se aprecia que prepara los parametros para la interfaz N3 con gNB y los recursos para el UE, los cuales son *10.60.0.0/16* y *10.61.0.0/16*.



```

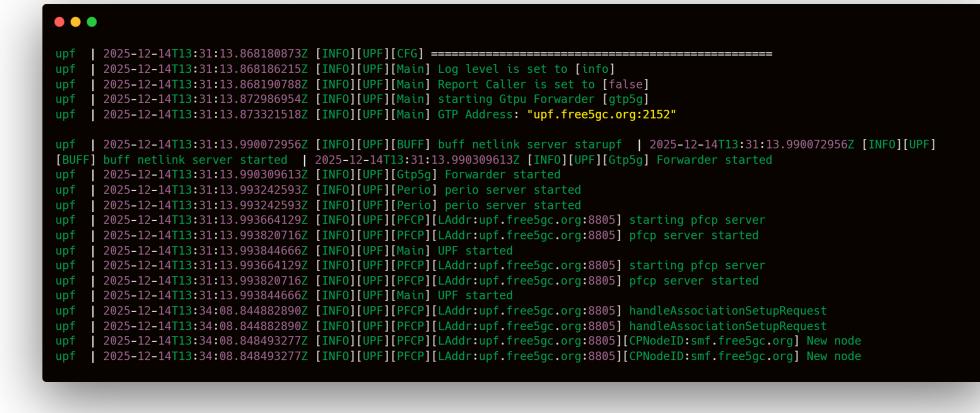
edinsonm@upf:~/free5gc-compose$ sudo docker compose -f docker-compose-build-upf.yaml up
[sudo] password for edinsonm:
WARN[0000] /home/edinsonm/free5gc-compose/docker-compose-build-upf.yaml: the attribute `version` is obsolete, it will be ignored,
please remove it to avoid potential confusion
Attaching to upf
upf | 2025-12-14T13:31:13.843687483Z [INFO][UPF][Main] UPF version:
upf |   free5GC version: v4.0.1-47-gf923972
upf |   build time: 2025-08-26T20:17:34Z
upf |   commit hash: 9740983c
upf |   commit time: 2025-07-18T14:47:46Z
upf |   go version: go1.24.5 linux/amd64
upf | 2025-12-14T13:31:13.857445343Z [INFO][UPF][CFG] Read config from [./config/upfcfg.yaml]
upf | 2025-12-14T13:31:13.843687483Z [INFO][UPF][Main] UPF version:
upf |   free5GC version: v4.0.1-47-gf923972
upf |   build time: 2025-08-26T20:17:34Z
upf |   commit hash: 9740983c
upf |   commit time: 2025-07-18T14:47:46Z
upf |   go version: go1.24.5 linux/amd64
upf | 2025-12-14T13:31:13.857445343Z [INFO][UPF][CFG] Read config from [./config/upfcfg.yaml]
upf | 2025-12-14T13:31:13.868135211Z [INFO][UPF][CFG] -----
upf | 2025-12-14T13:31:13.868173594Z [INFO][UPF][CFG] (*factory.Config)(0xc0001233b0)({})
upf | Version: (string) (len=5) "1.0.3",
upf | Description: (string) (len=31) "UPF initial local configuration",
upf | Pfcpc: (*factory.Pfcpc)(0xc000176e70)({
upf |   Addr: (string) (len=15) "upf.free5gc.org",
upf |   NodeID: (string) (len=15) "upf.free5gc.org",
upf |   -----
upf | 2025-12-14T13:31:13.868135211Z [INFO][UPF][CFG] -----
upf | 2025-12-14T13:31:13.868173594Z [INFO][UPF][CFG] (*factory.Config)(0xc0001233b0)({})
upf | Version: (string) (len=5) "1.0.3",
upf | Description: (string) (len=31) "UPF initial local configuration",
upf | Pfcpc: (*factory.Pfcpc)(0xc000176e70)({
upf |   Addr: (string) (len=15) "upf.free5gc.org",
upf |   NodeID: (string) (len=15) "upf.free5gc.org",
upf |   RetransTimeout: (time.Duration) 1s,
upf |   MaxRetrans: (uint8) 3
upf | },
upf | Gtpu: (*factory.Gtpu)(0xc000176f60)({
upf |   Forwarder: (string) (len=5) "gtp5g",
upf |   IfList: ([]factory.IfInfo) (len=1 cap=1) {
upf |     (factory.IfInfo) {
upf |       Addr: (string) (len=15) "upf.free5gc.org",
upf |       Type: (string) (len=2) "N3",
upf |       Name: (string) "",
upf |       IfName: (string) "",
upf |       MTU: (uint32) 0
upf |     }
upf |   },
upf |   DnnList: ([]factory.DnnList) (len=2 cap=2) {
upf |     (factory.DnnList) {
upf |       Dnn: (string) (len=8) "internet",
upf |       Cld: (string) (len=12) "10.60.0.0/16",
upf |       NatIfName: (string) ""
upf |     },
upf |     (factory.DnnList) {
upf |       Dnn: (string) (len=8) "internet",
upf |       Cld: (string) (len=12) "10.61.0.0/16",
upf |       NatIfName: (string) ""
upf |     }
upf |   },
upf |   Logger: (*factory.Logger)(0xc000226ea0)({
upf |     Enable: (bool) true,
upf |     Level: (string) (len=4) "info",
upf |     ReportCaller: (bool) false
upf |   })
upf | },
upf | 2025-12-14T13:31:13.8681800873Z [INFO][UPF][CFG] -----
upf | 2025-12-14T13:31:13.868186215Z [INFO][UPF][Main] Log level is set to [info]
upf | 2025-12-14T13:31:13.868190788Z [INFO][UPF][Main] Report Caller is set to [false]
upf | 2025-12-14T13:31:13.872986954Z [INFO][UPF][Main] starting Gtpu Forwarder [gtp5g]
upf | 2025-12-14T13:31:13.873321518Z [INFO][UPF][Main] GTP Address: "upf.free5gup"

