UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Comunicaciones

Evaluación y Comparativa de controladores SDN para despliegues de redes 5G.

Trabajo Fin de Máster

Máster Universitario en Tecnologías, Sistemas y Redes de Comunicaciones

AUTOR/A: Loza Lluco, Jhon Javier

Tutor/a: Gómez Barquero, David

CURSO ACADÉMICO: 2022/2023

Objetivos – El objetivo principal del presente trabajo es el estudio de controladores SDN open source para el despliegue en redes 5G que permitan la automatización, integración de nuevas funcionalidades y arquitecturas como NFV y redes MNO. Comparar los diferentes controladores y seleccionar el más adecuado para evaluar sus capacidades en un entorno de pruebas.

Desarrollo de prototipos y trabajo de laboratorio – Se evaluará los distintos tipos de controladores en un entorno de laboratorio, provisto de las capacidades de computo necesarias para alojar los controladores SDN y equipamiento de red real y virtualizado para probar de forma más compleja las bondades de integrar SDN en un despliegue 5G.

Resultados – La elección del controlador será en base a requerimientos óptimos de computo, flexibilidad, escalabilidad, integración con otras tecnologías o arquitecturas para NFV/MNOs, funcionalidades para un despliegue 5G. Una vez se decida por un controlador, se comprobará los beneficios de integrarlo en una red 5G mediante pruebas en un laboratorio virtualizado para desplegar Network Slicing.

Líneas futuras – Debido a las demandas de ultra baja latencia, flexibilidad y escalabilidad de las redes, nuevos modelos de infraestructura de red como Mobile Edge Computing están siendo desplegados para las redes 5G, en este tipo de entornos la automatización mediante tecnologías SDN y NFV tienen un papel muy importante al momento de realizar tareas de OAM. También en las nuevas capacidades de red como Network Slicing, OpenGateway o Multioperator Platforms, es necesario contar con agentes de red capaces de comunicarse de una forma flexible y ágil con todos los agentes internos y externos mediante APIs tanto NBI como SBI.

Abstract – The present work consists of the technical analysis of the different open source controllers available, which are OPENDAYLIGHT and TERAFLOW. The objective is to understand how they work and how they can be integrated in a 5G network deployment to offer the best performance and exploit the network capabilities offered by the new 5G core, which, thanks to the API-based interfaces, offers flexibility in the core construction and the possibility to integrate with external agents such as SDN controllers through SBI/NBI interfaces.

The evaluation metrics considered are scalability, flexibility, resource consumption, level of complexity in the integration with 5G infrastructure. Thanks to the availability of 5G infrastructure it is possible to perform tests in the environment corresponding to the laboratory of the mobile communications group in the iTeam of the UPV.

Resumen – El presente trabajo consiste en el análisis técnico de los diferentes controladores open source disponibles, los cuales son, OPENDAYLIGHT y TERAFLOW. El objetivo es comprender su funcionamiento y cómo pueden integrarse en un despliegue de red 5G para ofrecer el mejor rendimiento y explotar las capacidades de red que ofrece el nuevo core 5G, el cual, gracias a las interfaces basadas en APIs ofrece una flexibilidad en la construcción del core y la posibilidad de integrarse con agentes externos como controladores SDN mediante interfaces SBI/NBI.

Las métricas de evaluación consideradas es la escalabilidad, flexibilidad, consumo de recursos, nivel de complejidad en la integración con infraestructura 5G . Gracias a la disponibilidad de infraestructura 5G es posible realizar pruebas en el entorno correspondiente al laboratorio del grupo de comunicaciones móviles en el iTeam de la UPV.

Autor: Jhon Javier Loza Lluco, email: jjlozllu@teleco.upv.es
Director 1: David Gómez Barquero, email: dagobar@iteam.upv.es
Fecha de entrega: 07-09-23

Índice

1. Introducción y Motivación

El despliegue de la red 5G ha ido creciendo a lo largo de los últimos años a nivel mundial, en un principio conjugando el modelo híbrido con la tecnología 4G, 5G-NSA (Non Stand Alone), y actualmente ya existen despliegues comerciales de la red nativa 5G conocida como 5G-SA (Stand Alone). Este despliegue conlleva una evolución de los servicios reales que esta tecnología puede ofrecer y de los cuales hay mucha expectativa tanto a desde la perspectiva de los usuarios como de los propios operadores y demás actores de la industria.

El 2022 fue el despegue definitivo de esta nueva era de las comunicaciones móviles, la GSMA estima que para el 2025 más del 25 por ciento de todas las comunicaciones en el mundo será 5G, lo que equivale a 2.000 millones. En una primera fase de este despliegue 5G servirá como un acelerador para los servicios de alta velocidad y aplicaciones industriales, para el sector público y privado.[1]

En España la operadora Orange ha sido la primera en encender 5G-SA, espera llegar a 11 ciudades con 5G-SA, la cual está desplegada desde febrero del 2023 y esperan haber ofrecido esta cobertura al 30 por ciento la población española al termino del mes de junio. Su competidora local, Telefónica, ha encendido el 5G-SA o comercialmente conocido como 5G+, el pasado mes de julio, cubriendo un total de 11 ciudades españolas. El ministerio de asuntos económicos y transformación digital a través del programa UNICO Redes 5G backhaul, apoya e incentiva a los distintos operadores mediante incentivos económicos y ayudas para evitar que se genere una brecha digital con respecto a 5G.[7] [12]

Uno de los principales objetivos por parte de los operadores es monetizar al máximo el 5G y conseguir el retorno de la inversión en infraestructura que no obtuvieron con 4G, para ello se establecen varios pilares mediante los cuales lograr este objetivo común de la industria de las Telecomunicaciones.

Uno de estos pilares es las redes abiertas, ya que a diferencia de las redes 3G y 4G que dependían de un hardware y software específico provisto por un número reducido de proveedores, 5G se beneficia de la red abierta, de las cuales surgen nuevos conceptos como la programabilidad de la red gracias a plataformas de red como servicio (NaaS) y nuevas capacidades en la red soportados por tecnologías como NEF, NWDAF, PCF, OpenRAN, SDN, NFV, Coud-Native, etc. [10] [16]

1.1. 5G y SDN

El núcleo de red 5G ha sufrido una transformación en todos los niveles con respecto a su sucesor aunque son compatibles entre sí. La primera gran diferencia es su conceptualización con su arquitectura basada en servicios (SBA) con la cual se logra tener servicios de red modularizados. Estos servicios modularizados corresponden a las funciones de red 5G las cuales se comunican entre sí mediante el concepto de interfaces basadas en servicios (SBI), es decir, las funciones de red que se encargan de la señalización exponen y hacen disponibles los servicios de otras funciones de red. La Figura 1 muestra la arquitectura de red 5GC y las funciones de red que lo componen. [14] Esta nueva arquitectura permite que

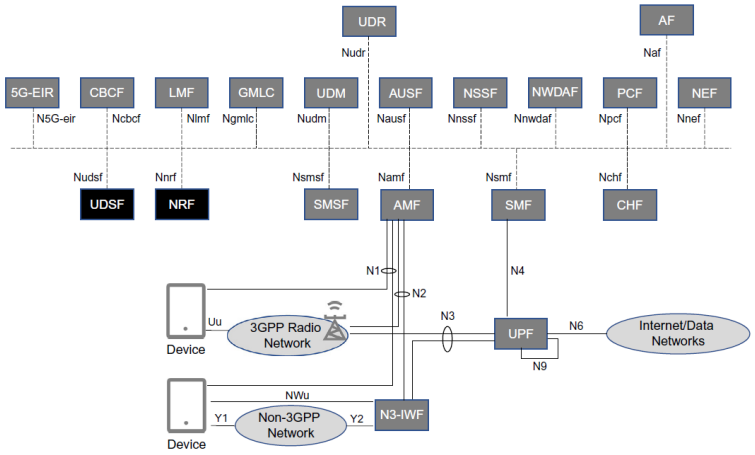


Figura 1: 5GC con interfaces basadas en servicios [14]

funcionalidades soportadas en una función de red específica esté disponible y accesible a través de una API mediante el protocolo de comunicación REST que se basa en HTPP.

La función de red que explota al máximo estas capacidades y permite la integración con servicios y tecnologías fuera del 5GC es el NEF (Network Exposurte Function). El cual expone capacidades de red seleccionadas que pueden ser utilizadas de diversas formas por aplicaciones. Esto es de interés para abrir nuevas oportunidades de negocio para los proveedores de servicios al permitir que terceros proveedores de aplicaciones ofrezcan servicios más avanzados. Una de las funcionalidades más interesantes del NEF es permitir que aplicaciones externas activen dispositivos para realizar acciones específicas relacionadas con la aplicación, incluida la conexión a la NEF o a la propia aplicación. [14]

Una vez se entiende la arquitectura y los conceptos de este nuevo core 5G, es fácil asociarlo con el paradigma de redes definidas por software implementado en la industria desde hace varios años y que cada vez tiene más presencia en el despliegue de redes. Gracias a que abre la posibilidad de separar el plano de control del plano de datos se puede integrar

aplicaciones que interactúen con equipos de red y en conjunto con el modelo de productor-consumidor del core 5G y su modelo SBA, permite introducir capacidades como Network Slicing o Mobile Edge Computing (MEC).

El uso de SDN permite básicamente la desagregación y virtualización de muchas de las funciones de telecomunicaciones y movilidad UPF (User Plane Function, AMF (Access and Mobility Management Function), SMF(Session Management Function), NEF (Network Exposure Function), Internet Protocol (IP) Routing, y Ethernet Encapsulation/Switching. Estas funciones se alojan como servicios de software y se instancian dinámicamente en diferentes partes de los segmentos de la red; así, la red 5G global está diseñada para ser configurable por software. En la Figura 2 se muestra como se integra SDN en una red 5G. [4]

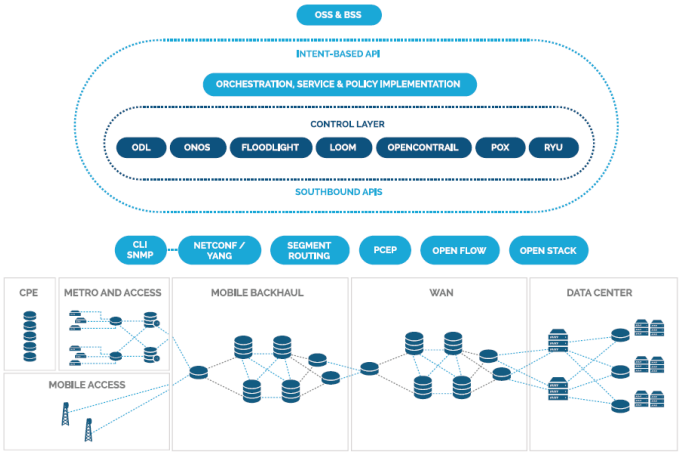


Figura 2: El rol de SDN para 5G [3]

Por lo tanto, la naturaleza de las dos arquitecturas hace que SDN sea considerado como uno de los principales factores tecnológicos para 5G y logre cumplir con su objetivo de ser una plataforma abierta, flexible, escalable, ágil y programable. SDN tiene como objetivo entonces de impulsar el desarrollo de las capacidades 5G, entre ellas, Network Slicing. Esta capacidad esta pensada para permitir a los operadores abrir su plataforma de infraestructura de red física al despliegue simultáneo de múltiples redes lógicas autónomas, orquestadas de diferentes maneras en función de los requisitos de servicio específicos que demanden.

1.2. Redes de transporte para 5G

5G ofrecerá más y mejores servicios, concurriendo en mayor carga de datos que los carriers deben transportar sin permitir que afecten a la calidad de los mismos, por lo que a nivel de transporte la red debe estar preparada para afrontar este aumento de

requerimientos en el rendimiento y también su integración al nuevo modelo de redes de siguiente generación las cuales deben entre otras cosas permitir la programabilidad.

Las redes de transporte no solo deben afrontar el aumento de tráfico en la red, si no también, con la descomposición del core 5G y la implementación del Mobile Edge Computing (MEC), debe ser capaz de ofrecer canales de comunicación dedicados, con alta disponibilidad, ultra baja latencia y ancho de banda simétrico constanste, para permitir que el nuevo núcelo de red 5G que puede estar distribuido geográficamente en diferentes Network Functions, se comunique con la rapidez necesaria para satisfacer la demanda de los clientes.

Network Slicing no solo consite en que el 5GC establezca un slice para un determinado servicio y se tarife su uso, requiere establecer canales específicos tanto en el acceso radio (RAN) como en la red de transporte, es decir, el objetivo final de Network Slicing es crear un túnel end-to-end con los recursos necesarios para satisfacer las necesidades de un servicio determinado, como se muestra en la Figura 3.

