

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

Dedicatoria	6
Acrónimos	7
1 Introducción	11
1.1 Contexto general de las redes 5G y beyond	11
1.2 Motivación del estudio	11
1.3 Problemática en la visibilidad y monitoreo de redes 5G y beyond	12
1.4 Hipótesis y preguntas de investigación	12
1.5 Objetivos generales y específicos	13
1.6 Metodología de trabajo	13
1.7 Estructura del documento	14
1.8 Resumen de capítulos	14
2 Fundamentos de las Redes 5G y Beyond	16
2.1 Evolución de tecnologías móviles	16
2.1.1 De 1G a 6G: Hitos clave	16
2.2 Arquitectura NR en 5G	17
2.3 Arquitectura 5G Non Standalone (NSA)	18
2.4 Arquitectura 5G Standalone (SA)	20
2.4.1 Visión general de la arquitectura 5G SA	20
2.4.2 Arquitectura basada en servicios (SBA)	21
2.4.3 Interfaces HTTP REST	22
2.4.4 Componentes de 5G Core (5GC)	23
2.4.5 Componentes principales de 5GC	23
2.4.6 Pila de protocolos del plano de control y plano de usuario	25
2.4.6.1 Plano de Control	25
2.4.6.2 Plano de Usuario	26
2.4.7 Interfaces clave en 5GC	26
2.4.8 Tendencias emergentes en redes beyond 5G	27

3 Telemetría y QoS en 5G	29
3.1 Desafíos de Telemetría en Redes 5G	30
3.1.1 Limitaciones de mecanismos tradicionales (SNMP, Net-Flow, sFlow)	30
3.1.2 Requisitos de visibilidad en URLLC, eMBB y mMTC .	30
3.2 In-Band Network Telemetry (INT)	30
3.2.1 Concepto y motivación	30
3.2.2 Arquitectura INT: source, transit y sink	30
3.2.3 Metadatos INT-MD y su estructura	30
3.2.4 INT vs telemetría out-of-band	30
3.3 Telemetría aplicada en 5G	30
3.3.1 Relevancia de INT en el plano de control N2	30
3.3.2 Relevancia de INT en el plano de usuario N3	30
3.3.3 Métricas clave para tráfico GTP-U	30
3.3.4 Desafíos y limitaciones en redes móviles	30
3.4 Calidad de Servicio en 5G	30
3.5 QoS basado en 5G Flujo	30
3.5.1 Definición de flujo en 5G	30
3.5.2 Identificación y clasificación de flujos	30
3.5.3 Mecanismos de gestión de QoS por flujo	30
3.6 Señalización de QoS en 5G	30
3.7 Características de QoS en 5G	30
4 Fundamentos de P4 y Programación del Plano de Datos	31
4.1 Introducción a P4	31
4.1.1 Historia y evolución	31
4.1.2 Modelo de arquitectura PISA	31
4.1.3 P4_14 vs P4_16	31
4.2 Herramientas y ecosistema P4	31
4.2.1 BMv2 como switch software	31
4.2.2 P4C: compilador	31
4.2.3 P4Runtime: control del data plane	31
4.3 Implementación de INT en P4	31
4.3.1 Definición de cabeceras INT	31
4.3.2 Parsing y deparsing en BMv2	31
4.3.3 Inyección hop-by-hop de metadatos	31
4.3.4 Limitaciones del pipeline P4	31
5 Entorno de Desarrollo Implementado	32
5.1 Arquitectura general del sistema	32
5.1.1 Diseño de la red 5G SA	33

5.1.1.1	Diseño de alto nivel (HLD)	34
5.1.1.2	Diseño de bajo nivel (LLD)	34
5.2	Despliegue del Core 5G SA	35
5.2.1	Configuracion de Core-Host	35
5.2.1.1	Obtención del código fuente 5GC	35
5.2.1.2	Configuracion de Docker Compose	37
5.2.1.2.1	DB	37
5.2.1.2.2	NRF	38
5.2.1.2.3	AUSF	39
5.2.1.2.4	NSSF	39
5.2.1.2.5	PCF	40
5.2.1.2.6	UDM	40
5.2.1.2.7	UDR	41
5.2.1.2.8	Despliegue del CHF	42
5.2.1.2.9	Despliegue del NEF	43
5.2.1.2.10	Despliegue del WebUI	43
5.2.1.2.11	Configuracion de Red y Volumenes .	44
5.2.1.3	Construccion y despliegue	44
5.2.2	Configuración de AMF	46
5.2.2.1	Construcción de la imagen Docker del AMF .	47
5.2.2.2	Despliegue del contenedor AMF	47
5.2.2.3	Registro del AMF en el NRF	48
5.2.3	Configuración de SMF	50
5.2.3.1	Instalacion del modulo GTP	50
5.2.3.2	Obtención e instalación del código fuente SMF	56
5.2.3.3	Configuración de Docker Compose	56
5.2.3.4	Registro del SMF en el NRF y conexión N4 con UPF	57
5.2.4	Configuración de UPF	59
5.2.4.1	Configuración de Docker Compose	59
5.2.4.2	Registro del UPF en el NRF	60
5.2.4.3	Conección N4 con SMF	61
5.2.5	Integración con UERANSIM	63
5.3	Implementación de la red de distribución P4	63
5.3.1	Diseño de los programas P4	63
5.3.2	Tablas y acciones configuradas	63
5.3.3	Inserción INT-MD en tráfico N2 (SCTP)	63
5.3.4	Inserción INT-MD en tráfico N3 (GTP-U)	63
5.4	Servidor de recolección y análisis	63
5.4.1	Implementación del sink INT	63
5.4.2	Procesamiento y parsing de metadatos	63