```

Figure 5.18: UPF registro y preparacion de interfaz N3

### 5.2.4.3 Conección N4 con SMF

En la figura 5.19 se aprecia que el UPF establece la conexión N4 (PFCP) con el SMF y queda listo para gestionar las sesiones de usuario y el tráfico de datos del UE.



A terminal window displaying log messages from the UPF application. The logs show the initialization of various servers and the handling of association setup requests.

```
upf | 2025-12-14T13:31:13.8681800873Z [INFO][UPF][CFG] -----
upf | 2025-12-14T13:31:13.868186215Z [INFO][UPF][Main] Log level is set to [info]
upf | 2025-12-14T13:31:13.868190788Z [INFO][UPF][Main] Report Caller is set to [false]
upf | 2025-12-14T13:31:13.872986954Z [INFO][UPF][Main] starting Gtpu Forwarder [gtp5g]
upf | 2025-12-14T13:31:13.873321518Z [INFO][UPF][Main] GTP Address: "upf.free5gc.org:2152"
[BUFF] buff netlink server started | 2025-12-14T13:31:13.990309613Z [INFO][UPF][Gtp5g] Forwarder started
upf | 2025-12-14T13:31:13.990309613Z [INFO][UPF][Gtp5g] Forwarder started
upf | 2025-12-14T13:31:13.993242593Z [INFO][UPF][Perio] perio server started
upf | 2025-12-14T13:31:13.993242593Z [INFO][UPF][Perio] perio server started
upf | 2025-12-14T13:31:13.993664129Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805] starting pfcp server
upf | 2025-12-14T13:31:13.993820716Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805] pfcp server started
upf | 2025-12-14T13:31:13.993844666Z [INFO][UPF][Main] UPF started
upf | 2025-12-14T13:31:13.993664129Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805] starting pfcp server
upf | 2025-12-14T13:31:13.993820716Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805] pfcp server started
upf | 2025-12-14T13:31:13.993844666Z [INFO][UPF][Main] UPF started
upf | 2025-12-14T13:34:08.844882890Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805] handleAssociationSetupRequest
upf | 2025-12-14T13:34:08.844882890Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805] handleAssociationSetupRequest
upf | 2025-12-14T13:34:08.848493277Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805][CPNodeID:smf.free5gc.org] New node
upf | 2025-12-14T13:34:08.848493277Z [INFO][UPF][PFCP] [Addr:upf.free5gc.org:8805][CPNodeID:smf.free5gc.org] New node
```