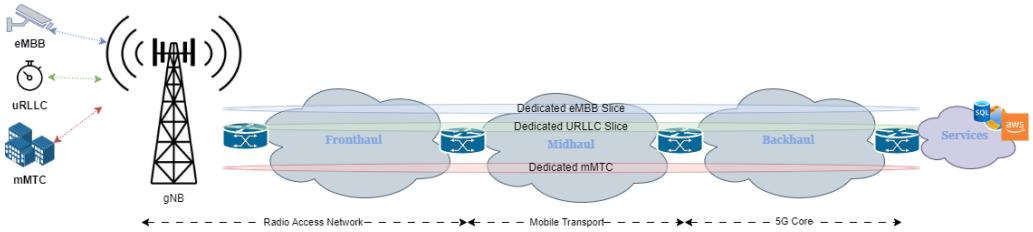


Figura 3: Network Slicing

La red midhaul transporta el tráfico entre la unidad de distribución (DU) que va desde el fronthaul al midhaul y la unidad centralizada (CU) que conecta el midhaul y el backhaul y tiene unos requisitos de latencia más estrictos, entre 1-5 milisegundos, que el tráfico de backhaul (<20 ms), pero no tan estrictos como fronthaul (150-200 us). Esto se logra a través de la interfaz F1 estandarizada 3GPP que utiliza encapsulación Ethernet e IP basada en estándares para la capa de transporte. El tráfico fronthaul puede combinarse con el tráfico backhaul e incluso con el tráfico midhaul. Esto significa que una plataforma macrocelular debe soportar tecnologías de transporte backhaul además de fronthaul. Estas tecnologías backhaul suelen incluir VPN L2 y L3 que se ejecutan sobre MPLS. En la actualidad, estas redes de backhaul MPLS se basan en protocolos como el Protocolo de Distribución de Etiquetas (LDP) o el Protocolo de Reserva de Recursos - Ingeniería de Tráfico (RSVP-TE), pero cada vez más las arquitecturas están evolucionando hacia MPLS con Segment Routing como un subyacente de paquetes impulsado por SDN. [4]

2. Objetivos

El objetivo de esta tesis está enfocado en evaluar y comparar distintos controladores SDN para elegir el que ofrezca mejores características para ser integrado en un despliegue de red 5G. Los resultados de esta tesis pretenden mostrar las ventajas y desventajas de utilizar SDN y cómo puede ayudar en el desarrollo de las capacidades 5G como Network Slicing. La integración de un controlador tiene como objetivo cumplir con el objetivo de construir redes flexibles, escalabes y programables por lo tanto es necesario diseñar y desplegar un entorno de pruebas en el cual se muestre cómo debe estar construida la red de transporte para poder cumplir con estos requisitos de despliegue. De igual forma es importante desplegar un core 5G para mostrar cómo funciona Network Slicing en el midhaul y backhaul, ya que es un entorno de pruebas no es posible hacer uso de una radio para integrar al laboratorio pruebas de Slicing en el fronthaul pero puede ser interesante un trabajo futuro en base a las conclusiones obtenidas.

3. Estado del arte

Telefónica Vivo Brasil lanzó en 2022 en conjunto con Cisco y NEC una red de transporte IP de nueva generación, flexible y escalable 5G para sus redes móviles y fijas convergentes. La simplificación y automatización de una red convergente multidominio y multicapa, el proyecto Fusion Network, junto con Segment Routing aumentará la escalabilidad y flexibilidad de la red de Telefónica Vivo mientras se prepara para lanzar 5G. Además, se implantará el enrutamiento por segmentos IPv6 para permitir una arquitectura unificada con capacidad de Network Slicing. [11]

La GSMA en su release sobre Network Slicing, establece que las redes definidas por software (SDN) apoyada por la Open Network Automation Platform (ONAP) que es una iniciativa de las operadoras en cooperación con la Linux Foundation, es una de las tecnologías más importantes para seguir impulsando el open source en 5G y ayude al desarrollo de las nuevas capacidades. También establece Segment Routing y L3VPN como protocolos para mejorar la red de transporte IP. [9]

3.1. Network Slicing

Esta capacidad surge con el objetivo de separar el tráfico en varias redes lógicas que se ejecutan y comparten una infraestructura física común. No existe un consenso general en cómo se establecerán estos slices a lo largo de toda la red, el 3GPP en sus especificaciones lo define sobre la parte radio y el core 5G. Mientras que otros actores de la industria como la GSMA definen Network Slicing como un concepto para ejecutar múltiples redes lógicas personalizadas en una infraestructura común compartida que cumpla los SLA (Service Legal Agremments) acordados para diferentes clientes y funcionalidades solicitadas. Para lograr este objetivo, Network Slicing necesita que la arquitectura de Slicing se diseñe desde

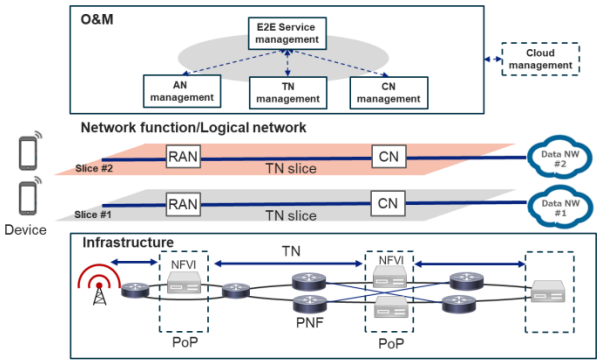


Figura 4: GSMA Arquitectura para Network Slicing [8]

una perspectiva E2E, abarcando diferentes dominios técnicos (por ejemplo, dispositivo, red de acceso, red central, red de transporte y sistema de gestión de red) y múltiples proveedores.[8]

Network Slicing en el core 5G definido por la 3GPP, se identifica mediante un parámetro denominado Single Network Slice Selection Assistance Information (S-NSSAI). Este parámetro tiene una longitud de 32 bits, 8 bits son para el tipo de slice o servicio (SST) y 24 bits son para el diferenciador de Slices (SD). El SD es opcional y sirve para diferenciar entre distintos Slicenes usados para distintos clientes. En la tabla 1 se puede ver los diferentes Slices que ha definido el 3GPP. Antes de que un equipo de usuario (UE) pueda

Valor de SST	Slice estandarizado	Reservado
0	eMBB	Para servicios de banda ancha móvil mejorada
1	URLLC	Para servicios de ultra fiable baja latencia
2	MIoT	Para servicios Massive IoT
3	V2X	Para servicios de V2X
5-127		Reservado
128-255		Especificado por el operador

Tabla 1: Valores de SST definidos por el 3GPP [14]

acceder a un Slice, tiene que registrarlo en la red, lo que se hace mediante el procedimiento de registro. Para que un mismo UE pueda acceder a varias Slices, es necesario que pueda enviar una o varias S-NSSAI al mismo tiempo desde el equipo de usuario a la red y desde la red al UE. Por lo tanto, en una NSSAI pueden proporcionarse una o más S-NSSAI. Aunque a veces se utiliza "S-NSSAI(s).º "S-NSSAIs"para describir que puede haber una o más S-NSSAIs. El conocimiento de dónde están disponibles los Slices, desde una perspectiva independiente del UE, puede ser configurado por OM y también difundido por señalización entre las entidades conectadas entre sí. La función de red NSSF es configurada por OM sobre dónde deben estar disponibles los Slices. A continuación, la información sobre la disponibilidad de los segmentos de red se difunde mediante señalización a través de las interfaces N22, N2 y Xn. [14]

3.2. Segment Routing

Segment Routing (SR) es un protocolo de red que está estandarizado por la IETF. SR combina las mejores características de MPLS y Source Routing. La codificación del camino consiste en colocar una secuencia de instrucciones en la cabecera del paquete, a cada instrucción se le conoce con el nombre de segmento, el cual utiliza una identificación recibiendo la nomenclatura para la instrucción de Segment ID. Por lo tanto, en Segment Routing la ruta no depende de la señalización mediante saltos como en un IGP, un protocolo de distribución de etiquetas (LDP) o RSVP. Una característica que hace de SR una tecnología muy interesante para el futuro de las redes es su adopción a SDN, ya que puede funcionar con un plano de datos MPLS o IPv6, consiguiendo que el mecanismo de reenvío de paquetes sirva como alternativa a OpenFlow.

En un dominio SR que tiene un nodo con capacidad SR-IGP anuncia segmentos para sus prefijos y adyacencias. Estos segmentos se denominan segmentos IGP o SID IGP y juegan un papel muy importante en segment routing ya que permiten la expresión de cualquier ruta en todo el dominio SR, dicha ruta se expresa como un solo segmento IGP o una lista de múltiples segmentos IGP. Los anuncios de los segmentos IGP requieren de extensiones en los protocolos de estado del enlace, los únicos protocolos que soportan estas extensiones son IS-IS y OSPF. Para las extensiones de SR en los IGPs, se definen los IGP-Prefix Segment el cual es un segmento IGP unido a un prefijo IGP, este segmento es global a menos que se anuncie lo contrario, dentro de un dominio SR. El contexto de un segmento de prefijo IGP (IGP-Prefix Segment) incluye los prefijos, la topología y el algoritmo, múltiples SIDs pueden ser asignadas al mismo prefijo a lo largo de una tupla (prefijo, topología y algorítmo), se mantiene único. Este IGP-Prefix Segment identifica la ruta, relacionada al prefijo, calculada según el algoritmo asociado. [5]

Segment Routing para 5G

SR combina el concepto de enrutamiento basado en la fuente con la determinación centralizada de rutas para permitir arquitecturas de red IP simplificadas, ingeniería de tráfico escalable y mecanismos de resistencia de red más eficientes. Cada una de estas características es fundamental para la realización y prestación de servicios 5G nuevos y emergentes.

La ingeniería de tráfico (TE) permite determinar la ruta que seguirán los flujos de servicio a través de una red y garantizar que la ruta pueda garantizar el ancho de banda, la latencia y/o la fiabilidad necesarias para que los servicios que se ejecuten sobre esa conexión funcionen según lo previsto. El enrutamiento por segmentos aporta un enfoque simplificado a la ingeniería de tráfico al eliminar la necesidad de protocolos como RSVP-TE y LDP. También tiene la capacidad de soportar ingeniería de tráfico de "grano grueso.º de "grano fino.en función de los requisitos del operador de red. Las funciones centralizadas de control y cálculo de rutas de SDN también pueden aprovecharse para aportar más

visibilidad y una mejor toma de decisiones a la selección de rutas con el fin de garantizar una utilización óptima de los recursos.[15]

SR ya está implantado en redes de producción de todo el mundo con proveedores de servicios como Telefónica, Bell Canada, BT, Colt, SoftBank, China Unicom, VF Germany y Comcast. A partir de los anuncios públicos de estos operadores, se desplegó el enrutamiento por segmentos para alcanzar los siguientes objetivos:

- Mejorar la escalabilidad de la red, reducir el capex y el opex

- Mejorar la disponibilidad de la red

- Admite políticas por servicio

- Soporte de SLA comprometidos y/o deterministas

- Apoyo a la fragmentación de la red

- Ofrecer servicios diferenciados/de valor añadido

- Red móvil altamente fiable que puede gestionar servicios 5G/IoT

La mayoría de las implantaciones SR actuales utilizan MPLS como plano de reenvío. Esto se conoce en el sector como SR-MPLS, que funciona sin problemas en redes IPv4 e IPv6. Con la madurez de los estándares y soluciones SR-MPLS, los proveedores de routers están desarrollando y sacando al mercado otras implementaciones SR que pueden desplegarse en otros planos de reenvío, incluido IPv6 sin MPLS. [2]

3.3. Controladores SDN

El controlador es el elemento más importante en una implementación con SDN, es el encargado de gestionar, administrar y controlar el plano de control por lo tanto las características que tenga el controlador permitirá tener más o menos aplicaciones para realizar. En el mercado se puede encontrar distintas soluciones tanto propietarias como open source.

Los controladores propietarios son desarrollados por vendors especializados y tienen como ventaja que ofrecen productos con aplicaciones de muy alto nivel con lo cual permiten al usuario gestionar las redes en todos los aspectos, sin embargo, su gran desventaja es la necesidad de comprar una licencia y que funcionará sin problemas únicamente con su hardware. Ejemplos de estos controladores son Cisco ACI, VMware NSX, HP VAN, Nauge Nokia, Juniper NorthStart, etc.

Entre las soluciones open source existing una gama muy amplia de controladores, dependerá de las características que se quieran utilizar, la industria ha impulsado el desarrollo de algunos proyectos como OpenDaylight; fundado por la Linux Foundation en 2013, en colaboración con vendors como Brocade, Juniper, Cisco, Ericsson, IBM, Microsoft, NEC, Red Hat y VMware, es una plataforma modular abierta para personalizar y automatizar redes de cualquier tamaño y escala. ONOS; es también un proyecto de la Linux Foundation desde el 2014, tiene como objetivo gestionar componentes de red, como routers y enlaces, y ejecutar programas o módulos de software para proporcionar servicios de comunicación a hosts finales y a la red vecina. TeraFlowSDN; es un proyecto desarrollado por la ETSI para crear un controlador SDN cloud-native para redes IP y ópticas. Tiene el apoyo de grandes empresas como Telefónica, NEC, Atos, etc.[6]

4. Evaluación y comparación de controladores SDN

4.1. Comparación y selección del controlador

Entre las soluciones open source, Opendaylight y ONOS son controladores muy parecidos, construidos sobre la misma arquitectura, utilizan REST API para la comunicación de sus módulos y aplicaciones y las construyen sobre la plataforma de Apache Karaf OSGi. La mayor diferencia recae en que Opendaylight tiene un enfoque para tecnologías como BGP y está orientada a redes de siguiente generación NGN. Mientras que ONOS se centra en el rendimiento y clusterización de las redes, aumentando la disponibilidad y la escalabilidad de las redes.

Debido a la similitud en la arquitectura de ODL y ONOS, se va a evaluar Opendaylight y TeraFlowSDN.