5.4.3	Modelo de base de datos	63
5.4.4	Visualización en Grafana	63
5.5	Tecnologías empleadas	63
5.5.1	Docker / Docker Compose	63
5.5.2	GNS3 / GNS3 VM	63
5.5.3	Wireshark y herramientas auxiliares	63
6	Resultados Experimentales	64
6.1	Metodología de evaluación	65
6.1.1	Escenarios de prueba	65
6.1.2	Tráfico analizado	65
6.1.3	Métricas recolectadas	65
6.2	Validación funcional	65
6.2.1	INT en tráfico N2	65
6.2.2	INT en tráfico N3	65
6.2.3	Recepción de metadatos en el servidor	65
6.3	Resultados cuantitativos	65
6.3.1	Latencia hop-by-hop	65
6.3.2	Carga adicional introducida por INT	65
6.3.3	Impacto en el plano de usuario	65
6.4	Resultados cualitativos	65
6.4.1	Dashboards obtenidos	65
6.4.2	Interpretación de tendencias	65
6.4.3	Problemas encontrados	65
7	Discusión	66
7.1	Análisis crítico de los resultados	66
7.1.1	Comparación con el estado del arte	66
7.1.2	Validación de la hipótesis planteada	66
7.2	Beneficios del uso de P4 en redes 5G	66
7.3	Desafíos de escalabilidad	66
7.4	Consideraciones de seguridad para INT	66
7.5	Limitaciones del trabajo realizado	66
8	Conclusiones y Trabajo Futuro	67
8.1	Conclusiones principales	67
8.2	Contribuciones del trabajo	67
8.3	Limitaciones del estudio	67
8.4	Líneas futuras de investigación	67
8.4.1	INT en 6G	67
8.4.2	UPF programable con P4	67

CONTENTS	5
----------	---

8.4.3 IA para análisis de telemetría	67
--	----

A Apéndices	68
--------------------	-----------

A.1 Código P4	68
A.2 Topologías de red	68
A.3 Configuraciones del core 5G	68
A.4 Capturas de tráfico	68
A.5 Dashboards de Grafana	68
A.6 Scripts y herramientas auxiliares	68

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].	18
2.2	Arquitectura 5GC basada en interfaces basadas en servicios [5].	21
2.3	Arquitectura 5GC con interfaces punto a punto. [5].	22
2.4	Pila de protocolos del plano de control en 5GC [1].	25
2.5	Pila de protocolos del plano de usuario en 5GC. [1, 2].	26
5.1	Arquitectura general	33
5.2	Topología de red implementada en GNS3	34
5.3	Clonación del repositorio de free5gc	35
5.4	Clonación de base de NFs adicionales	36
5.5	Compilación de NFs adicionales	37
5.6	Construcción de imágenes en Core-Host	45
5.7	Despliegue de contenedores en Core-Host	46
5.8	Construcción de la imagen Docker del AMF	47
5.9	Registro del AMF en el NRF	49
5.10	Verificación del registro del AMF en el NRF	50
5.11	Instalación de kernel compatible con GTP	51
5.12	Edición del archivo y actualización de /etc/default/grub	52
5.13	Clonación del repositorio del módulo GTP	53
5.14	Compilación e instalación del módulo GTP	54
5.15	Carga y verificación del módulo GTP	55
5.16	Verificación del módulo GTP en los logs del sistema	56
5.17	Registro del SMF en el NRF	58
5.18	UPF registro y preparacion de interfaz N3	61
5.19	UPF registro y preparacion de interfaz N3	62

List of Tables

5.1 Diseño de bajo nivel (LLD)	34
--	----

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.

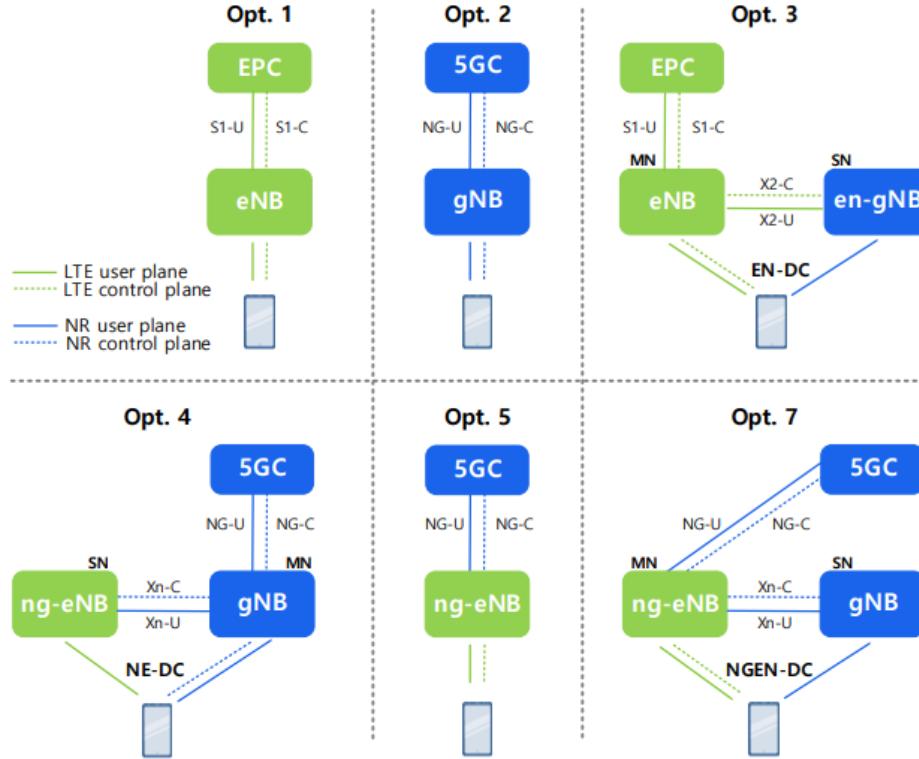


Figure 2.1: Opciones de implementación de NR en 5G: NSA y SA [1].

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.