Figure 5.19: UPF registro y preparacion de interfaz N3

### 5.2.5 Integración con UERANSIM

## 5.3 Implementación de la red de distribución P4

### 5.3.1 Diseño de los programas P4

### 5.3.2 Tablas y acciones configuradas

### 5.3.3 Inserción INT-MD en tráfico N2 (SCTP)

### 5.3.4 Inserción INT-MD en tráfico N3 (GTP-U)

## 5.4 Servidor de recolección y análisis

### 5.4.1 Implementación del sink INT

### 5.4.2 Procesamiento y parsing de metadatos

### 5.4.3 Modelo de base de datos

### 5.4.4 Visualización en Grafana

## 5.5 Tecnologías empleadas

### 5.5.1 Docker / Docker Compose

### 5.5.2 GNS3 / GNS3 VM

### 5.5.3 Wireshark y herramientas auxiliares

## 5.6 Despliegue del NGRAN con UERANSIM

El NGRAN se ha desplegado utilizando la herramienta UERANSIM, la cual simula tanto el gNB como el UE en modo 5G Standalone (SA). UERANSIM se ha configurado para conectarse al Core 5G SA desplegado previamente, estableciendo el enlace N2 (SCTP) con el AMF y el enlace N3 (GTP-U) con el UPF.

### 5.6.1 Despliegue de UERANSIM

El despliegue de UERANSIM se realiza mediante Docker Compose, creando un archivo llamado *docker-compose-build-ngran.yaml* que contiene la configuración tanto del gNB como del UE. El archivo Docker Compose se muestra en el siguiente código 5.15.

```

1 version: "3.8"
2 services:
3   ueransim:
4     container_name: ueransim
5     build:
6       context: ./ueransim
7       command: ./nr-gnb -c ./config/gnbcfg.yaml
8     expose:
9       - "38412"
10    volumes:
11      - ./config/gnbcfg.yaml:/ueransim/config/gnbcfg.yaml
12      - ./config/uecfg.yaml:/ueransim/config/uecfg.yaml
13    cap_add:
14      - NET_ADMIN
15    devices:
16      - "/dev/net/tun"
17    ports:
18      - "38412:38412"
19    networks:
20      macvlan_net:
21        ipv4_address: 172.18.10.3
22        mac_address: f6:b8:eb:4d:0f:3c
23        aliases:
24          - gnb.free5gc.org
25    extra_hosts:
26      - "amf.free5gc.org:172.18.0.20"
27      - "upf.free5gc.org:172.18.0.24"
28  networks:
29    macvlan_net:
30      external: true

```

Código 5.15: Configuración del Docker Compose UERANSIM

### 5.6.2 Configuración del gNB

La configuración del gNB se realiza mediante el archivo *gnb.yaml*, donde se especifican los parámetros necesarios para la conexión con el Core 5G SA, como la dirección IP del AMF, el PLMN ID, y los parámetros de la interfaz N3. Un ejemplo de configuración del gNB se muestra en la figura 5.20.