Capacidad	TeraflowSDN	Opendaylight
Despliegue	VM	VM, Dockers, K8s, Clúster
Construcción	Microservicios, K8s	Apache Karaf OSGi
Requerimientos	4 vCPUs, 8GB RAM, 60GB HDD	2 vCPUs, 2GB RAM, 16GB HDD
Interfaz Usuario	GUI, CLI, GRAFANA	GUI, CLI
API	TAPI, SBI,NBI	API REST, SBI, NBI
Protocolos	Netconf, Openconfig,P4, gNMI	Netconf, Openflow, BGP, Openconfig,etc
Servicios	L2/L3 VPN/Slices, ACLs, IPv4	BGP-LS, SR-TE, MPLS, L2/L3 VPN
Vendors	Nokia SR Linux	Cisco, Juniper, Nokia, Ericsson, etc
Integración	OpenSource MANO	Openstack, ONAP

Tabla 2: Comparación Opendaylight vs TeraFlowSDN

En la tabla 2 se puede observar las ventajas y desventajas de cada controlador, cabe resaltar que TeraFlowSDN está aún en una fase de desarrollo, mientras que Opendaylight

lleva años desarrollado con varias releases de por medio pero es interesante esta comparativa ya que TeraFlowSDN pretende ser ese controlador que permite programar las redes en todo el xHaul como se observa en la Figura 5.

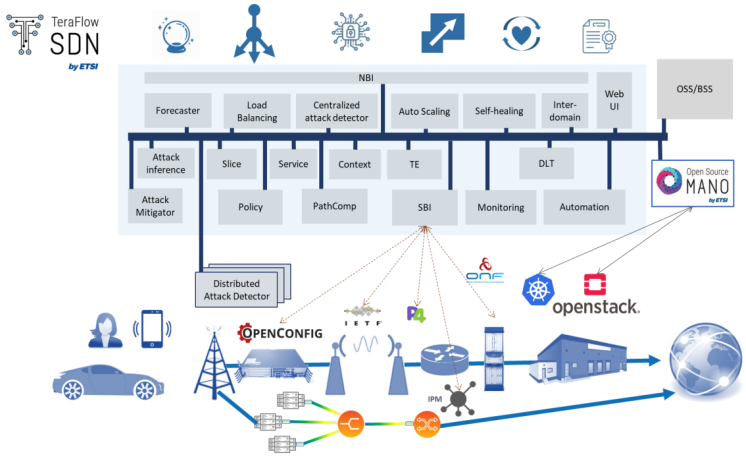


Figura 5: Arquitectura TeraflowSDN Release 2 [13]

Por lo tanto se instalaron los dos controladores en el laboratorio del grupo de comunicaciones móviles en el iTEAM de la UPV, para comprobar su despliegue, rendimiento y capacidades disponibles para poder integrarlo a una red 5G para probar Network Slicing.

4.2. Instalación y pruebas de los controladores

Como se puede observar en la Figura 6, se han desplegado 2 máquinas virtuales con el sistema operativo Ubuntu Server 22.04 LTS en uno de los servidores virtualizados del laboratorio, las cuales se conectan al router Juniper MX204 para establecer una sesión de netconf. En primer lugar se ha instalado el controlador TFS el cuál al estar basado

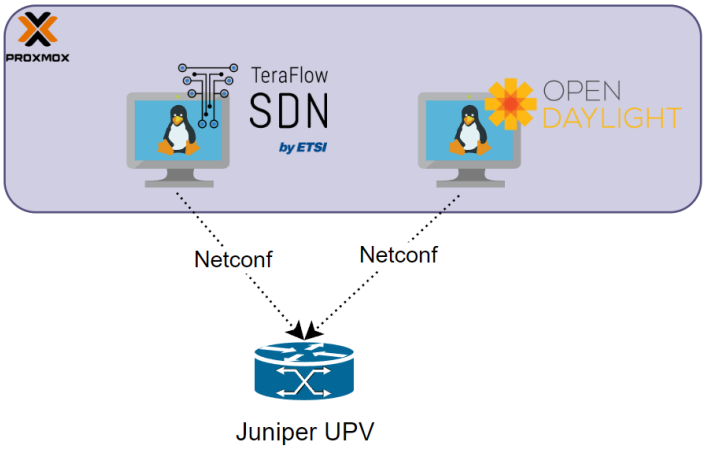


Figura 6: Topología laboratorio UPV para evaluar el rendimiento de ODL y TFS

en microservicios hace uso de MicroK8s Kubernetes para desplegar los contenedores y

orquestarlos, ya que cada contenedor va a ejecutar una capacidad para el controlador. Gracias a esta arquitectura, permite desplegar TFS con un script en el cual se puede indicar las capacidades a desplegar, por ejemplo, indicar que se despliegue Grafana para controlar los KPIs de los dispositivos añadidos, desplegar el componente de gestión de políticas y automatización, componente de ciberseguridad o el componente de ingeniería de tráfico para establecer los slices. En el ejemplo siguiente se muestran los pasos a seguir para iniciar el controlador y comprobar su funcionamiento.

```
cd ~/tfs-ctrl           //Directorio donde se ubican los recursos descargados del controlador
cat /my_deploy.sh       //Observar los componentes habilitados
vi /my_deploy.sh        //Se modifica los componentes a ser desplegados
source my_deploy.sh     //Indicar al SO las variables a utilizar
./deploy/all.sh         //Iniciar el despliegue
kubectl get all --all-namespaces   //Verificar el despliegue
```

Si la instalación ha sido exitosa, se deben haber generado los pods correspondientes a los servicios deseados, a continuación se puede observar como debería lucir la respuesta al verificar el despliegue.

```
NAMESPACE            NAME                                      READY   UP-TO-DATE   AVAILABLE   AGE
monitoring           deployment.apps/grafana                   1/1     1            1           31d
kube-system          deployment.apps/hostpath-provisioner      1/1     1            1           31d
kube-system          deployment.apps/coredns                   1/1     1            1           31d
monitoring           deployment.apps/blackbox-exporter         1/1     1            1           31d
monitoring           deployment.apps/prometheus-adapter        2/2     2            2           31d
monitoring           deployment.apps/kube-state-metrics        1/1     1            1           31d
kube-system          deployment.apps/calico-kube-controllers   1/1     1            1           31d
monitoring           deployment.apps/prometheus-operator       1/1     1            1           31d
linkerd              deployment.apps/linkerd-identity          1/1     1            1           31d
container-registry   deployment.apps/registry                  1/1     1            1           31d
kube-system          deployment.apps/metrics-server            1/1     1            1           31d
linkerd-viz          deployment.apps/grafana                   1/1     1            1           31d
linkerd              deployment.apps/linkerd-destination       1/1     1            1           31d
linkerd-viz          deployment.apps/web                       1/1     1            1           31d
linkerd-viz          deployment.apps/tap-injector              1/1     1            1           31d
linkerd              deployment.apps/linkerd-proxy-injector    1/1     1            1           31d
linkerd-viz          deployment.apps/metrics-api               1/1     1            1           31d
linkerd-viz          deployment.apps/tap                       1/1     1            1           31d
linkerd-viz          deployment.apps/prometheus                1/1     1            1           31d
```

Una vez se tiene listo el controlador, se puede generar el archivo de configuración en formato JSON para añadir un dispositivo mediante netconf como se muestra a continuación.

```
{
    "devices": [
        {
            "device_id": {"device_uuid": {"uuid": "R1"}},
            "device_type": "packet-router",
            "device_config": {"config_rules": [
                {"action": 1, "custom": {"resource_key": "_connect/address",
                "resource_value": "172.31.0.4"}},
                {"action": 1, "custom": {"resource_key": "_connect/port",
                "resource_value": "8301"}},
                {"action": 1, "custom": {"resource_key": "_connect/settings",
                "resource_value": {
                    "username": "admin", "password": "LKADSKLAHDLA",
                    "force_running": false, "hostkey_verify": false,
                    "look_for_keys": false,
                    "allow_agent": false, "commit_per_rule": true,
                    "device_params": {"name": "default"},
                    "manager_params": {"timeout" : 15}
```

```
                }}}
            ]},
            "device_operational_status": 1,
            "device_drivers": [1],
            "device_endpoints": []
        }
    ]
}
```

En el router juniper se debe configurar netconf de la siguiente forma, donde se habilita el protocolo netconf en el mismo puerto en el que va a enviar las peticiones el controlador.

```
root@juniper1> show configuration
## Last commit: 2023-07-30 14:33:00 UTC by root
version 20.4R2.7;
fxp0 {
        unit 0 {
            family inet {
                address 172.31.0.4/16;
            }
        }
    }
    services {
        ssh {
            root-login allow;
        }
        telnet;
        netconf {
            ssh {
                port 8301;
            }
            rfc-compliant;
            yang-compliant;
            traceoptions {
                file netconf-ops.log size 5m;
                flag all;
            }
        }
    }
```

Si las conexiones son correctas al iniciar la sesión netconf desde el controlador, la respuesta debería ser la siguiente:

```
root@jjlozlluvm:~/tfs-ctrl/hackfest/netconf# python3 client_topology.py
{'urn:ietf:params:netconf:capability:validate:1.0',
'urn:ietf:params:netconf:capability:confirmed-commit:1.0',
'urn:ietf:paramate:1.0', 'http://xml.juniper.net/dmi/system/1.0',
'urn:ietf:params:xml:ns:netconf:capability:candidate:1.0',
'urn:ietf:paramsonitoring',
'urn:ietf:params:xml:ns:netconf:capability:validate:1.0',
'urn:ietf:params:netconf:base:1.0',
'http://xml.juniper.n:ietf:params:xml:ns:netconf:capability:url:1.0
scheme=http,ftp,file', 'urn:ietf:params:netconf:capability:url:1.0
```

En esta respuesta, el controlador SDN ha obtenido las capabilities que soporta el router. Es decir que la conexión ha sido exitosa, por lo tanto se puede hacer ya un GET para ver el estado y la configuración del juniper.

```
root@jjlozlluvm:~/tfs-ctrl/hackfest/netconf# python3 client_topology.py
---GET CONFIG---
b'<?xml version=\'1.0\' encoding=\'utf8\'?>\n<nc:data
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:junos="http://x/junos">\n<configuration
xmlns="http://yang.juniper.net/junos/conf/root" junos:commit
seconds="1690727580" junos:commit-localtC" junos:commit-user="root">\n
```

```
<version>20.4R2.7</version>\n    <system
xmlns="http://yang.juniper.net/junos/conf/system">\1</host-name>\n
</login>\n        <services>\n            <ssh>\n                <root-
login>allow</root-login>       <telnet>\n            </telnet>\n
<netconf>\n                <ssh>\n                    <port>8301</port>\n
<rfc-compliant/>\n                <yang-compliant/>\n
<traceoptions>\n                    <file>\n ename>netconf-
ops.log</filename>\n                        <size>5m</size>\n
</file>\n                  <f    <name>all</name>\n
</flag>\n                </traceoptions>\n            </netconf>\n
</services>per1.campus5g.corp</domain-name>\n        <backup-router>\n
<address>172.30.0.2</address>\n        </backup-router>
<name>172.31.0.2</name>\n        </name-server>\n        <schema>\n
<openconfig>\n                <unhideg>\n        </schema>\n
<interfaces xmlns="http://yang.juniper.net/junos/conf/interfaces">\n<intee-
0/1/0</name>\n
<vlan-tagging/>\n<unit>\n
<name>0</name>\n<vlan-id>1313 <family>\n
<inet>\n<address>\n<name>192.168.13.1/24</n</address>\n
</inet>\n
</family>\n</unit>\n</interface>\n<int fxp0</name>\n<unit>\n
<name>0</name>\n
<family>\n<inet>\n<name>172.31.0.4/16</name>\n
</address>\n</client-identifier>\n mx204:FG443</vendor-id>\n
<routing-options xmlns="http://yang.juniper.net/junos/conf/routing
options">\n
<name>172.30.2.0/24</name>\n<next-hop>172.31.0.3</next-hop>\n
<retain/>\n     >\n             </route>\n            <route>\n
<name>0.0.0.0/0</name>\n                <next-hop>172.31.0.3</nee>\n
</static>\n    </routing-options>\n    <protocols
xmlns="http://yang.juniper.net/junos/conf/protocols">\n        <
<interface>\n                <name>fxp0.0</name>\n            </interface>\n
</router-advertisement>\n    </pro</nc:data>\n'
```

Se pueden observar que contiene la información correspondiente al router, como la versión del sistema operativo que corresponde a la 20.4R2.7, la configuración de netconf, las direcciones IP, las VLAN, rutas estáticas y otra información de la configuración. Algo que se destaca es la presentación de esta respuesta, ya que no es fácil leerla y no interactúa directamente con el proceso que corresponde a la página web, es decir, aunque TSF se ha conectado con el router, no se puede observar el dispositivo en la interfaz gráfica. En la Figura 7 se observa la captura de los paquetes al iniciar la sesión netconf en el puerto 8301 y el intercambio de información.

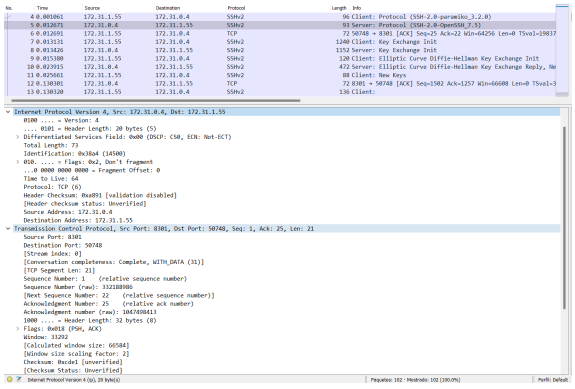
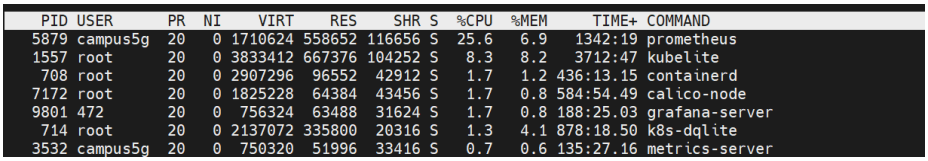


Figura 7: Captura de paquetes entre TFS y Juniper UPV

En la Figura 8 se puede observar el consumo de recursos, donde se consume el 22,6 de la RAM y el 41 de CPU.