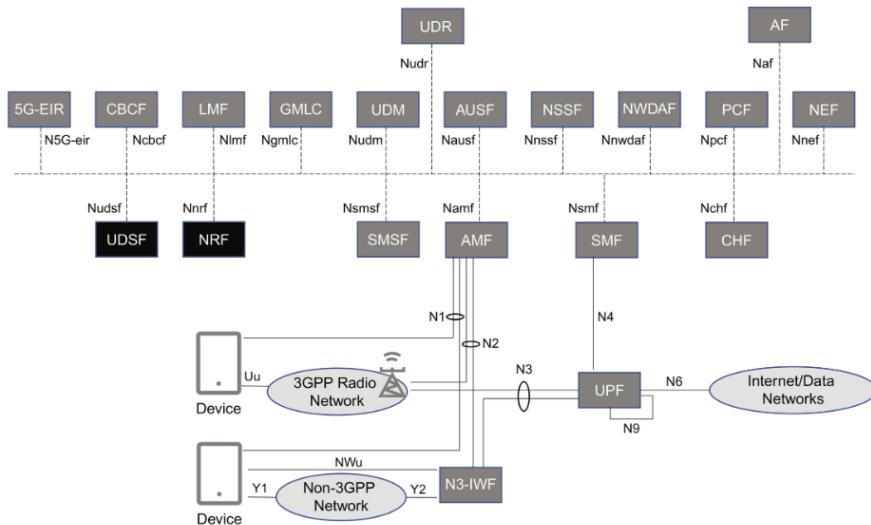


Figure 2.2: Arquitectura 5GC basada en interfaces basadas en servicios [5].

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.

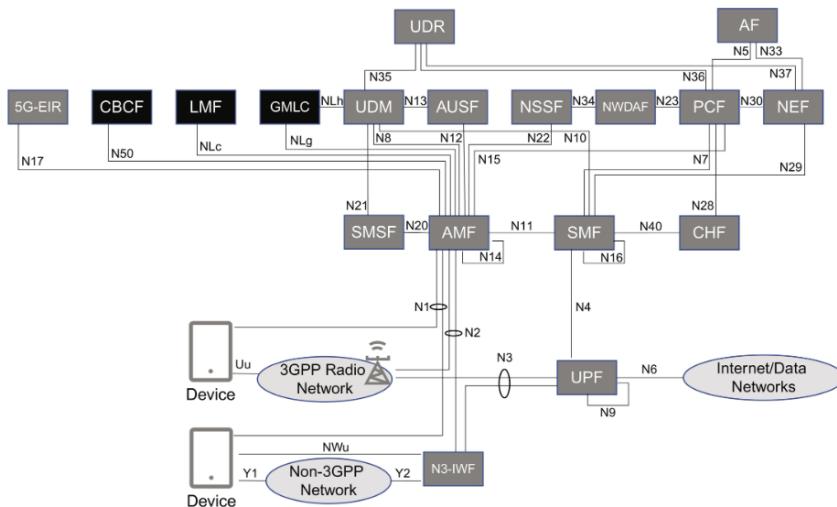


Figure 2.3: Arquitectura 5GC con interfaces punto a punto. [5].

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.

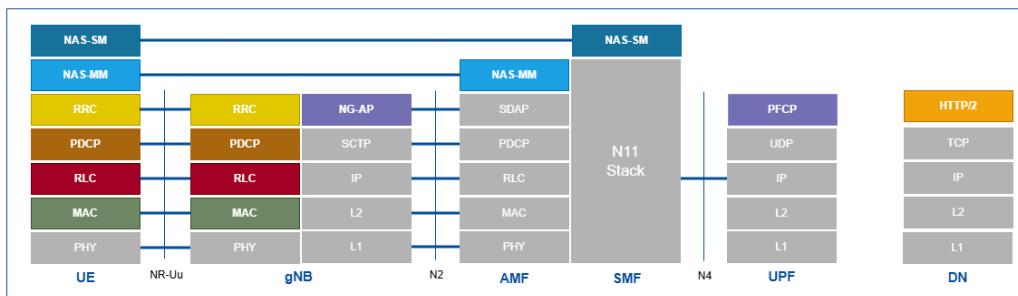


Figure 2.4: Pila de protocolos del plano de control en 5GC [1].

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.

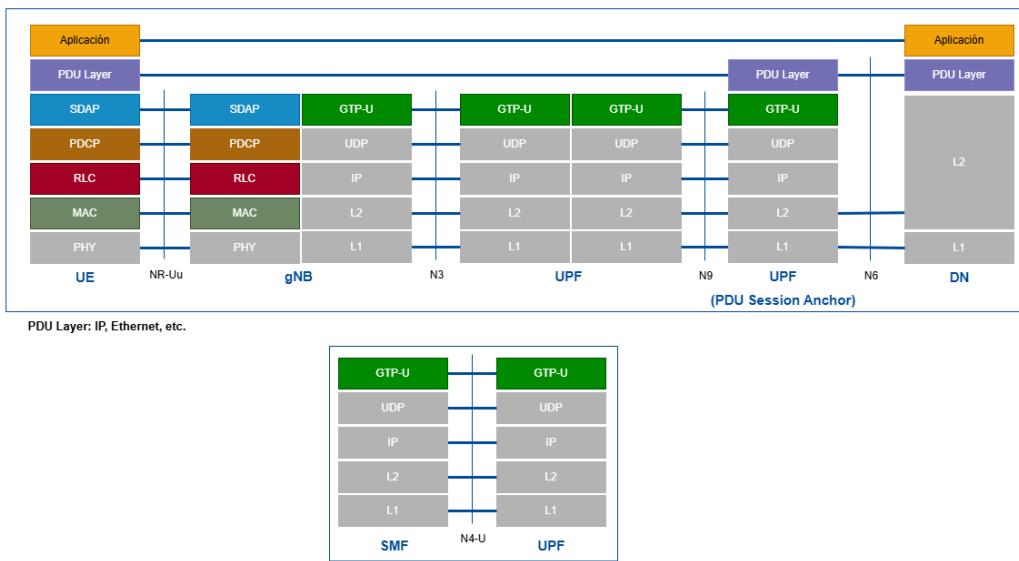


Figure 2.5: Pila de protocolos del plano de usuario en 5GC. [1, 2].