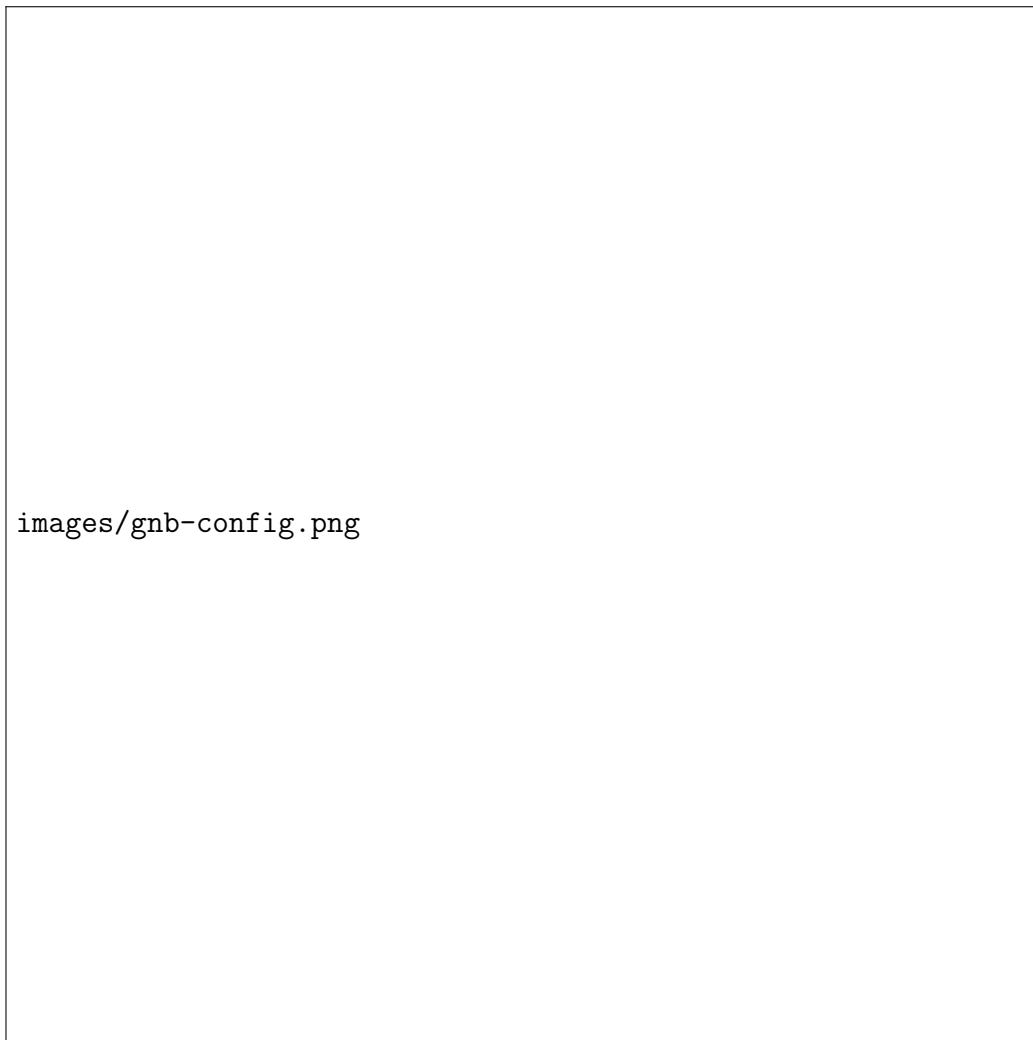


Figure 5.20: Configuración del gNB en UERANSIM

### 5.6.3 Configuración del UE

La configuración del UE se realiza mediante el archivo *ue.yaml*, donde se especifican los parámetros necesarios para la conexión con el gNB, como el IMSI, la clave de seguridad, y los parámetros de la interfaz N1. Un ejemplo de configuración del UE se muestra en la figura 5.21.