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 5879 campus5g  20   0 1710624 558652 116656 S  25.6   6.9   1342:19 prometheus
 1557 root      20   0 3833412 667376 104252 S   8.3   8.2   3712:47 kubelite
  708 root      20   0 2907296  96552  42912 S   1.7   1.2 436:13.15 containerd
 7172 root      20   0 1825228  64384  43456 S   1.7   0.8 584:54.49 calico-node
 9801 472       20   0  756324  63488  31624 S   1.7   0.8 188:25.03 grafana-server
  714 root      20   0 2137072 335800  20316 S   1.3   4.1 878:18.50 k8s-dqlite
 3532 campus5g  20   0  750320  51996  33416 S   0.7   0.6 135:27.16 metrics-server
```

Figura 8: Consumo de recursos TFS

Para instalar Opendaylight es necesario JAVA ya que Karaf hace uso de este recurso para iniciar el controlador. Luego simplemente es necesario iniciar el controlador como se muestra a continuación:

```
root@jjlozlluvm:~/karaf-0.7.3/bin# ./karaf
Apache Karaf starting up. Press Enter to open the shell now...
100% [========================================================================]
Karaf started in 11s. Bundle stats: 393 active, 394 total

    ________                       ________                .__  .__       .__     __
    \_____  \ ______   ____   ____ \______ \ _____  ___.__.|  | |__| ____ |  |___/  |_
     /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
    /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
    \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
            \/|__|        \/     \/        \/     \/\/            /_____/      \/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

Una vez se muestra la consola de Karaf, se deben instalar los módulos de opendaylight, en esta parte, es algo parecido a lo que hace TeraflowSDN, los módulos ya vienen cargados en el controlador por lo que no es necesario descargarlos de internet, karaf lo que hace es iniciar esos procesos para que puedan ser usados, en este caso se habilitará el proceso de la interfaz gráfica, netconf y restconf como se muestra.

```
opendaylight-user@root>feature:install odl-netconf odl-dlux-all odl-netconf-connector odl-restconf
```

Una vez se tiene listo el controlador, se puede utilizar Postman ya que el protocolo restconf permite al controlador interactuar con aplicaciones de terceros. Para conectar el router juniper se realiza de la siguiente forma.

```
PUT http://localhost:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/P1

<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
    <node-id>P1</node-id>
    <host xmlns="urn:opendaylight:netconf-node-topology">172.31.0.4
    </host>
    <port xmlns="urn:opendaylight:netconf-node-topology">8301</port>
    <username xmlns="urn:opendaylight:netconf-node-topology">admin
    </username>
    <password xmlns="urn:opendaylight:netconf-node-topology">AJKBDSAJBAKL
    </password>
    <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false
    </tcp-only>
    <keepalive-delay xmlns="urn:opendaylight:netconf-node-topology">0
    </keepalive-delay>
</node>
```

Una vez conectado se puede hacer un GET para obtener la información del router.

```
GET http://172.16.1.10:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/P1
```


```
{"node":[{"node-id":"P1","netconf-node-topology:available-capabilities":
{"capability":"http://xml.juniper.net/dmi/system/1.0","capability,
{"capability":"urn:ietf:params:xml:ns:netconf:capability:url:1.0?
scheme=http,ftp,file","capability-origin":"device-advertised"},
{"capability":"(http://yang.juniper.net/junos/rpc/dynamic-profile?
revision=2019-01-01)junos-rpc-dynamic-profile","failure-reason":"unable-
to-resolve"},{"capability":"(http://yang.juniper.net/junos/rpc/ethernet-
switching?revision=2019-01-01)junos-rpc-ethernet-switching","failure-
reason":"unable-to-resolve"},{"capability":"
(http://yang.juniper.net/junos/rpc/path-computation-client?revision=2019-
01-01)junos-rpc-path-computation-client","failure-reason":"unable-to-
resolve"}]},"netconf-node-topology:connection-
status":"connected","netconf-node-topology:port":8301}}]}
```


Se puede observar que el router ha enviado las capabilities que soporta y el estado de la sesión de netconf al final. En la Figura 9 se muestra el consumo de recursos que tiene este controlador. Consume un 24,3 de RAM y un 4,7 de CPU.

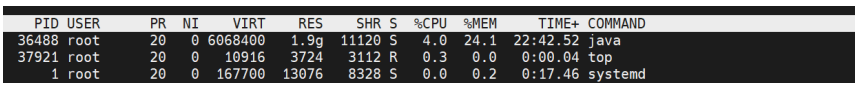


Figura 9: Consumo de recursos ODL

4.3. Selección del controlador

Como se puede observar en la comparativa, Opendaylight es mejor controlador SDN a día de hoy, porque tiene una mayor cantidad de componentes, protocolos, y características que han sido probadas con dispositivos reales y virtualizados de los vendors más importantes de la industria, esto también es importante decirlo, porque fue desarrollado por varios de ellos. Además se observa que a nivel de consumo de recursos TeraflowSDN aún no está optimizado y consume mucho más que Opendaylight.

TeraflowSDN es un proyecto que a penas en el mes de julio ha lanzado su versión 2.0 del controlador, tiene un enfoque acorde a las necesidades de la industria y pretende ser el integrador de la inteligencia en una red 5G, una de sus mayores características es la capacidad de poder establecer slicing end-to-end, gracias a que en su arquitectura pretenden integrar un módulo para NSSMF (Network Slice Subnet Management Function) por cada parte de la red, Acceso radio, red de transporte y core 5G, como se observa en la Figura 10.

Por lo que parece ser que TeraflowSDN será una de las tecnologías disruptivas para el futuro de las redes 6G.

Es por ello que para el presente trabajo se va a utilizar el controlador Opendaylight para ver cómo ofrecer una solución SDN en un despliegue de red 5G sobre una red legacy.

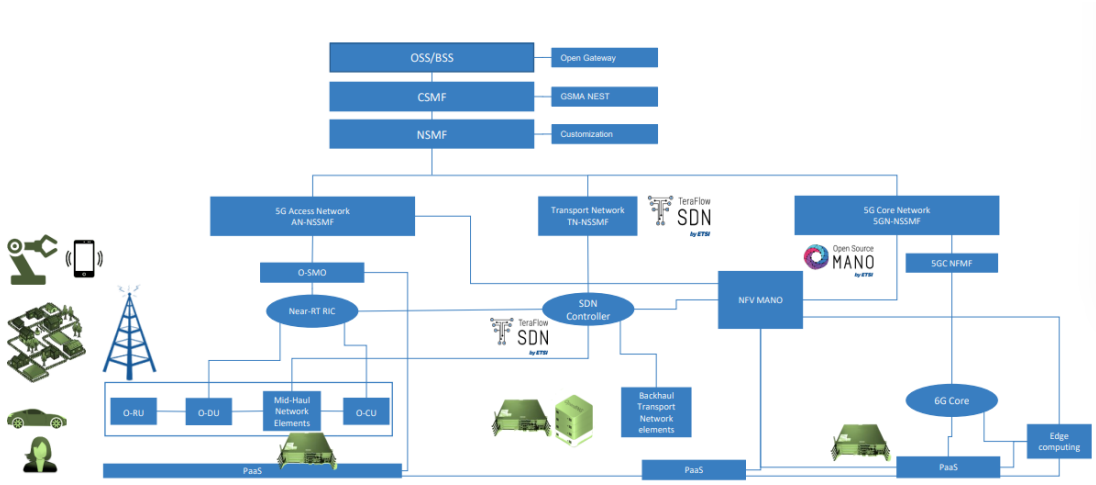


Figura 10: TeraflowSDN arquitectura para redes 6G [13]

5. Diseño y despliegue de red 5G virtualizada

5.1. Herramientas

GSN3:

Es un software open source utilizado para emular, configurar, probar y solucionar problemas de redes virtuales y reales. Este software proporciona una interfaz gráfica de usuario intuitiva para diseñar y configurar redes virtuales, funciona en hardware de PC tradicional y puede utilizarse en múltiples sistemas operativos, incluidos Windows, Linux y MacOS X.

Docker:

Es una plataforma abierta para desarrollar, distribuir y ejecutar aplicaciones. Permite separar las aplicaciones de su infraestructura para que pueda entregar software rápidamente. Docker ofrece la posibilidad de empaquetar y ejecutar una aplicación en un entorno poco aislado denominado contenedor. El aislamiento y la seguridad permiten ejecutar varios contenedores simultáneamente en un mismo host. Los contenedores son ligeros y contienen todo lo necesario para ejecutar la aplicación, por lo que no es necesario depender de lo que esté instalado en el host. Las Network Functions de Open5GS se desplegarán mediante esta tecnología.

CISCO IOSXR:

Es un sistema operativo de redes desarrollado por Cisco Systems Inc. Soporta tecnologías IP y Routing, MPLS, Multicast, QoS, BGP, Netconf, SDN, Segment Routing, IS-IS, OSPF, Openconfig, etc. Está orientado a dispositivos para operadores de servicios, como la linea NCS, 8000 y ASR.

Opendaylight:

Es el controlador elegido para evaluar su integración en un despliegue de red 5G. Se utilizará para comunicarse con la red de transporte y crear los túneles Segment Routing Traffic Engineering.

Pathman-SR:
Es una aplicación open source construida para comunicarse con un controlador SDN vía Restconf, consume los datos que obtiene del controlador como la topología de la red mediante BGP-LS, y utiliza PCEP para calcular y programar las rutas que se van a instalar en la red.

Postman:
Es una plataforma de API para crear y utilizar API. Postman simplifica cada paso del ciclo de vida de las API y agiliza la colaboración para crear mejores API, más rápido. Se utiliza para probar capacidades que no integra la aplicación Pathman-SR, como fijar un ancho de banda específico a los túneles SR-TE.

Open5Gs:
Es un proyecto de código abierto que proporciona funcionalidades de núcleo de red de paquetes móviles 5G para construir una red 5G privada bajo licencia GNU AGPL 3.0. Actualmente es compatible con la versión 16 de 3GPP y proporciona funciones de red 5G Core (AMF, SMF, UPF, PCF, UDR, UDM, AUSF, NRF).

UERANSIM:
Es una implementación de código abierto de UE 5G y RAN 5G (gNodeB). Se puede considerar como un teléfono móvil 5G y una estación base en términos básicos. Hay 3 interfaces principales en la perspectiva UE/RAN, 1) Interfaz de Control (entre RAN y AMF), 2) Interfaz de Usuario (entre RAN y UPF), 3) Interfaz de Radio (entre UE y RAN). UERANSIM es compatible con Open5GS y se puede conectar probar la funcionalidad.

5.2. Diseño de la red

En la Figura 11 se muestra la arquitectura de la red propuesta para validar la funcionalidad del despliegue integrando SDN, y apoyandose en Segment Routing se probará a configurar túneles SR-TE para proveer Network Slicing a nivel de red de transporte. Para esto se utilizará la aplicación de Postman que permite mediante una interfaz gráfica por web, permite manipular estos túneles virtuales de una forma muy dinámica. Con Postman se podrán probar capacidades adicionales las cuales no están incluidas en Pathman SR, pero que el controlador Opendaylight permite, además esto muestra como cualquier aplicación de terceros puede hacer uso de los datos que tiene el controlador mediante el protocolo de Restconf.

El controlador Opendaylight se ha configurado con BGP ya que mediante su propiedad de Link-State puede obtener la topología en tiempo real de la red, y de esta forma la puede mostrar en la interfaz gráfica tanto del controlador como de la aplicación Pathman SR lo

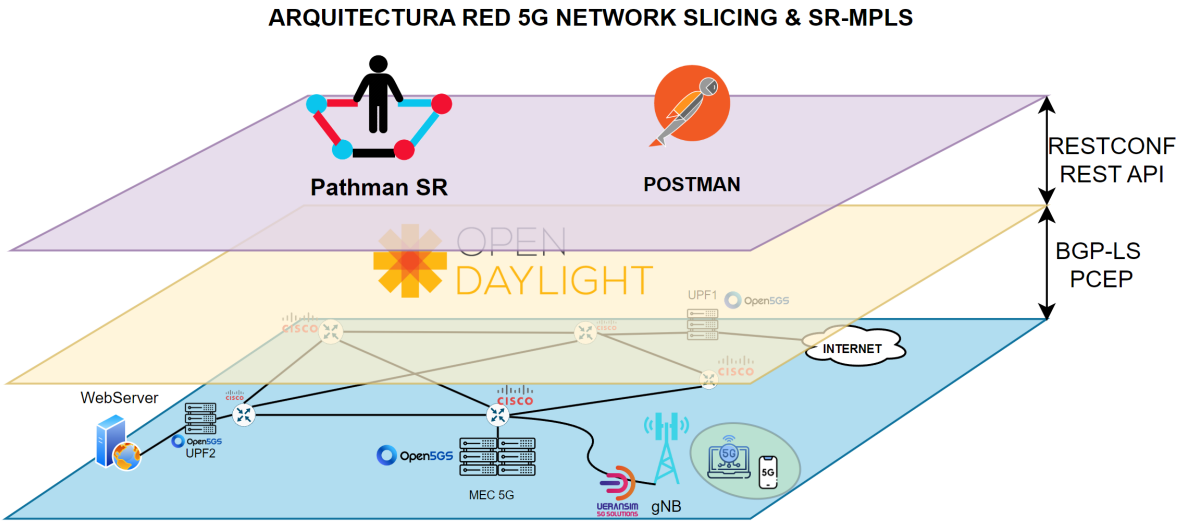


Figura 11: Red diseñada para probar la integración del controlador

cual es de gran ayuda para los operadores de red, que pueden ver de forma visual todos los dispositivos que tienen desplegados en la red, haciendo mucho más fácil y eficiente su trabajo. De la misma forma, mediante Postman, se puede consultar mediante el método GET la topología de la red y de esta forma obtener los datos, mediante los cuales puede manipular las rutas, calcular la ruta más corta, etc.