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 [4], 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.

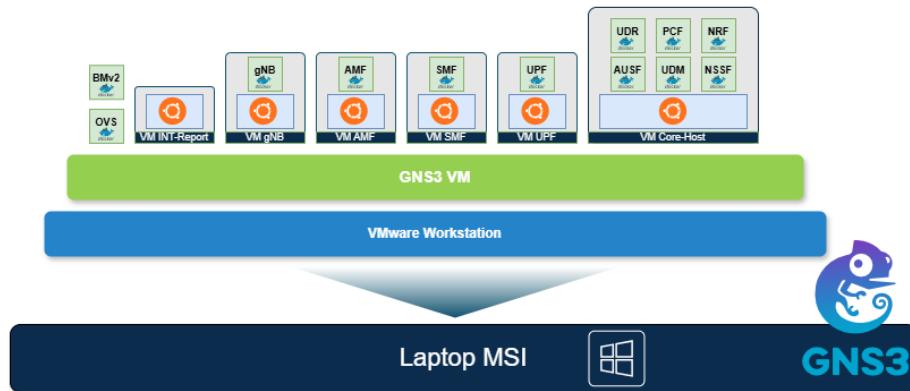


Figure 5.1: Arquitectura general

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)

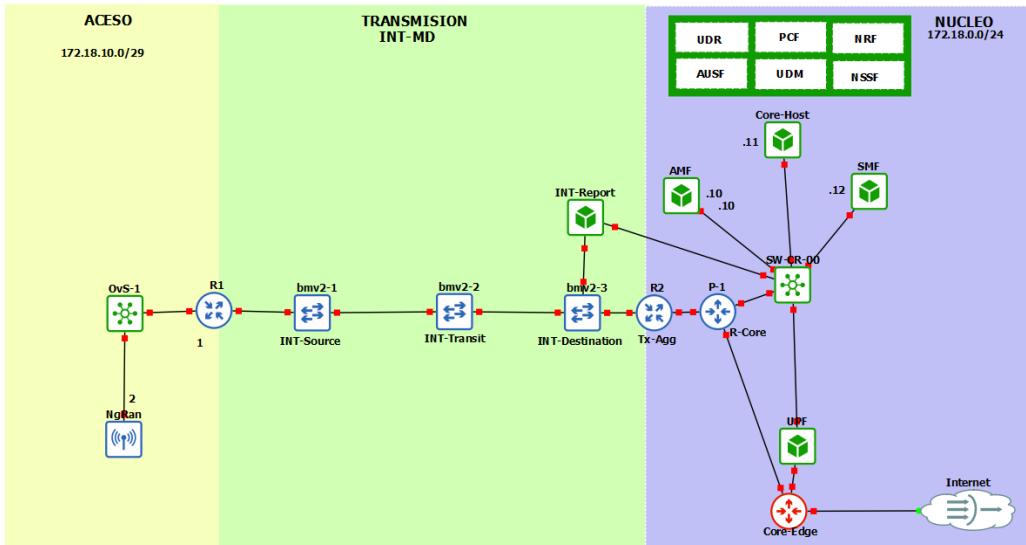


Figure 5.2: Topología de red implementada en GNS3

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.

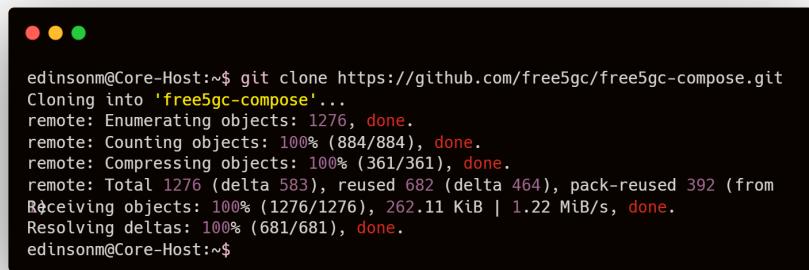
La configuracion de red de cada contenedor Docker se realizó con la modalidad de red **macvlan**, detallada en la tabla 5.1, que permite asignar una direccion IP en el mismo ranfo que la red fisica de la VM anfitriona. Esto facilita la comunicacion entre los contenedores y hace posible el establecimiento 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 [3].