Figure 5.21: Configuración del UE en UERANSIM

---

## 5.7 código switch int source

```
1 #include <core.p4>
2 #include <v1model.p4>
3 #define MAX_HOP_COUNT 8
4 #define INT_INSTRUCTIONS_MASK 0x00FF
5 #define TYPE_INT 0xFA
6 #define PROTOCOL_IPV4 0x0800
7 #define PROTOCOL_SCTP 132
```

```
8 #define NGAP_PORT 38412
9 #define PPID_NGAP 60
10
11 /***** HEADER DEFINITIONS *****/
12 header ethernet_t {
13     bit<48> dstAddr;
14     bit<48> srcAddr;
15     bit<16> etherType;
16 }
17
18 header ipv4_t {
19     bit<4> version;
20     bit<4> ihl;
21     bit<8> diffserv;
22     bit<16> totalLen;
23     bit<16> identification;
24     bit<3> flags;
25     bit<13> fragOffset;
26     bit<8> ttl;
27     bit<8> protocol;
28     bit<16> hdrChecksum;
29     bit<32> srcAddr;
30     bit<32> dstAddr;
31 }
32
33 header sctp_common_t {
34     bit<16> src_port;
35     bit<16> dst_port;
36     bit<32> verification_tag;
37     bit<32> checksum;
38 }
39
40 header sctp_chunk_t {
41     bit<8> type;
42     bit<8> flags;
43     bit<16> length;
44 }
45
46 header sctp_data_chunk_t {
47     bit<32> tsn;
48     bit<16> stream_id;
49     bit<16> stream_seq;
50     bit<32> payload_protocol_id;
51 }
52
53 header int_shim_t {
54     bit<8> type;
55     bit<8> reserved;
56     bit<16> length;
```

```

57     bit<16> next_proto;
58 }
59
60 header int_header_t {
61     bit<4> version;
62     bit<2> d;
63     bit<2> q;
64     bit<5> m;
65     bit<3> reserved1;
66     bit<8> hop_ml;
67     bit<16> instruction;
68     bit<8> reserved2;
69     bit<8> remaining_hop_cnt;
70 }
71
72 header int_md_t {
73     bit<32> switch_id;
74     bit<16> ingress_port_id;
75     bit<16> egress_port_id;
76     bit<32> hop_latency;
77     bit<32> queue_occupancy;
78     bit<32> ingress_timestamp;
79     bit<32> egress_timestamp;
80     bit<8> congestion_notification;
81     bit<8> reserved1;
82     bit<16> reserved2;
83 }
84
85 /***** METADATA STRUCTURES *****/
86 struct metadata {
87     bit<32> switch_id;
88     bit<9> ingress_port;
89     bit<32> ingress_ts;
90     bit<32> egress_ts;
91     bit<8> is_int_packet;
92     bit<9> clone_port;
93     bit<32> queue_occupancy;
94     bit<8> should_insert_int;
95     bit<16> sctp_src_port;
96     bit<16> sctp_dst_port;
97     bit<8> is_ngap_traffic;
98 }
99
100 struct headers {
101     ethernet_t ethernet;
102     int_shim_t int_shim;
103     int_header_t int_header;
104     int_md_t int_md;
105     ipv4_t      ipv4;

```

```

106     sctp_common_t sctp;
107     sctp_chunk_t sctp_chunk;
108     sctp_data_chunk_t sctp_data_chunk;
109 }
110
111 /***** PARSER *****/
112 parser MyParser(packet_in packet,
113                  out headers hdr,
114                  inout metadata meta,
115                  inout standard_metadata_t standard_metadata)
116 {
117     state start {
118         packet.extract(hdr.ethernet);
119         meta.ingress_port = standard_metadata.ingress_port;
120         meta.ingress_ts = (bit<32>)standard_metadata.
121             ingress_global_timestamp;
122         meta.is_int_packet = 0;
123         meta.is_ngap_traffic = 0;
124         meta.should_insert_int = 0;
125         transition select(hdr.ethernet.etherType) {
126             0x0800: parse_ipv4;
127             default: accept;
128         }
129     }
130
131     state parse_ipv4 {
132         packet.extract(hdr.ipv4);
133         transition select(hdr.ipv4.protocol) {
134             PROTOCOL_SCTP: parse_sctp_common;
135             default: accept;
136         }
137     }
138     state parse_sctp_common {
139         packet.extract(hdr.sctp);
140         meta.sctp_src_port = hdr.sctp.src_port;
141         meta.sctp_dst_port = hdr.sctp.dst_port;
142
143         // Check if this is NG-RAN to AMF traffic
144         if (hdr.sctp.dst_port == NGAP_PORT) {
145             meta.is_ngap_traffic = 1;
146         }
147         transition parse_sctp_chunk;
148     }
149
150     state parse_sctp_chunk {
151         packet.extract(hdr.sctp_chunk);
152         transition select(hdr.sctp_chunk.type) {