Para configurar los túneles de Network Slicing, se usa el protocolo PCEP, el cual es un protocolo estandarizado por la IETF, y que comunica el PCE (Path Computation Element) que será el rol de Opendaylight y el PCC (Path Computation Client) que es el rol de los dispositivos de red CISCO.

Los dispositivos de red se configuran con IS-IS que es un IGP para enrutamiento dinámico y mediante el cual pueden comunicarse entre sí. De igual forma se configura el enrutamiento BGP para que el controlador pueda obtener la topología mediante BGP-LS. Para la parte de Network Slicing se configura MPLS, pero sin usar LDP.

La parte del core 5G, gNB y UE, se han desplegado en dockers a los cuales se asigna la IP del rango privado y se indica mediante rutas estáticas por donde deben reenviar el tráfico para llegar a los UPFs, MEC 5G, gNB y UE. Direccionamiento IP, BGP, MPLS y SR:

5.3. Conectividad

Datos para la conexión por BGP:

Dispositivo	Interfaz	IP	Sistema Autónomo
ODL	eth0	172.23.29.120	65504
PE1	Loopback	192.168.99.1	65504
P1	Loopback	192.168.99.2	65504
P2	Loopback	192.168.99.4	65504
P3	Loopback	192.168.99.3	65504
PE2	Loopback	192.168.99.5	65504

Tabla 3: Direccionamiento IP de los dispositivos para BGP

Datos para la conexión por SR-MPLS:

Dispositivo	Interfaz	IP	SID
PE1	Loopback	192.168.99.1	16000
P1	Loopback	192.168.99.2	16001
P2	Loopback	192.168.99.4	16003
P3	Loopback	192.168.99.3	16002
PE2	Loopback	192.168.99.5	16004

Tabla 4: Datos del despliegue SR-MPLS

Datos para la conexión por IP:

Dispositivo	Interfaz	IP
UPF1	Eth0	172.23.28.112
UPF2	Eth0	172.23.27.112
SMF1	Eth0	192.168.3.9
SMF2	Eth0	192.168.3.10
AMF	Eth0	192.168.3.2
NSSF	Eth0	192.168.3.6
NRF	Eth0	192.168.3.7
gNB	Eth0	192.168.5.2
UE1	Eth0	192.168.5.3
UE2	Eth0	192.168.5.4
WebServer	Eth0	192.168.1.3

Tabla 5: Direccionamiento IP de los dispositivos para la red privada

Las tablas anteriores son las necesarias para configurar la comunicación entre todas las capas de la arquitectura, para el enrutamiento por IS-IS se utiliza la red 192.168.x.x/16. Y para postman se utiliza el PC Host con la IP 172.23.29.119, cualquier dispositivo que

se encuentre dentro del rango 172.23.29.96/27 podrá comunicarse con el controlador. La aplicación de Pathman-SR se ha instalado en la misma máquina virtual que Opendaylight.

En la Figura 12 se muestra con más detalle cómo es el despliegue de la red.

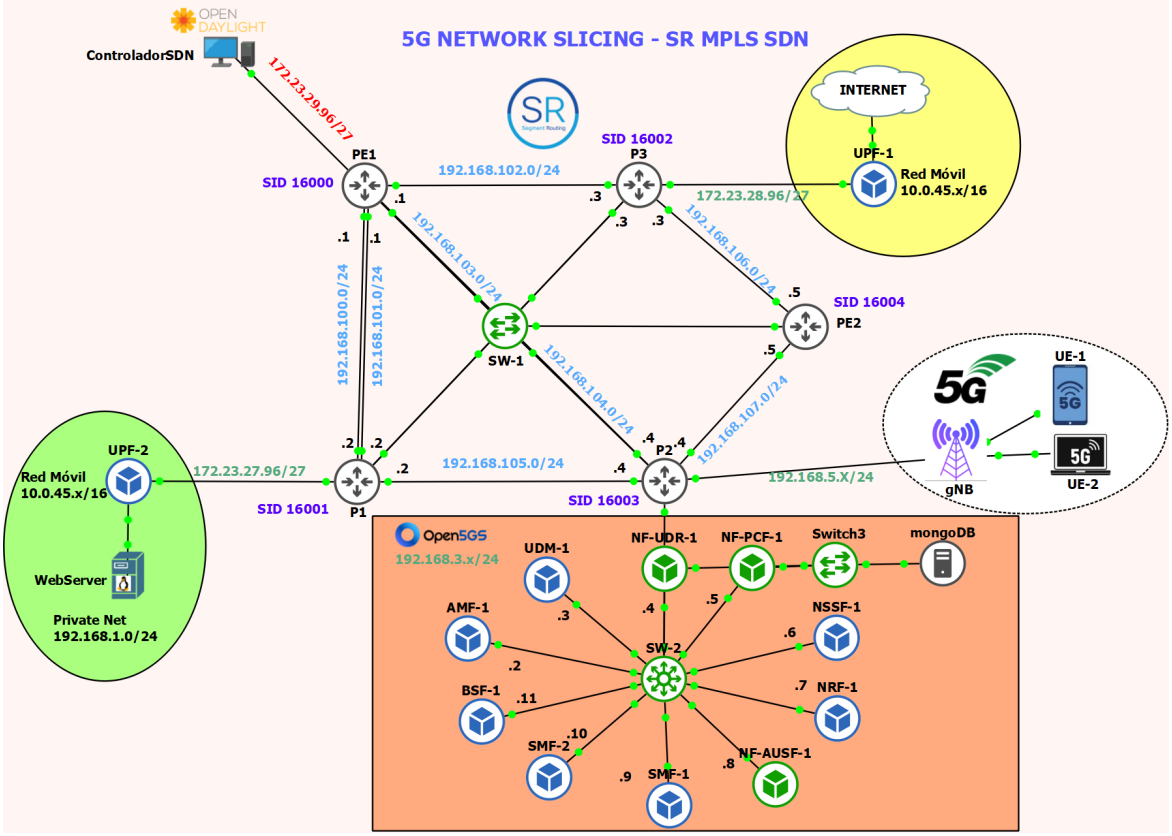


Figura 12: Topología del despliegue en el entorno GNS3

5.4. Pruebas de funcionamiento

Es necesario corroborar que todas las instancias tienen comunicación entre ellas, para que las pruebas del despliegue 5G Network Slicing funcionen correctamente.

Si el enrutamiento por IS-IS, la configuración de las redes privadas para las funciones de red de Open5GS y la configuración de cada NF para el despliegue del core es correcto, el NRF (Network Repository Function) deberá mostrar en sus logs cómo las distintas network functions se van registrando, a continuación se muestra el ejemplo de configuración del AMF y en la Figura 13 se observa cómo se registraron las NFs en el NRF.

```
logger:
  file: /open5gs/install/var/log/open5gs/amf.log
sbi:
  server:
    no_tls: true
    cacert: /open5gs/install/etc/open5gs/tls/ca.crt
    key: /open5gs/install/etc/open5gs/tls/amf.key
    cert: /open5gs/install/etc/open5gs/tls/amf.crt
  client:
```

```
    no_tls: true
    cacert: /open5gs/install/etc/open5gs/tls/ca.crt
    key: /open5gs/install/etc/open5gs/tls/amf.key
    cert: /open5gs/install/etc/open5gs/tls/amf.crt
amf:
  sbi:
    - address: 192.168.3.2
      port: 7777
  ngap:
    - address: 192.168.3.2
  metrics:
      address: 192.168.3.2
      port: 9090
  guami:
    - plmn_id:
        mcc: 001
        mnc: 01
      amf_id:
        region: 2
        set: 1
  tai:
    - plmn_id:
        mcc: 001
        mnc: 01
      tac: 1
  plmn_support:
    - plmn_id:
        mcc: 001
        mnc: 01
      s_nssai:
        - sst: 1
  security:
      integrity_order : [ NIA2, NIA1, NIA0 ]
      ciphering_order : [ NEA0, NEA1, NEA2 ]
  network_name:
      full: Open5GS
  amf_name: open5gs-amf0
nrf:
  sbi:
    - address: 192.168.3.7
      port: 7777
```

```
NRF-1 console is now available... Press RETURN to get started.
Open5GS daemon v2.6.0

08/25 11:36:41.318: [app] INFO: Configuration: '/open5gs/install/etc/open5gs/nrf
.yaml' (../lib/app/ogs-init.c:126)
08/25 11:36:41.318: [app] INFO: File Logging: '/open5gs/install/var/log/open5gs/
nrf.log' (../lib/app/ogs-init.c:129)
08/25 11:36:41.325: [sbi] INFO: nghttp2_server() [http://192.168.3.7]:7777 (../l
ib/sbi/nghttp2-server.c:238)
08/25 11:36:41.325: [app] INFO: NRF initialize...done (../src/nrf/app.c:31)
08/25 11:36:44.884: [nrf] INFO: [ab2646e2-433b-41ee-add4-0d93d17476bd] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:36:47.581: [nrf] INFO: [acc32c0e-433b-41ee-9213-9bf73173ecdc] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:37:00.224: [nrf] INFO: [b44d9c70-433b-41ee-b037-a32008866e0f] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:37:02.847: [nrf] INFO: [b5dceb22-433b-41ee-aa5b-1953a9545c1a] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:37:11.461: [nrf] INFO: [baeca30a-433b-41ee-b577-b3b078fe2fae] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:37:16.043: [nrf] INFO: [bdb87ed8-433b-41ee-872b-a97576033eea] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:37:20.645: [nrf] INFO: [c07183d6-433b-41ee-a682-dfdb69f3e248] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:37:24.885: [nrf] INFO: [c2f6cddc-433b-41ee-9e82-012f7d70d95f] NF regist
ered [Heartbeat:10s] (../src/nrf/nf-sm.c:190)
08/25 11:41:45.223: [app] INFO: Signal-NUM[28] received (Window changed) (../src
/main.c:63)
08/25 11:41:45.915: [app] INFO: Signal-NUM[28] received (Window changed) (../src/main.c:63)
```

Figura 13: Registro de las Network Functions en el NRF

Una vez se ha desplegado el Core 5G correctamente, se procede a iniciar las vecindades de BGP entre el controlador SDN y los dispositivos de red. Para habilitar BGP y PCEP en ODL es necesario instalar las siguientes herramientas.

```
opendaylight-user@root>feature:install odl-bgpcep-bgp odl-bgpcep-pcep
```

En primer lugar se debe configurar BGP el controlador SDN, que se hace mediante Postman con la siguiente API

```
<protocol xmlns="http://openconfig.net/yang/network-instance">
    <name>example-bgp-rib</name>
    <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
    <bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
        <global>
            <config>
                <router-id>172.23.29.120</router-id>
                <as>65504</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name>LINKSTATE</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

Con esta API, se configura BGP en el controlador, indicando el sistema autónomo al que va a pertenecer, su identificador dentro del sistema y que tendrá habilitado BGP-LS para conocer la topología en tiempo real. Después se puede configurar los vecinos, en este caso el vecino va a ser uno de los routers de la red desplegada, en este caso va a formar vecindad con PE1, y será este el encargado de pasarle toda la información de la red al controlador, la configuración para esto se muestra a continuación.

```
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>172.23.29.121</neighbor-address>
    <timers>
        <config>
            <hold-time>90</hold-time>
            <connect-retry>10</connect-retry>
        </config>
    </timers>
    <transport>
        <config>
            <remote-port>179</remote-port>
            <passive-mode>false</passive-mode>
        </config>
    </transport>
    <config>
        <peer-type>INTERNAL</peer-type>
    </config>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name>LINKSTATE</afi-safi-name>
        </afi-safi>
    </afi-safis>
</neighbor>
```

En la API se indica la IP del vecino (PE1), el puerto por el que va a propagar los mensajes de BGP, que por defecto y como se define en la IETF es el 179, el tipo de

vecindad si será iBGP (Interno) o eBGP (Externo), en este caso ya que estará dentro del mismo sistema autónomo, es Interno. Y también se indica que van a transmitirse mensajes con el estado del enlace. Si se ha configurado todo bien, se deberían levantar todas las vecindades, en la Figura 14 se muestra que en PE1 se han levantado todas las vecindades, entre los otros routers y también entre el controlador. En este momento ya

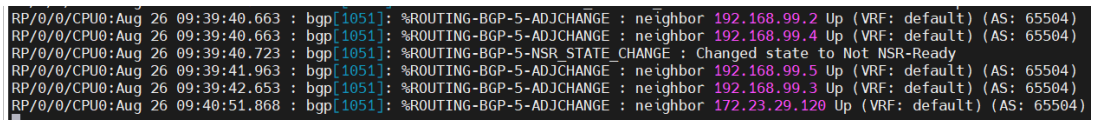


Figura 14: Topología del despliegue en el entorno GNS3

es posible mediante un método GET desde Postman, consultar la topología, en la Figura 15 se muestra un fragmento de la respuesta que corresponde a la información de P1.

Figura 15: Respuesta a la consulta de la topología BGP-LS desde Postman

Por último es necesario comprobar que el protocolo PCEP puede funcionar para ello en uno de los routers (PCC) se verifica que haya iniciado exitosamente la sesión con Opendaylight (PCE), en la Figura 16 se muestra el ejemplo del router PE1 y en la Figura 17 se muestra la información que tiene el controlador.