5.2.1.1 Obtención del código fuente 5GC

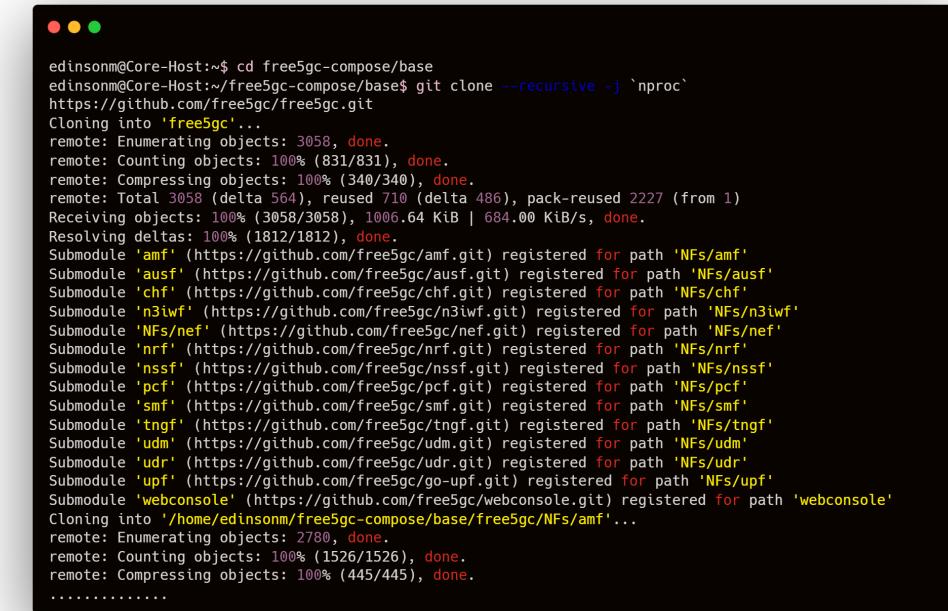
El primer paso para desplegar el Core 5G SA es clonar el repositorio oficial de free5gc desde GitHub como se muestra en la figura 5.3.



```
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:d2b4675e1918dcba8634d2459306d1f3b91f08c7293380f10585 15.76MB / 15.76MB 39.9s
=> sha256:f767b1746a83d257a6398cf8eec47bfa1f854670097ea4234f12857fcfd593 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
=> sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 32B / 32B 199.8s
=> => extracting sha256:ec335f17d0c74f7a270925cb1bbd29acc72ae904c6f4570f9ae369e3eebb64ed 25.2s
=> => extracting sha256:d2b4675e1918dcba8634d2459306d1f3b91f08c7293380f10585 3.7s
=> => extracting sha256:f7f67b1746a83d257a6398cf8eec47bfa1f854670097ea4234f12857fcfd5932 28.4s
=> => extracting sha256:fb73e3517bc8fa1e9863e448b6ad3b54c5482345b15ce6ad28439cf1f017523 23.0s
=> => extracting sha256:0635831cef590cc18d45a50a86211bcdcb7e2dc2aeac5a3d19671604e6b7e3d9 38.8s
=> => extracting sha256:579c95fc9df094db635a801a036ce19e0793e293b61b4621bac859996c4bbd47 0.0s
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 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

```

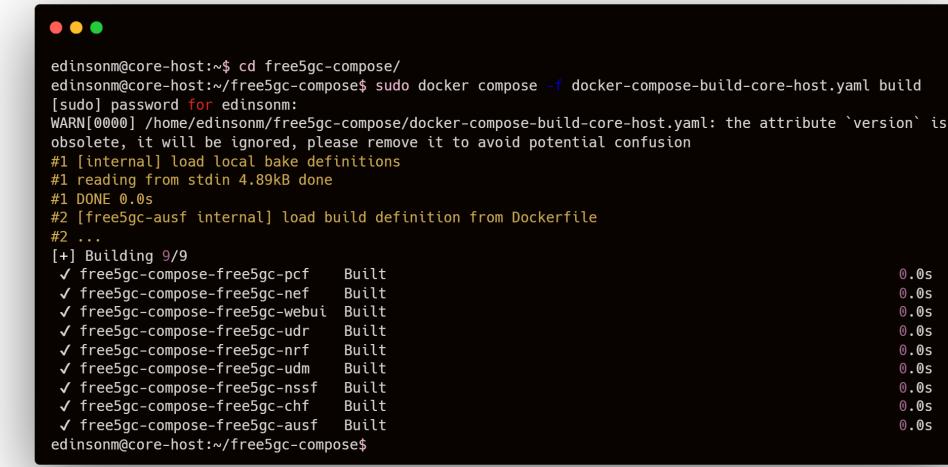
1 networks:
2   macvlan_net:
3     external: true
4
5 volumes:
6   dbdata:

```

Código 5.11: Configuración de red y volúmenes en Docker Compose

5.2.1.3 Construccion y despliegue

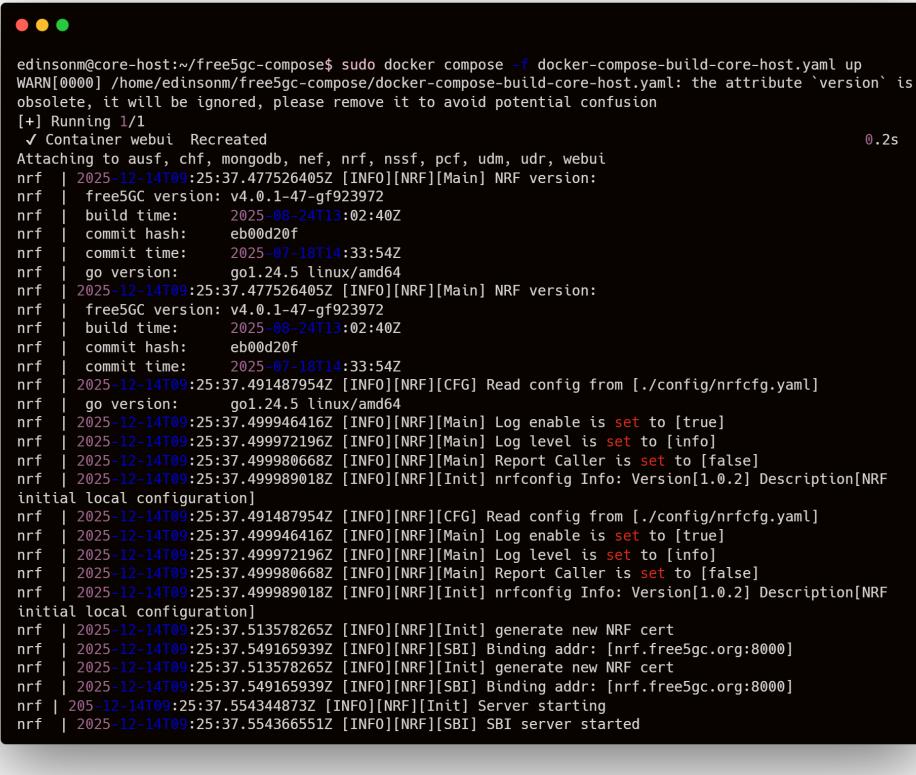
Una vez definido el archivo Docker Compose con la configuración de todas las NFs de Core-Host, se procede a construir las imágenes de acuerdo con el archivo Docher Compose como se puede apreciar en la figura 5.6.

A screenshot of a terminal window on a Mac OS X system. 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. The build process then lists nine services: free5gc-pcf, free5gc-nef, free5gc-webui, free5gc-udr, free5gc-nrf, free5gc-free5gc-udm, free5gc-nssf, free5gc-chf, and free5gc-ausf. Each service is marked with a checkmark and labeled 'Built'. To the right of each service name is its build duration, all of which are 0.0s.