```

```

153         0x00: parse_sctp_data_chunk;
154         default: accept;
155     }
156 }
157
158 state parse_sctp_data_chunk {
159     packet.extract(hdr.sctp_data_chunk);
160     transition accept;
161 }
162 }
163
164 /***** INGRESS PIPELINE *****/
165 control MyIngress(inout headers hdr,
166                     inout metadata meta,
167                     inout standard_metadata_t standard_metadata
168                     ) {
169
170     action int_source_action() {
171         // Mark that we should insert INT
172         meta.should_insert_int = 1;
173     }
174
175     action do_insert_int() {
176         // Store original Ethernet type
177         bit<16> original_ether_type = hdr.ethernet.etherType;
178
179         // Change Ethernet type to INT
180         hdr.ethernet.etherType = TYPE_INT;
181
182         // Set INT shim header
183         hdr.int_shim.setValid();
184         hdr.int_shim.type = 1;
185         hdr.int_shim.reserved = 0;
186         hdr.int_shim.length = 44;
187         hdr.int_shim.next_proto = original_ether_type;
188
189         // Set INT header
190         hdr.int_header.setValid();
191         hdr.int_header.version = 1;
192         hdr.int_header.d = 0;
193         hdr.int_header.q = 0;
194         hdr.int_header.m = 1;
195         hdr.int_header.reserved1 = 0;
196         hdr.int_header.hop_ml = 1;
197         hdr.int_header.instruction = INT_INSTRUCTIONS_MASK;
198         hdr.int_header.reserved2 = 0;
199         hdr.int_header.remaining_hop_cnt = MAX_HOP_COUNT - 1;
200
201         // Add metadata block

```

```

201     hdr.int_md.setValid();
202     hdr.int_md.switch_id = meta.switch_id;
203     hdr.int_md.ingress_port_id = (bit<16>)meta.
204         ingress_port;
205     hdr.int_md.ingress_timestamp = meta.ingress_ts;
206     hdr.int_md.egress_port_id = 0;
207     hdr.int_md.hop_latency = 0;
208     hdr.int_md.egress_timestamp = 0;
209     hdr.int_md.queue_occupancy = 0;
210     hdr.int_md.congestion_notification = 0;
211     hdr.int_md.reserved1 = 0;
212     hdr.int_md.reserved2 = 0;
213 }
214
215     action forward_action(bit<9> egress_port) {
216         standard_metadata.egress_spec = egress_port;
217     }
218
219 // Table for INT insertion - only for NG-RAN to AMF
220 // traffic
221 table int_source_table {
222     key = {
223         meta.is_ngap_traffic: exact;
224     }
225     actions = {
226         int_source_action;
227         NoAction;
228     }
229     size = 1024;
230     default_action = NoAction();
231 }
232
233 table ipv4_lpm {
234     key = {
235         hdr.ipv4.dstAddr: lpm;
236     }
237     actions = {
238         forward_action;
239         NoAction;
240     }
241     size = 1024;
242     default_action = NoAction();
243 }
244
245 apply {
246     meta.switch_id = 1;
247
248     if (hdr.ethernet.isValid() && hdr.ipv4.isValid()) {
249         // Apply INT insertion for NG-RAN to AMF traffic

