



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y
TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y
TELECOMUNICACIONES

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENICIÓN DEL TÍTULO DE
INGENIEROS EN ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: EVALUACIÓN DEL DESEMPEÑO DE LAS REDES DEFINIDAS
POR SOFTWARE (SDN) EN UN AMBIENTE INALÁMBRICO DE
ROAMING Y COMPARACIÓN CON UNA INFRAESTRUCTURA DE RED
INALÁMBRICA WI-FI DE DATOS EN LA UNIVERSIDAD DE LAS
FUERZAS ARMADAS - ESPE.

AUTORES: CURAY VALDIVIEZO, JEFFERSON EDUARDO
NOBOA GAVILANES, EDGAR SEBASTIÁN

DIRECTOR: ING. ROMERO GALLARDO, CARLOS GABRIEL

SANGOLQUÍ

2019



DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, ***“EVALUACIÓN DEL DESEMPEÑO DE LAS REDES DEFINIDAS POR SOFTWARE (SDN) EN UN AMBIENTE INALÁMBRICO DE ROAMING Y COMPARACIÓN CON UNA INFRAESTRUCTURA DE RED INALÁMBRICA WI-FI DE DATOS EN LA UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE”*** fue realizado por los señores ***Curay Valdiviezo, Jefferson Eduardo y Noboa Gavilanes, Edgar Sebastián***, el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustenten públicamente.

Sangolquí, 15 de Julio de 2019



.....
Carlos Gabriel Romero Gallardo

C.C. 1712198066



DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES


AUTORÍA DE RESPONSABILIDAD

Nosotros, **Curay Valdiviezo, Jefferson Eduardo** y **Noboa Gavilanes, Edgar Sebastián**, declaramos que el contenido, ideas y criterios del trabajo de titulación ***“Evaluación del desempeño de las redes definidas por software (SDN) en un ambiente inalámbrico de roaming y comparación con una Infraestructura de red inalámbrica Wi-Fi de datos en la Universidad de las Fuerzas Armadas – ESPE”*** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente, el contenido de la investigación mencionada es veraz.

Sangolquí, 15 de Julio de 2019


.....
Jefferson Eduardo Curay Valdiviezo
C.C. 1721058897


.....
Edgar Sebastián Noboa Gavilanes
C.C. 1750236687



DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Nosotros, **Curay Valdiviezo, Jefferson Eduardo** y **Noboa Gavilanes, Edgar Sebastián**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: ***“Evaluación del desempeño de las redes definidas por software (SDN) en un ambiente inalámbrico de roaming y comparación con una infraestructura de red inalámbrica Wi-Fi de datos en la Universidad de las Fuerzas Armadas – ESPE”*** en el Repositorio Institucional cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 15 de Julio de 2019

Jefferson Eduardo Curay Valdiviezo

C.C. 1721058897

Edgar Sebastián Noboa Gavilanes

C.C. 1750236687

DEDICATORIA

A mis padres

Luis y Patricia quienes me han acompañado en esta etapa de mi vida con su amor, apoyo y motivación, por sus consejos y tiempo dedicado, por ser mi inspiración y modelo para seguir cada día.

A mis hermanos

Luis y Steven por formar parte de mi vida y siempre estar presentes a pesar de las adversidades o problemas que se puedan presentar, porque ustedes han sido mis guías y me han formado como persona.

Jefferson Curay

DEDICATORIA

A mis padres

Hugo y Gioconda, por el amor y apoyo incondicional durante toda mi vida, tanto en lo personal como lo académico, sin importar las diferencias que podamos tener.

A mis hermanos

Daniel y Hugo, por las risas y formar parte tan importante de mi crianza, para llevarme a donde hoy estoy.

Sebastián Noboa

AGRADECIMIENTOS

Me podrían faltar páginas para agradecer todas las personas que formaron parte de mi vida universitaria. Primero agradecer a mis padres por el sacrificio hecho para darme la oportunidad de continuar con mi vida profesional, por su amor y motivación para cada día continuar sin rendirme.

Asimismo, agradezco eternamente a mis hermanos por todo el apoyo brindado, por ser siempre las personas con las que puedo contar a pesar de las adversidades.

Quiero agradecer a mis amigos que estuvieron conmigo a lo largo de mi carrera universitaria, Diego Pacheco, Daniel Caiza, Victoria Yáñez, Estefanny Gavilema, Luis Tapia, Christian Pachacama, Alex Suquillo, Santiago Pazmiño, Dennis Córdoba, de manera especial quiero agradecer a Daniel Dominguez, Diego Guzmán, Alexis Estevez, Marcelo Ortiz y Fernando Lara por cada minuto compartido y soportarme a lo largo de los años y a Sebastián Noboa porque desde el semestre que nos conocimos ha sido mi amigo y compañero de trabajo hasta mi último trabajo universitario.