```
edinsonm@core-host:~/free5gc-compose$ cd 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 .dockignore
=> => 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 .dockignore
=> => 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:acd76169aa75d716f3875b30dba2ac69789c10595cc38315181c9374d6a35dee
=> => 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
  docker image ls 'free5gc'/'base':'latest')

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-amf.yaml* con la configuración del AMF mostrado en el siguiente código 5.12.

```
1 | services:
```

```

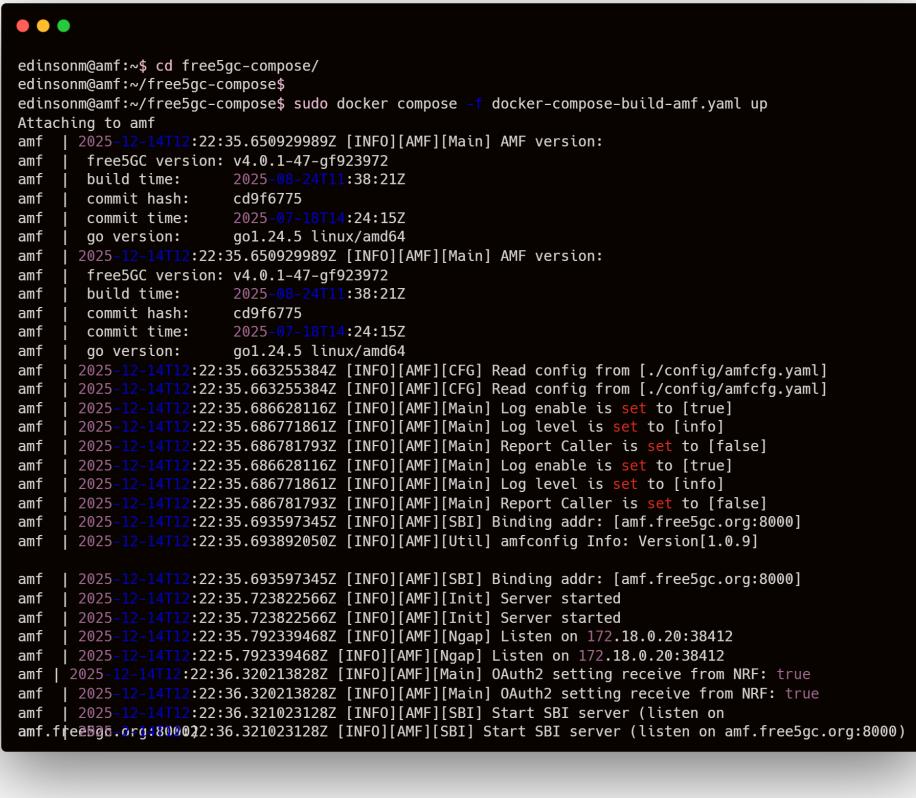
2   free5gc-amf:
3     container_name: amf
4     build:
5       context: ./nf_amf
6       args:
7         DEBUG_TOOLS: "false"
8       command: ./amf -c ./config/amfcfg.yaml
9     expose:
10      - "8000"
11      - "38412"
12     volumes:
13       - ./config/amfcfg.yaml:/free5gc/config/amfcfg.yaml
14       - ./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 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    | 2025-12-14T12:22:36.184283550Z [INFO][NRF][urillist update
nrf    | 2025-12-14T12:22:36.184283550Z [INFO][NRF][urillist update
nrf    | 2025-12-14T12:22:36.220642118Z [INFO][NRF][NFM] Create NF Profile: 8773b008-7c8e-4476-aef0-5ba67af6c6d2
nrf    | 2025-12-14T12:22:36.220642118Z [INFO][NRF][NFM] Create NF Profile: 8773b008-7c8e-4476-aef0-5ba67af6c6d2
nrf    | 2025-12-14T12:22:36.241341176Z [INFO][NRF][NFM] Use NF certPath: cert/amf.pem
nrf    | 2025-12-14T12:22:36.241341176Z [INFO][NRF][NFM] Use NF certPath: cert/amf.pem
nrf    | 2025-12-14T12:22:36.299832727Z [INFO][NRF][GIN] | 201 |      172.18.0.20 | PUT | /nrrf-nfm/v1/nf-instances/8773b008-7c8e-4476-aef0-5ba67af6c6d2 |
nrf    | 2025-12-14T12:22:36.299832727Z [INFO][NRF][GIN] | 201 |      172.18.0.20 | PUT | /nrrf-nfm/v1/nf-instances/8773b008-7c8e-4476-aef0-5ba67af6c6d2 |