```

```

248         if (hdr.ipv4.protocol == PROTOCOL_SCTP && hdr.
249             sctp.isValid()) {
250                 int_source_table.apply();
251             }
252
253             // Apply forwarding
254             ipv4_lpm.apply();
255
256             // Insert INT headers if marked
257             if (meta.should_insert_int == 1) {
258                 do_insert_int();
259             }
260         }
261     }
262
263 /***** EGRESS PIPELINE *****/
264 control MyEgress(inout headers hdr,
265                     inout metadata meta,
266                     inout standard_metadata_t standard_metadata)
267 {
268
269     action update_int_metadata() {
270         if (hdr.int_md.isValid()) {
271             hdr.int_md.egress_port_id = (bit<16>)
272                 standard_metadata.egress_port;
273             hdr.int_md.egress_timestamp = (bit<32>)
274                 standard_metadata.egress_global_timestamp;
275             hdr.int_md.hop_latency = hdr.int_md.
276                 egress_timestamp - hdr.int_md.
277                 ingress_timestamp;
278             hdr.int_md.queue_occupancy = (bit<32>)
279                 standard_metadata.enq_qdepth;
280         }
281     }
282
283     apply {
284         if (hdr.int_shim.isValid()) {
285             update_int_metadata();
286         }
287     }
288 }
289
290 /***** CHECKSUM VERIFY/COMPUTE *****/
291 control MyVerifyChecksum(inout headers hdr, inout metadata
292     meta) {
293     apply { }
294 }
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2478
2479
2479
2480
2481
2482
2483
2484
2485
24
```

```

289 control MyComputeChecksum(inout headers hdr, inout metadata
290 meta) {
291     apply { }
292 }
293 /***** DEPARSER *****/
294 control MyDeparser(packet_out packet, in headers hdr) {
295     apply {
296         // Always emit Ethernet
297         packet.emit(hdr.ethernet);
298
299         // Emit INT headers if valid - deparser will handle
300         // validity automatically
301         packet.emit(hdr.int_shim);
302         packet.emit(hdr.int_header);
303         packet.emit(hdr.int_md);
304
305         // Emit inner packet headers
306         packet.emit(hdr.ipv4);
307         packet.emit(hdr.sctp);
308         packet.emit(hdr.sctp_chunk);
309         packet.emit(hdr.sctp_data_chunk);
310     }
311 }
312 /***** SWITCH *****/
313 V1Switch(
314     MyParser(),
315     MyVerifyChecksum(),
316     MyIngress(),
317     MyEgress(),
318     MyComputeChecksum(),
319     MyDeparser()
320 ) main;

```

Código 5.16: Código P4 del switch INT Source

## 5.8 Implicación del código switch INT source

El programa P4 del switch INT source implementa la inserción de metadatos de telemetría en banda (INT-MD) específicamente para el tráfico NGAP transportado sobre SCTP en el enlace N2 entre el NG-RAN y el AMF. El parser del programa analiza secuencialmente las cabeceras Ethernet, IPv4, SCTP y los chunks SCTP, extrayendo información crítica como los puertos de origen y destino del protocolo SCTP. Cuando el parser detecta que

el puerto de destino es 38412 (puerto estándar de NGAP), activa la bandera `is_ngap_traffic` en los metadatos internos del switch. Esta identificación selectiva es fundamental para aplicar la inserción INT únicamente al tráfico de señalización 5G, evitando la sobrecarga innecesaria en otros flujos de datos. El pipeline de ingreso consulta la tabla `int_source_table` que, al encontrar una coincidencia exacta con tráfico NGAP, marca el paquete para inserción INT mediante la acción `int_source_action`. Posteriormente, la función `do_insert_int()` modifica el EtherType original a 0xFA (indicando presencia de INT), inserta la cabecera INT shim que preserva el protocolo original en el campo `next_proto`, y configura la cabecera INT con parámetros como el contador de saltos restantes (`remaining_hop_cnt`) inicializado a 7 (`MAX_HOP_COUNT - 1`) y la máscara de instrucciones que indica qué metadatos deben recolectarse en cada salto.