```
RP/0/0/CPU0:PE1#sh mpls traffic-eng pce peer
Tue Aug 22 17:04:41.155 UTC
        Address    Precedence        State         Learned From
--------------- ------------ ------------ --------------------
  172.23.29.120           255           Up        Static config
```

Figura 16: Conexión entre PCC (PE1) y PCE (ODL)


```
{
    "node-id": "pcc://192.168.99.1",
    "network-topology-pcep:path-computation-client": {
        "stateful-tlv": {
            "odl-pcep-ietf-stateful07:stateful": {
                "lsp-update-capability": true,
                "odl-pcep-ietf-initiated00:initiation": true
            }
        },
        "ip-address": "192.168.99.1",
        "state-sync": "synchronized"
    }
}
```


Figura 17: Información del estado del PCC (PE1) en ODL

En este momento toda la conectividad está lista, solo queda por comprobar que la aplicación Pathman-SR se pueda comunicar con el controlador y obtener la información necesaria para permitir la creación de los túneles SR-TE. Para ello se lanza la aplicación desde la máquina virtual y se accede a la interfaz web, en la Figura 18 se puede observar que la comunicación ha sido exitosa. En la parte central se muestra la topología con el

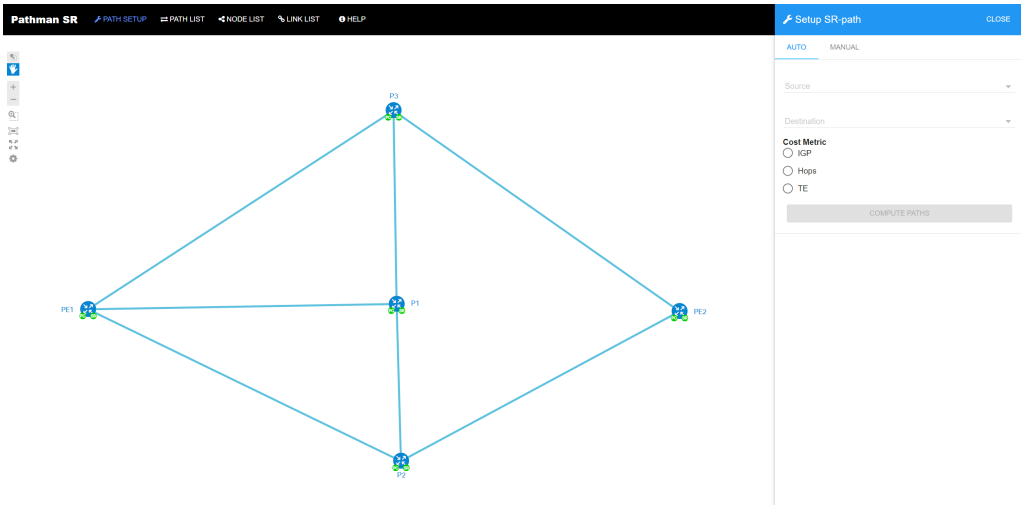


Figura 18: Topología de la red en Pathman-SR

nombre de los routers, en la parte superior están las opciones de configuración en las que con PATH SETUP se puede instalar un túnel SR-TE, en la parte de la derecha se muestran las opciones de esta herramienta, en la cual se indica el nodo origen y el nodo

destino del túnel y la forma de calcular la mejor ruta, ya sea con la métrica de IS-IS, en base al número de saltos, o mediante las políticas de Traffic Engineering. En PATH LIST se puede consultar los túneles ya creados, en NODE LIST se puede ver información relacionada a cada router y en LINK LIST se puede ver los enlaces point-to-point que tiene cada router.

6. Resultados y análisis

6.1. Creación de túneles SR-TE

Con el despliegue ya realizado, se instalan los túneles para Network Slicing, los túneles son unidireccionales por lo que para comunicar cada UPF con su respectivo SMF, debe existir un túnel para el uplink y otro para downlink. En la Figura 19 se pude ver el Slicing de Downlink para UE1.

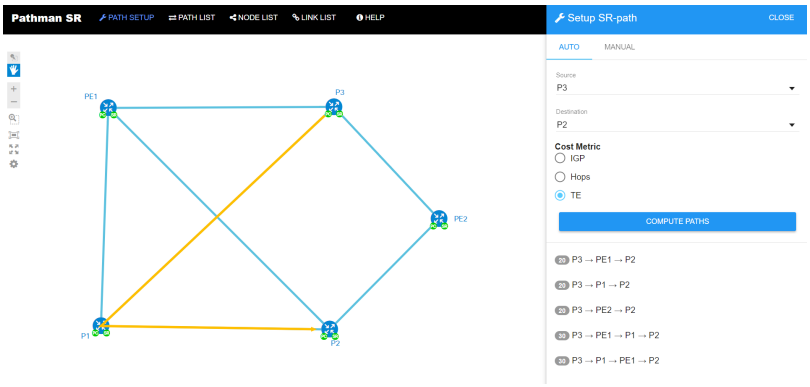


Figura 19: Instalación Slicing para Downlink UE1

La ruta configurada es P3 − > PE1 − > P2, se muestra una notificación de que la instalación ha sido exitosa como se observa en la Figura 20, incluso la ruta pasa del color naranja a color verde.

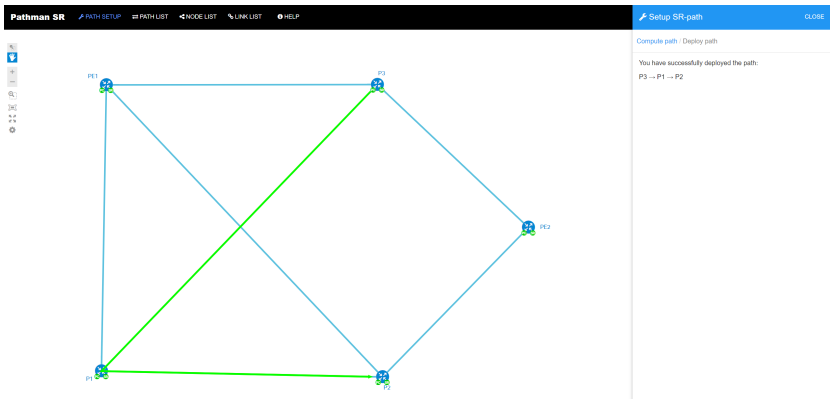


Figura 20: Instalación exitosa Slicing para Downlink UE1

Se verifica también los logs de Pathman-SR con los resultados mostrados a continuación:

```
root:initialize   INFO: Init 2 done - debug saved
root:rest_interface_parser INFO: Commands Relieved: {u'path': [u'P3',
u'P1', u'P2'], u'option': u'create', u'name': u'Slicing UPF1'}
root:get_loop_list INFO: Path: [u'P3', u'P1', u'P2']
root:get_loop_list INFO: Loop list: [u'192.168.99.2', u'192.168.99.4']
root:get_sid_list INFO: Path: [u'P3', u'P1', u'P2']
root:get_sid_list INFO: SID list: [16001, 16003]
root:createSRtunnel INFO: Create SR Tunnel response: {u'output': {}}
root:post         INFO: {"response": [{"option": "create", "success":
true, "name": "Slicing UPF1"}]}
```

Lo más significativo es en la línea 7, donde se observa los segmentos correspondientes a las rutas configuradas, en la línea 8 se muestra cómo Pathman-SR manda la orden de instalar el túnel, a lo que obtiene una respuesta con el objeto success como true, lo cual indica que se ha configurado correctamente. En este momento ya está creado el túnel. En la Figura 21 se muestra el túnel creado en el router P3.

```
Name: tunnel-te101  Destination: 192.168.99.4  Ifhandle:0xb0 (auto-tunnel pcc)
  Signalled-Name: Slicing UPF1
  Status:
    Admin:    up Oper:   up   Path:  valid   Signalling: connected

    path option 10, (Segment-Routing) type explicit (autopcc_te101) (Basis for Se
      Protected-by PO index: 20
    G-PID: 0x0800 (derived from egress interface properties)
    Bandwidth Requested: 0 kbps  CT0
    Creation Time: Fri Aug 25 21:35:02 2023 (00:01:41 ago)
  Config Parameters:
    Bandwidth:        0 kbps (CT0) Priority:  7  7 Affinity: 0x0/0xffff
    Metric Type: TE (global)
    Path Selection:
      Tiebreaker: Min-fill (default)
      Protection: any (default)
    Hop-limit: disabled
    Cost-limit: disabled
    Path-invalidation timeout: 10000 msec (default), Action: Tear (default)
    AutoRoute: disabled  LockDown: disabled   Policy class: not set
    Forward class: 0 (default)
    Forwarding-Adjacency: disabled
    Autoroute Destinations: 0
    Loadshare:          0 equal loadshares
    Auto-bw: disabled
    Path Protection: Not Enabled
    BFD Fast Detection: Disabled
    Reoptimization after affinity failure: Enabled
    SRLG discovery: Disabled
  Auto PCC:
    Symbolic name: Slicing UPF1
    PCEP ID: 102
    Delegated to: 172.23.29.120
    Created by: 172.23.29.120
  History:
    Tunnel has been up for: 00:01:41 (since Fri Aug 25 21:35:02 UTC 2023)
    Current LSP:
      Uptime: 00:01:41 (since Fri Aug 25 21:35:02 UTC 2023)

  Segment-Routing Path Info (PCE controlled)
    Segment0[Node]: 192.168.99.2, Label: 16001
    Segment1[Node]: 192.168.99.4, Label: 16003
Displayed 1 (of 1) heads, 0 (of 0) midpoints, 0 (of 0) tails
Displayed 1 up, 0 down, 0 recovering, 0 recovered heads
```

Figura 21: Instalación Slicing para Downlink UE1

Se observa en la primera parte los datos generales del túnel, entre los que se destaca que no hay fijado un ancho de banda, por defecto es 0 kbps lo que indica que el BW usado por UE1 en Downlink dependerá de lo que tenga disponible el enlace. En la parte intermedia se observan todas las capacidades que se puede indicar en el túnel, como Path Protection, BFD, Policy Class, etc. Estas capacidades son muy interesantes ya que permiten añadir características de alta disponibilidad para el túnel y QoS. En la parte final se observa que el túnel ha sido creado por el PCE con la IP(172.23.29.120) que hace referencia a ODL e indica que su gestión está delegada al mismo, lo que quiere decir que desde el router no se puede modificar este túnel, solamente lo hará el controlador. La última parte hace referencia a la ruta programada en forma de segmentos. En la Figura 22 se muestra una

captura de Wireshark en la que se puede ver la información que se transmite mediante PCEP.

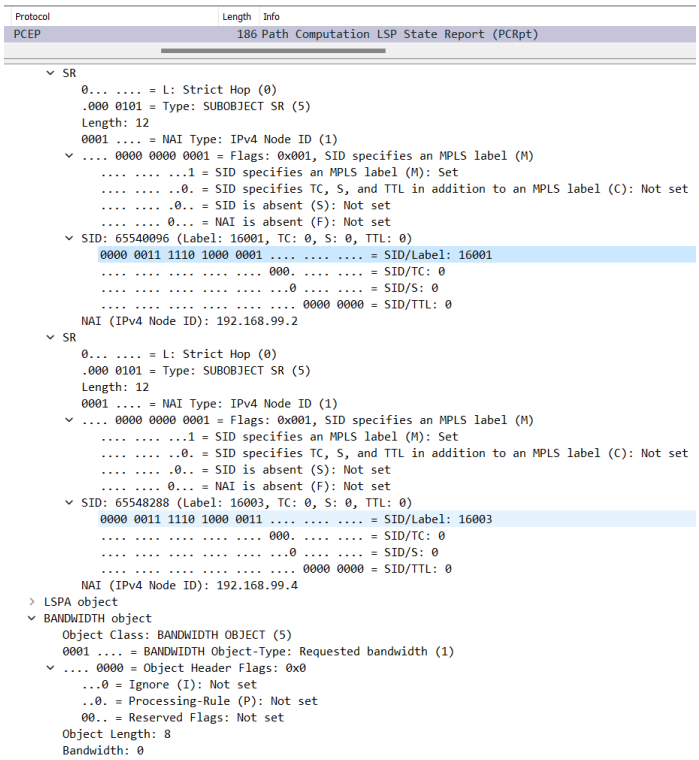


Figura 22: Instalación Slicing para Downlink UE1

Todo ha funcionado correctamente y se procede a crear los siguientes túneles:

Slice	Ruta
Uplink UE1	P2 − > P1 − > P3
Downlink UE1	P3 − > PE1 − > P2
Uplink UE2	P2 − > PE2 − > P3 − > P1
Downlink UE2	P1 − > P3 − > PE1 − > P2

Tabla 6: Túneles para Network Slicing

6.2. Modificación de los túneles

Cómo se ha podido observar en la Figura 21, los túeneles no se configuran con un Bandwidth específico ya que en la aplicación de Pathman-SR no existe esta opción, sin embargo es algo que se puede realizar y para ello se usa Postman, con esta aplicación se va a configurar un ancho de banda específico para el Uplink de 500 kbps y para el Downlink de 2Mbps para el acceso a la red 5G de los dispositivos UE2. En la Figura 23 se puede observar el túnel para el Uplink y en la Figura 24 el Downlink, con el respectivo ancho de banda configurado.

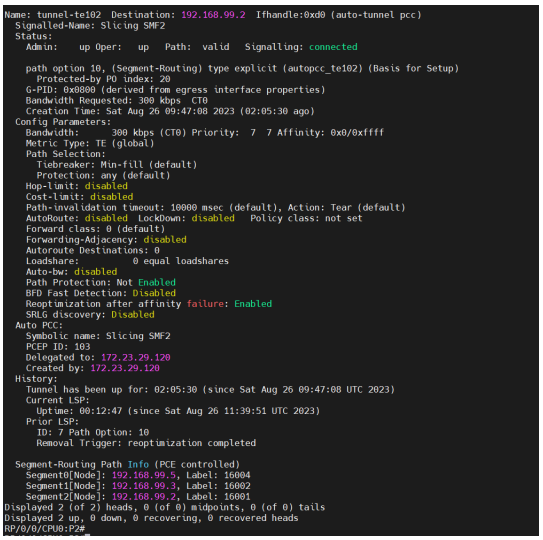


Figura 23: Slice para Uplink UE2

Se puede observar en ambas figuras, que el parámetro Bandwidth Requested ya no es cero y tienen configurado el que se ha indicado.

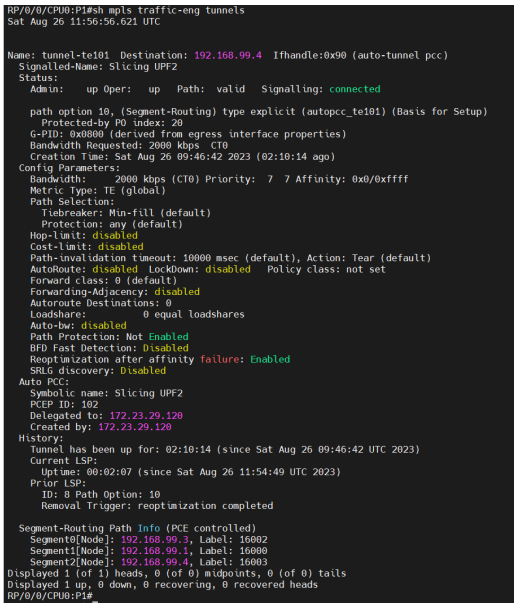


Figura 24: Slice para Downlink UE2

El contenido de la API para modificar el ancho de banda es el siguiente:

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:topology:pcep">
    <node>pcc://192.168.99.2</node>
    <name>Slicing UPF2</name>
    <arguments>
        <bandwidth>
            <ignore>false</ignore>
            <processing-rule>false</processing-rule>
```

```
            <bandwidth>SHQkAA==</bandwidth>
        </bandwidth>
    </arguments>
</input>
```