```

Figure 5.10: Verificación del registro del AMF en el NRF

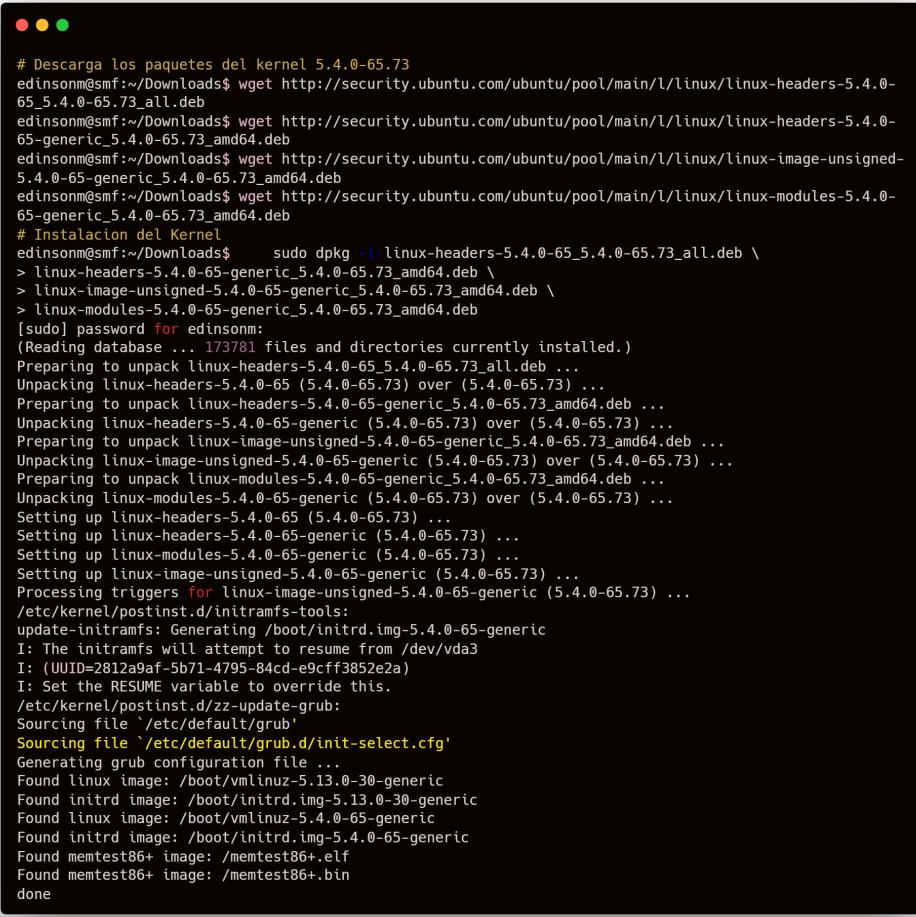
Las configuraciones del AMF, como las direcciones IP y puertos de escucha, se realizan en el archivo de configuración *amfcfg.yaml* ubicado en el directorio *./config/* del repositorio del AMF, las cuales no se muestran en este documento por cuestiones de espacio.

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.



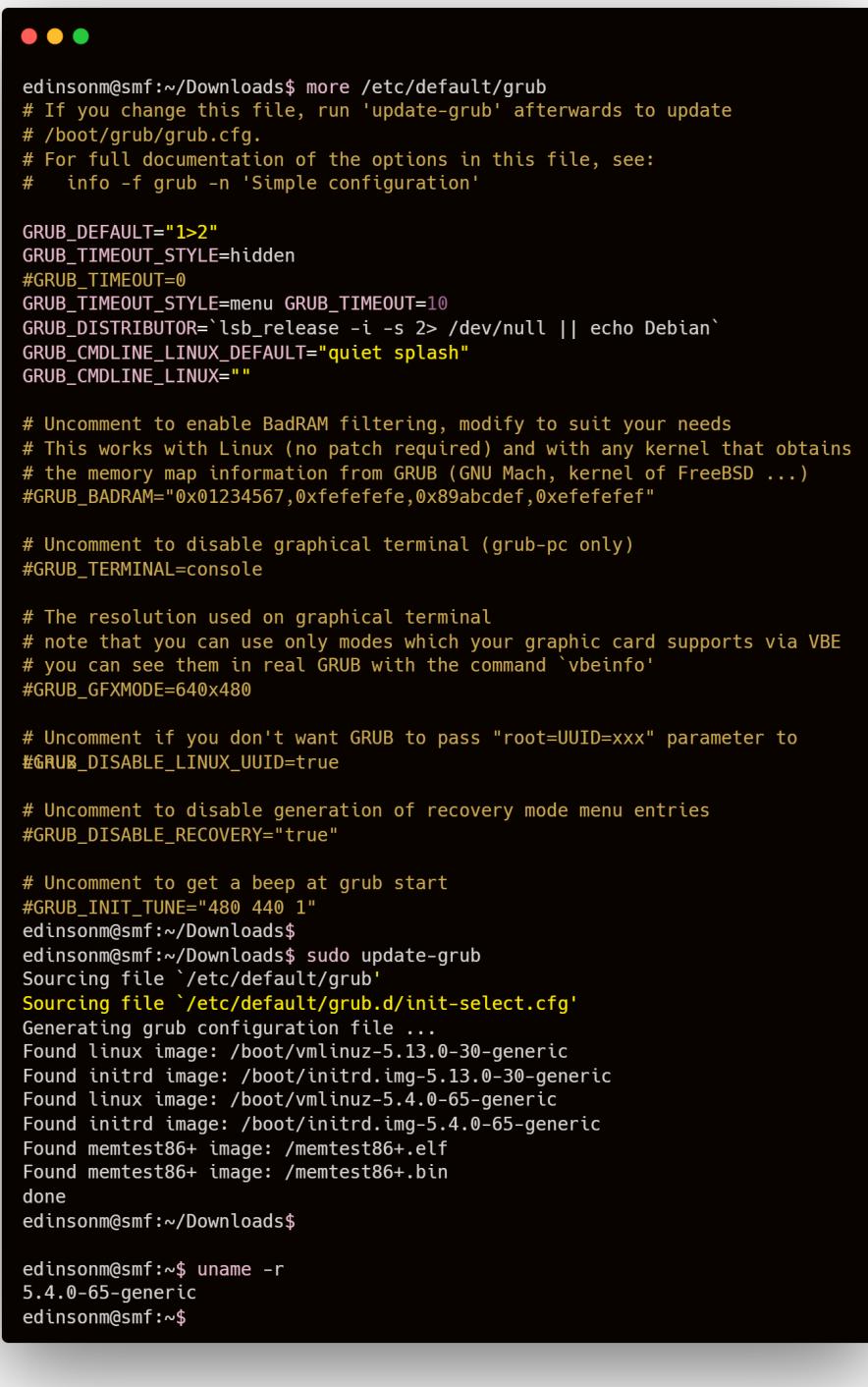
```

# 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 also contains comments about BadRAM filtering and VBE resolution.

After the configuration file, the terminal shows the command:

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

Followed by the output of the update-grub command, which lists found Linux and initrd images:

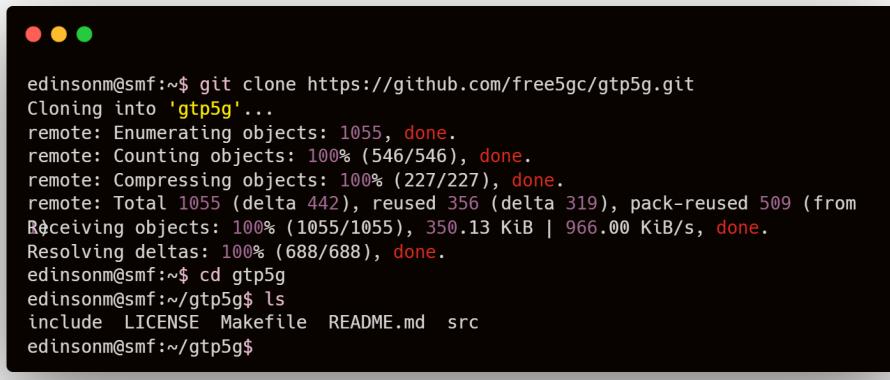
```
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.13.0-30-generic
Found initrd image: /boot/initrd.img-5.13.0-30-generic
Found linux image: /boot/vmlinuz-5.4.0-65-generic
Found initrd image: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