La inserción del primer bloque de metadatos INT-MD se realiza creando una cabecera `int_md_t` que contiene información del switch fuente como el identificador único del switch (`switch_id`), el puerto de entrada (`ingress_port_id`) y la marca temporal de ingreso (`ingress_timestamp`) capturada desde el metadato estándar `ingress_global_timestamp`. Los campos relacionados con el egreso como `egress_port_id`, `hop_latency` y `queue_occupancy` se inicializan en cero durante la fase de ingreso, siendo actualizados posteriormente en el pipeline de egreso mediante la función `update_int_metadata()`. Esta actualización en egreso calcula la latencia del salto restando la marca temporal de ingreso de la marca temporal de egreso (`egress_global_timestamp`), y registra la ocupación de cola mediante el metadato `enq_qdepth`, proporcionando visibilidad en tiempo real del comportamiento del switch. El deparser ensambla el paquete modificado emitiendo secuencialmente todas las cabeceras válidas: primero Ethernet con el nuevo EtherType INT, seguido de las cabeceras INT shim, INT header e INT metadata, y finalmente las cabeceras del paquete original (IPv4, SCTP y sus chunks), preservando así la integridad del payload mientras se añaden las capas de telemetría. Esta arquitectura permite que los switches downstream (transit nodes) procesen los metadatos INT existentes y agreguen sus propios bloques de metadatos, construyendo una traza completa del recorrido del paquete NGAP a través de la red de transporte 5G.



# Chapter 6

## Resultados Experimentales

### 6.1 Metodología de evaluación

#### 6.1.1 Escenarios de prueba

#### 6.1.2 Tráfico analizado

#### 6.1.3 Métricas recolectadas

### 6.2 Validación funcional

#### 6.2.1 INT en tráfico N2

#### 6.2.2 INT en tráfico N3

#### 6.2.3 Recepción de metadatos en el servidor

### 6.3 Resultados cuantitativos

#### 6.3.1 Latencia hop-by-hop

#### 6.3.2 Carga adicional introducida por INT

#### 6.3.3 Impacto en el plano de usuario

### 6.4 Resultados cualitativos

#### 6.4.1 Dashboards obtenidos

#### 6.4.2 Interpretación de tendencias

#### 6.4.3 Problemas encontrados

# **Chapter 7**

## **Discusión**

### **7.1 Análisis crítico de los resultados**

#### **7.1.1 Comparación con el estado del arte**

#### **7.1.2 Validación de la hipótesis planteada**

### **7.2 Beneficios del uso de P4 en redes 5G**

### **7.3 Desafíos de escalabilidad**

### **7.4 Consideraciones de seguridad para INT**

### **7.5 Limitaciones del trabajo realizado**

# **Chapter 8**

## **Conclusiones y Trabajo Futuro**

### **8.1 Conclusiones principales**

### **8.2 Contribuciones del trabajo**

### **8.3 Limitaciones del estudio**

### **8.4 Líneas futuras de investigación**

#### **8.4.1 INT en 6G**

#### **8.4.2 UPF programable con P4**

#### **8.4.3 IA para análisis de telemetría**

# Chapter A

## Apéndices

A.1 Código P4

A.2 Topologías de red

A.3 Configuraciones del core 5G

A.4 Capturas de tráfico

A.5 Dashboards de Grafana

A.6 Scripts y herramientas auxiliares

# Bibliography

- [1] 3GPP. 3gpp ts 23.501: System architecture for the 5g system (5gs). Technical report, 2021.
- [2] 3GPP. 3gpp ts 38.300: Nr; overall description of the 5g nr physical layer. Technical report, 2023.
- [3] Free5GC Community. Free5gc: Open source 5g core network, 2021.
- [4] Ulises O'Neill, Amelia Mackey, and Victor C.M. Leung. *5G Core Networks: A Comprehensive Guide to 5GC Architecture and Deployment*. O'Reilly Media, 1st edition, 2021.