El valor de bandwidth está en formato bit64, RxJ8AA== corresponde a 0.3 Mbps y SHQkAA== corresponde a 2Mbps.

6.3. Uso de los slices por UE

En primer lugar es necesario añadir los Slices en el core 5G, para ello se modifica el AMf añadiendo las siguientes líneas a la configuración mostrada anteriormente.

```
s_nssai:
  - sst: 1
    sd: 000001
  - sst: 1
    sd: 000002
```

Una vez configurados en el AMF, es necesario configurar los slices en el NSSF el se encarga de indicar en qué NRF se debe buscar al SMF que sirve a cada Slice, en este caso, hay solo un NRF por lo tanto en el NSSF se indica a los dos slices que el SMF los pregunte al mismo NRF. Con el NSSF configurado, se configura el SMF-1 y SMF-2 para atender a cada slice.

En los SMF se va a indicar la red móvil la cuál se va a entregar cuando la sesión PDU se levante entre el UPF y el móvil, en la Figura 12 se observan las redes móviles de cada UPF. También se indica el identificador del Slice denominado SD.

Una vez configurado Network Slicing en el Core 5G se procede a levantar el gNB y que se registre en el AMF, el registro se realizó exitosamente como se observa en la Figura 25.

```
^Croot@UERANSIM-2:/ueransim# ./nr-ue -c config/open5gs-ue.yaml
UERANSIM v3.2.6
[2023-08-24 22:06:17.267] [nas] [info] UE switches to state [MM-DEREGISTERED/PLMN-SEARCH]
[2023-08-24 22:06:17.270] [rrc] [debug] New signal detected for cell[1], total [1] cells in coverage
[2023-08-24 22:06:17.271] [nas] [info] Selected plmn[001/01]
[2023-08-24 22:06:17.272] [rrc] [info] Selected cell plmn[001/01] tac[1] category[SUITABLE]
[2023-08-24 22:06:17.272] [nas] [info] UE switches to state [MM-DEREGISTERED/PS]
[2023-08-24 22:06:17.272] [nas] [info] UE switches to state [MM-DEREGISTERED/NORMAL-SERVICE]
[2023-08-24 22:06:17.272] [nas] [debug] Initial registration required due to [MM-DEREG-NORMAL-SERVICE]
[2023-08-24 22:06:17.272] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_sig]
[2023-08-24 22:06:17.272] [nas] [debug] Sending Initial Registration
[2023-08-24 22:06:17.272] [nas] [info] UE switches to state [MM-REGISTER-INITIATED]
[2023-08-24 22:06:17.272] [rrc] [debug] Sending RRC Setup Request
[2023-08-24 22:06:17.274] [rrc] [info] RRC connection established
[2023-08-24 22:06:17.274] [rrc] [info] UE switches to state [RRC-CONNECTED]
[2023-08-24 22:06:17.274] [nas] [info] UE switches to state [CM-CONNECTED]
```

Figura 25: Log del gNB al registrarse correctamente en el AMF

Con la radio lista, se registra el móvil en la base de datos para que pueda autenticarse correctamente y se provisiona el Slice correspondiente, UE1 tendrá configurado el Slice SD 000001 y para UE2 se configura el Slice SD 000002. El registro de UE1 se muestra en la Figura 26.

```
[2023-08-24 22:06:17.283] [nas] [debug] Authentication Request received
[2023-08-24 22:06:17.289] [nas] [debug] Security Mode Command received
[2023-08-24 22:06:17.289] [nas] [debug] Selected integrity[2] ciphering[0]
[2023-08-24 22:06:17.305] [nas] [debug] Registration accept received
[2023-08-24 22:06:17.305] [nas] [info] UE switches to state [MM-REGISTERED/NORMAL-SERVICE]
[2023-08-24 22:06:17.305] [nas] [debug] Sending Registration Complete
[2023-08-24 22:06:17.306] [nas] [info] Initial Registration is successful
[2023-08-24 22:06:17.306] [nas] [debug] Sending PDU Session Establishment Request
[2023-08-24 22:06:17.306] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_sig]
[2023-08-24 22:06:17.527] [nas] [debug] Configuration Update Command received
[2023-08-24 22:06:17.555] [nas] [debug] PDU Session Establishment Accept received
[2023-08-24 22:06:17.555] [nas] [info] PDU Session establishment is successful PSI[1]
[2023-08-24 22:06:17.563] [app] [info] Connection setup for PDU session[1] is successful, TUN interface[uesimtun0, 10.45.0.2] is up.
```

Figura 26: Establecimiento de la sesión PDU para UE1

El dispositivo UE1 establece la sesión PDU con el UPF-2 por lo tanto obtiene una de las IPs del rango 10.45.0.0/16, como se observa en la Figura 26 obtiene la IP 10.45.0.2, por lo tanto podrá acceder a la red Privada 192.168.1.0/24 en la cual se encuentra un servidor web. En la figura 27 se observa que efectivamente UE1 ha creado una nueva interfaz virtual llamada uesimtun0 con la IP de la red móvil y que mediante esta interfaz se produce con éxito la conexión hacia el servidor en la IP 192.168.1.3. Esta conexión tiene un Slice a nivel de transporte con los anchos de banda limitados. UE2 establece la

```
/ueransim # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: uesimtun0: <POINTOPOINT,UP,LOWER_UP120> mtu 1400 qdisc fq_codel qlen 500
    link/[65534]
    inet 10.45.0.2/32 scope global uesimtun0
       valid_lft forever preferred_lft forever
    inet6 fe80::752b:c41:8395:19a8/64 scope link
       valid_lft forever preferred_lft forever
156: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel qlen 1000
    link/ether 36:cf:e6:4b:84:1d brd ff:ff:ff:ff:ff:ff
    inet 192.168.5.3/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::34cf:e6ff:fe4b:841d/64 scope link
       valid_lft forever preferred_lft forever
157: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel qlen 1000
    link/ether de:74:5b:e7:8d:76 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::dc74:5bff:fee7:8d76/64 scope link
       valid_lft forever preferred_lft forever
158: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel qlen 1000
    link/ether 7e:db:57:6c:47:cb brd ff:ff:ff:ff:ff:ff
    inet6 fe80::7cdb:57ff:fe6c:47cb/64 scope link
       valid_lft forever preferred_lft forever
159: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel qlen 1000
    link/ether da:26:e9:e8:7c:35 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::d826:e9ff:fee8:7c35/64 scope link
       valid_lft forever preferred_lft forever
/ueransim # ping 192.168.1.3 -I uesimtun0 -n
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: seq=0 ttl=63 time=4.436 ms
64 bytes from 192.168.1.3: seq=1 ttl=63 time=4.224 ms
64 bytes from 192.168.1.3: seq=2 ttl=63 time=4.403 ms
^C
--- 192.168.1.3 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 4.224/4.354/4.436 ms
```

Figura 27: Provisionamiento de la IP y Ping a la red privada

sesión PDU con el UPF-1 por lo tanto obtiene una de las IPs del rango 10.46.0.0/16, por lo tanto podrá tener conectividad a internet. Esta conexión no tiene ningún limite en el ancho de banda para acceder a internet.

En la Figura 28 se puede observar que el UE1 puede acceder al servidor Web, mostrandose el contenido del mismo al hacer un curl con dirección al servidor.

```
/ueransim # curl 192.168.1.3
<!DOCTYPE html>
<html>
<head>
<title>5G Network Slicing SR-MPLS</title>
</head>
<body style="background-color: #003c4f; color: white;">
<table border="0" cellspacing="0" cellpadding="0" align="center">
<tbody>
<tr>
<td style="text-align: center;"><img src="gns3.png" /></td>
</tr>
<tr>
<td>
<h1 style="text-align: center;">JAVIER LOZA - UPV GCM</h1>
</td>
</tr>
<tr>
<td>
<strong>WWW:</strong> Files located at /var/www/html<br /> <strong>FTP:</strong> You get to /root after logging
 in but you're not chrooted<br /><strong>TFTP:</strong> You can place your files in /tftpboot; upload is permit
ted<br /><strong>DHCP:</strong> The DHCP server is installed but not configured to avoid any possible conflict<
br /><strong>Syslog server &amp;amp; SNMP trap receiver:</strong> They both log to /var/log/syslog
</td>
</tr>
</tbody>
</table>
</body>
</html>
/ueransim #
```

Figura 28: Acceso al servidor Web desde UE1

En la figura 29 se observa la captura de tráfico del enlace entre P3 y PE1, en el cuál está el trafico de downlink de internet para el UE2.

```
151.101.134.132 10.46.0.2  GTP <TCP>    94 80 → 54044 [ACK] Seq=1771000 Ack=487 Win=64240 Len=1460 [TCP segment of a reassembled PDU]

> Frame 8871: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0
> Ethernet II, Src: 0c:c9:0c:2d:00:03 (0c:c9:0c:2d:00:03), Dst: 0c:c3:74:bc:00:02 (0c:c3:74:bc:00:02)
> MultiProtocol Label Switching Header, Label: 16000, Exp: 0, S: 0, TTL: 63
> MultiProtocol Label Switching Header, Label: 16003, Exp: 0, S: 1, TTL: 63
> Internet Protocol Version 4, Src: 172.23.27.112, Dst: 192.168.5.2
> User Datagram Protocol, Src Port: 2152, Dst Port: 2152
v GPRS Tunneling Protocol
  > Flags: 0x34
    Message Type: T-PDU (0xff)
    Length: 1508
    TEID: 0x00000002 (2)
    Next extension header type: PDU Session container (0x85)
  v Extension header (PDU Session container)
      Extension Header Length: 1
    v PDU Session Container
        0000 .... = PDU Type: DL PDU SESSION INFORMATION (0)
        .... 0000 = Spare: 0x0
        0... .... = Paging Policy Presence (PPP): Not Present
        .0.. .... = Reflective QoS Indicator (RQI): Not Present
        ..00 0001 = QoS Flow Identifier (QFI): 1
      Next extension header type: No more extension headers (0x00)
> Internet Protocol Version 4, Src: 151.101.134.132, Dst: 10.46.0.2
> Transmission Control Protocol, Src Port: 80, Dst Port: 54044, Seq: 1771000, Ack: 487, Len: 1460
```

Figura 29: Acceso a internet desde UE2

6.4. Comportamiento de la red SR-MPLS

Uno de los objetivos es determinar cómo responde la configuración del ancho de banda en los túneles SR-TE, para ello se ha instalado en el UE1 y en el servidor web la herramienta iPerf para medir la velocidad. Cabe recalcar que al ser una simulación, el troughput máximo no es tan elevado, se ha medido sin configurar ningún ancho de banda y el promedio obtenido es de 2.7 Mbps.