Finally, the terminal shows the system's kernel version:

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

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

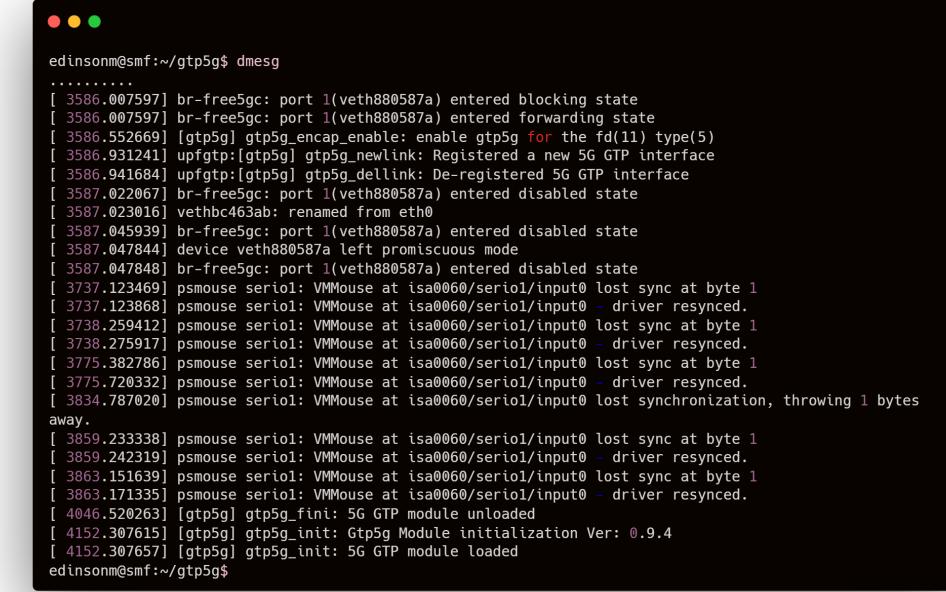
Luego de instalado el modulo GTP, se carga el modulo gtp5g en el kernel y luego se listan los modulos cargados para verificar que el modulo GTP se haya cargado correctamente, como se muestra en la figura 5.15.



```
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 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-smf.yaml* con la configuración del SMF mostrado 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-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*.



```

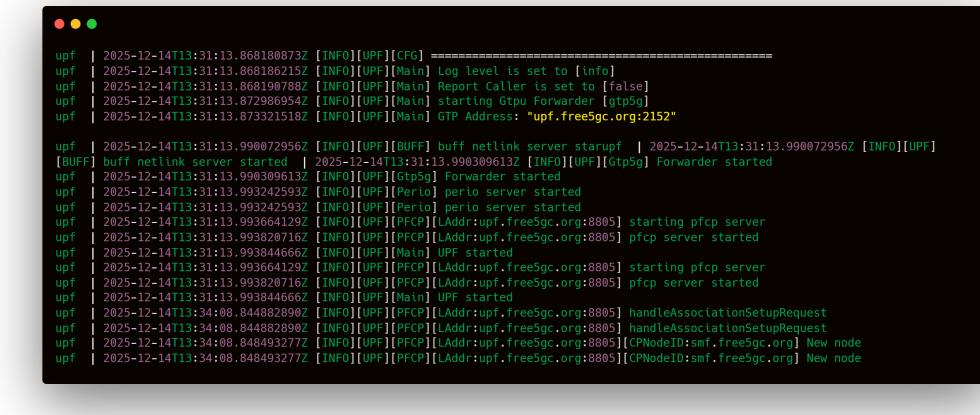
edinsonm@upf:~/free5gc-compose$ sudo docker compose -f docker-compose-build-upf.yaml up
[sudo] password for edinsonm:
WARN[0000] /home/edinsonm/free5gc-compose/docker-compose-build-upf.yaml: the attribute `version` is obsolete, it will be ignored,
please remove it to avoid potential confusion
Attaching 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

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] Docker Inc. Install docker desktop on ubuntu, 2024.
- [4] Free5GC Community. Free5gc: Open source 5g core network, 2021.
- [5] 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.