En la Figura 30 se puede observar el resultado del test para medir el ancho de banda, si bien la respuesta no es exacta en 2Mbps, es un buen resultado, por lo tanto mediante SR-TE se puede regular el ancho de banda disponible.

```
/ueransim # iperf3 -c 192.168.1.3 -i 1
Connecting to host 192.168.1.3, port 5201
[  5] local 10.45.0.2 port 42532 connected to 192.168.1.3 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   432 KBytes  1.94 Mbits/sec    1   1.32 KBytes
[  5]   1.00-2.00   sec   253 KBytes  2.07 Mbits/sec    5   32.9 KBytes
[  5]   2.00-3.00   sec   226 KBytes  2.04 Mbits/sec   10   25.0 KBytes
[  5]   3.00-4.00   sec   226 KBytes  2.03 Mbits/sec   15   21.1 KBytes
[  5]   4.00-5.00   sec   253 KBytes  2.07 Mbits/sec    6   18.4 KBytes
[  5]   5.00-6.00   sec   226 KBytes  2.03 Mbits/sec    1   1.32 KBytes
[  5]   6.00-7.00   sec   253 KBytes  2.07 Mbits/sec    6   21.1 KBytes
[  5]   7.00-8.00   sec   226 KBytes  2.04 Mbits/sec   10   15.8 KBytes
[  5]   8.00-9.00   sec   226 KBytes  2.03 Mbits/sec    1   1.32 KBytes
[  5]   9.00-10.00  sec   253 KBytes  2.07 Mbits/sec    5   21.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  2.03 MBytes  2.05 Mbits/sec   60             sender
[  5]   0.00-10.49  sec  1.96 MBytes  1.91 Mbits/sec                  receiver

iperf Done.
```

Figura 30: Resultado del test de ancho de banda para el Downlink en UE1

En la Figura 31 se observa el test para el Uplink de UE1 realizado desde el servidor Web y aunque es menos preciso que en el Downlink, también fija el ancho de banda con una respuesta aceptable, por lo tanto se confirma la usabilidad de SR-TE para Network Slicing end-to-end

```
root@WebServer:~# iperf3 -c 10.45.0.2 -i 1 -f k
Connecting to host 10.45.0.2, port 5201
[  5] local 192.168.1.3 port 58418 connected to 10.45.0.2 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  44.8 KBytes  316 Kbits/sec    0   32.9 KBytes
[  5]   1.00-2.00   sec  40.8 KBytes  302 Kbits/sec    0   35.5 KBytes
[  5]   2.00-3.00   sec  50.6 KBytes  298 Kbits/sec    0   35.5 KBytes
[  5]   3.00-4.00   sec  46.7 KBytes  303 Kbits/sec    0   35.5 KBytes
[  5]   4.00-5.00   sec  40.8 KBytes  316 Kbits/sec    0   32.9 KBytes
[  5]   5.00-6.00   sec  50.6 KBytes  296 Kbits/sec    0   36.9 KBytes
[  5]   6.00-7.00   sec  51.3 KBytes  316 Kbits/sec    0   36.9 KBytes
[  5]   7.00-8.00   sec  46.7 KBytes  313 Kbits/sec    0   36.9 KBytes
[  5]   8.00-9.00   sec  40.8 KBytes  311 Kbits/sec    0   32.9 KBytes
[  5]   9.00-10.00  sec  50.6 KBytes  323 Kbits/sec    0   39.5 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec   409 KBytes   307 Kbits/sec    0             sender
[  5]   0.00-10.08  sec   409 KBytes   302 Kbits/sec                  receiver

iperf Done.
```

Figura 31: Resultado del test de ancho de banda para el Uplink en UE1

También es interesante saber cómo responde la red SR cuando se cae uno de los enlaces que forma el túnel y debe conmutar a otro enlace para formar uno nuevo. En la Figura 32 se observar la respuesta al ping y en la que se puede observar que en la actualización del túnel hubo un 16 por ciento de pérdida de paquetes y una latencia de 15.4 ms. En

```
64 bytes from 8.8.8.8: seq=49 ttl=63 time=30.349 ms
64 bytes from 8.8.8.8: seq=50 ttl=63 time=31.728 ms
64 bytes from 8.8.8.8: seq=51 ttl=63 time=11.400 ms
64 bytes from 8.8.8.8: seq=52 ttl=63 time=10.978 ms
64 bytes from 8.8.8.8: seq=53 ttl=63 time=13.923 ms
64 bytes from 8.8.8.8: seq=54 ttl=63 time=10.947 ms
64 bytes from 8.8.8.8: seq=55 ttl=63 time=10.876 ms
64 bytes from 8.8.8.8: seq=56 ttl=63 time=11.159 ms
64 bytes from 8.8.8.8: seq=57 ttl=63 time=10.883 ms
64 bytes from 8.8.8.8: seq=58 ttl=63 time=10.295 ms
64 bytes from 8.8.8.8: seq=59 ttl=63 time=10.397 ms
64 bytes from 8.8.8.8: seq=60 ttl=63 time=11.822 ms
64 bytes from 8.8.8.8: seq=61 ttl=63 time=17.814 ms
64 bytes from 8.8.8.8: seq=62 ttl=63 time=11.085 ms
64 bytes from 8.8.8.8: seq=63 ttl=63 time=10.933 ms
^C
--- 8.8.8.8 ping statistics ---
77 packets transmitted, 64 packets received, 16% packet loss
round-trip min/avg/max = 9.502/15.375/46.745 ms
/ueransim/config # ping 8.8.8.8 -I uesimtun0
```

Figura 32: Pérdida de paquetes al caerse uno de los enlaces

la Figura 33 se muestra la captura de paquetes en el momento exacto en el que debe conmutar al nuevo túnel, ya que se ha preparado la simulación para que en lugar de tener el uplink y el downlink por túneles separados, se caiga el enlace y el nuevo se uplink se levante por los mismos enlaces que el downlink.

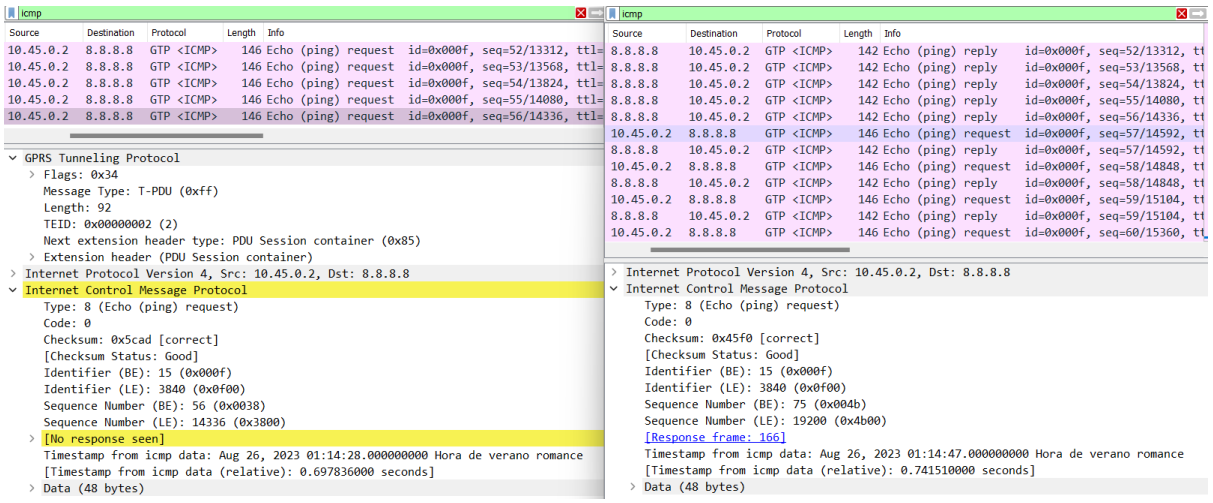


Figura 33: Captura de paquetes al caerse uno de los enlaces

6.5. Análisis de los resultados obtenidos

En base a los resultados obtenidos en los puntos anteriores se puede determinar lo siguiente:

- Segment Routing permite la programabilidad de la red, dependiendo de las características que se quiera en el túnel, se puede usar herramientas open source ya

existentes como Pathman-SR o se debe trabajar a nivel de APIs con lo complejo que se puede volver manejar el contenido de una API a medida que los campos aumentan.

- SR es un gran aliado para redes densas en las que la tarea de configurar lo necesario en el router para que funcione se puede hacer larga y tediosa lo que puede conllevar en errores ya que dependerá del operador de red, los comandos que necesite, conocimientos técnicos necesarios, configuración mediante CLI o en su defecto adquirir software propietario lo cual tiene un coste considerable y no entra dentro de la filosofía de 5G para redes abiertas. Mientras que con SDN, se reduce a conocer la red y en unos cuantos clicks se despliega el Slice de transporte en toda la red.
- En redes de dos o tres nodos que geográficamente se encuentren en el mismo lugar y no se conecten vía internet (BGP/MPLS), es decir, lo hagan mediante una red privada, es recomendable desplegar túneles L2/L3 VPN, que si bien no son dinámicos para este tipo de redes es más eficiente que desplegar un protocolo IGP (IS-IS u OSPF), luego MPLS y entonces SR. Sería recomendable SR en redes pequeñas si el despliegue IP es IPv6, ya que SRv6 no necesita de MPLS, se encapsula los segmentos en la cabecera de IPv6.
- SR por sí solo es eficiente y ofrece capacidades fundamentales para satisfacer el requirimiento que tiene Network Slicing, sin embargo, dependiendo del caso de uso, se puede apoyar en protocolos que son extensiones definidas por la IETF para SR, como BFD, TI-LFA, Flex-Algo. Estos protocolos mejoran la respuesta a caídas de enlaces, protección contra fallos, generación de lazos, baja latencia, entre otras capacidades.

7. Trabajos futuros

Vistas las capacidades que permite SDN, se puede trabajar en una aplicación que consuma los datos del controlador Opendaylight para configurar Network Slicing en la capa de transporte, incluso diseñar una aplicación que se pueda comunicar tanto con un NEF como con ODL, ya que ambos soportan API REST. Y así facilitar la configuración de Slices en el Core 5G y Red de Transporte.

En los próximos meses el grupo de comunicaciones móviles de la UPV, tendrá disponibles dos nodos 5G, por lo que uno de los objetivos cuando la red esté disponible, es desplegar una solución de SDN y Network Slicing. Por lo que el presente trabajo puede brindar una base técnica y teórica para el futuro despliegue 5G en la UPV.

8. Conclusiones

Una vez realizado el presente trabajo, se puede concluir que SDN es uno de las características principales a tener en cuenta para un despliegue 5G ya que su integración permite el desarrollo de capacidades 5G. Dentro de los controladores TeraFlowSDN tiene un potencial muy alto para ser el encargado de desplegar Network Slicing e-2-e, sin

embargo a día de hoy sus capacidades están limitadas y no cuenta con todo el soporte para equipos de red reales. Por ello Opendaylight es el controlador que mejor se adapta a las redes legacy que actualmente tienen desplegados los operadores de red y que permite como se ha visto, desplegar Network Slicing en red de transporte y permite la integración a aplicaciones para desplegar Slicing end-to-end.

En la red de transporte existen ifinidad de posibles soluciones dependiendo del despliegue que se pretenda hacer y el alcance que se requiera. Sin embargo Segment Routing es el protocolo que permite la programabilidad de las redes y ofrece un mejor rendimiento frente a otras soluciones basadas en enrutamiento tradicional.

Network Slicing es una de las capacidades que las operadoras pretenden ofrecer al mercado para monetizar su red gracias a 5G, el enfoque de redes abiertas enmarcado en el proyecto de Opengateway y la búsqueda de los operadores, vendors y toda la industria para ahorrar costes, hacen que las soluciones open source que permitan el despliegue 5G tengan más relevancia, por ello se ha demostrado en el presente trabajo que Opendaylight, Segment Routing y Open5GS permiten un despliegue enfocado hacia la evolución de las capacidades 5G.

Referencias

[1] Alejandro Adamowicz. 5g non-stand alone vs. 5g stand alone: Esta es la diferencia.

[2] Heidi Adams. *Segment routing in the 5G era*. Juniper adn Omdia, 2020.

[3] Altran. *Why SDN/NFV will be key for the evolution of the mobile networks*. Altran, Madrid, 2018.

[4] 5G Americas. *5G at the Edge*. 5G Americas, United States, 2019.

[5] C. Filsfils, S. Previdi, L. Ginsberg, Cisco Systems, B. Decraene, S. Litkowski, Orange, R. Shakir, and Google. Segment routing architecture. *IETF RFC 8402*, 9256, 2018.

[6] Y. Mika L. Madhusanka, G. Andrei. *Software Defined Mobile Networks (SDMN)*. John Wiley, Oulu, 2015.

[7] Nicolás Larocca. Seguiremos con el despliegue 5g sa hasta 11 ciudades en 2023": Orange españa.

[8] H. Masaharu and S. Jaiver. Official document ng.127 - e2e network slicing architecture. *GSMA 5G Networks*, 127:10–14, 2021.

[9] Z. Michele. *NETWORK SLICING USE CASE REQUIREMENTS*. Kelvin Qin, London, 2018.

[10] Enrique Blanco Nadales. El 5g se basará en redes abiertas.

[11] NEC. Telefónica vivo launches 5g-ready ip transport network with cisco and nec to provide seamless connections across brazil.

[12] Europa Press. El gobierno lanza una nueva convocatoria del programa Único sectorial 5g con 10 m en ayudas.

[13] V. Ricard and G. Lluís. Controlling and monitoring of packet optical networks.

[14] S. Rommer, P. Headman, M. Olsson adn L. Frid, S. Sultana, and C. Mulligan. *5G CORE NETWORKS*. Mara Conner, London, 2020.

[15] I. Kashif S. Syed, O. Alexander. *A Network Architect's Guide to 5G*. Addison-Wesley's, Boston, 2022.

[16] Ruth Gamero Tinoco. Explorando la red del futuro en 5tonic: más allá del 5g.