De igual forma agradecer a la Universidad de las Fuerzas Armadas por su gran labor estudiantil formando profesionales de categoría y a mis profesores que además de guiarme en el campo laboral han sido parte de mi vida brindándome su amistad.

Jefferson Curay

AGRADECIMIENTOS

Este proyecto, y haber llegado a este punto de mi vida, es algo que no se pudiera haber logrado sin el apoyo de estas personas.

Agradezco a mi familia por siempre motivarme a seguir mis sueños, y ayudarme a lograrlos, así como criarme con mucho amor.

A mi amiga Andrea, por todos los años de amistad incondicional, y haber estado ahí para mí en las buenas y en las malas.

Al estimado Ing. Carlos Romero, por toda su ayuda en este proyecto y los conocimientos impartidos antes de y durante el mismo.

Finalmente, a la Universidad de las Fuerzas Armadas y todos sus docentes por haber contribuido a mi formación como profesional y como persona.

Sebastián Noboa

ÍNDICE DE CONTENIDO

CERTIFICADO DEL DIRECTOR	i
AUTORÍA DE RESPONSABILIDAD.....	ii
AUTORIZACIÓN	iii
DEDICATORIA.....	iv
AGRADECIMIENTOS	vi
ÍNDICE DE CONTENIDO	viii
ÍNDICE DE FIGURAS	x
ÍNDICE DE TABLAS	xiii
RESUMEN.....	xiv
ABSTRACT	xv
CAPÍTULO I	
ANTECEDENTES Y JUSTIFICACIÓN	
1.1. Antecedentes	1
1.2. Justificación	3
1.3. Objetivos	5
1.3.1. Objetivo General	5
1.3.2. Objetivos específicos	5
CAPÍTULO II	
FUNDAMENTOS TEÓRICOS	
2.1. Redes Definidas por Software	7
2.2. Estándares y Normas de la arquitectura SDN	8
2.2.1. Open Networking Foundation	8
2.2.2. RFC 7426 – SDN: Layers and Architecture Terminology.....	12
2.3. OpenFlow.....	17
2.4. Controladores.....	32
2.4.1. Controlador Ryu.....	34
2.5. Mininet-WiFi	35
2.6. Servidor Radius.....	38
2.6.1. Autenticación, Autorización y Contabilidad	38
2.6.2. Funcionamiento	39
2.7. Redes Inalámbricas de Área Local (WLAN)	40
2.7.1. Seguridad en Redes Inalámbricas WLAN.....	42
2.7.2. Autenticación 802.1X (Seguridad Empresarial)	43
2.8. EDUROAM.....	46
CAPÍTULO III	
IMPLEMENTACIÓN	
3.1. Consideraciones para el diseño del Testbed.....	48
3.2. Hardware y Software utilizado	49
3.3. Simulación en Mininet-Wifi	53

3.3.1. Instalación de Mininet-wifi	53
3.3.2. Pruebas de funcionamiento	54
3.3.3. Python y Mininet-wifi	55
3.4. Implementación del Testbed	62
3.4.1. Topología de la red	63
3.4.2. Instalación y configuración de ryu y SimpleSwitch 2.0.....	64
3.4.3. Instalación y configuración de FreeRADIUS.....	66
3.4.4. Configuración de los Access Points.....	72
CAPÍTULO IV	
METODOLOGÍA DE PRUEBAS	
4.1. Descripción de los datos a ser obtenidos	78
4.2. Ubicación y configuración de potencia de los equipos	80
4.3. Descripción de las pruebas	90
4.3.1. Escenarios de pruebas	91
4.3.2. Parámetros de la inyección de tráfico	92
4.3.3. Configuración de la captura de paquetes	95
4.4. Desarrollo de las pruebas.....	97
CAPÍTULO V	
RESULTADOS	
5.1. Pruebas de red Wi-Fi en 2,4 GHz	103
5.1.1. Comparación de arquitecturas de red implementadas en 2.4 GHz.....	103
5.1.2. Comparación de arquitectura SDN implementada y red universitaria	108
5.2. Pruebas de red Wi-Fi 5 GHz	115
5.2.1. Comparación de arquitecturas de red implementadas en 5 GHz.....	115
5.2.2. Comparación de arquitectura SDN implementada y red universitaria.....	121
CAPÍTULO VI	
CONCLUSIONES Y RECOMENDACIONES	
6.1. Conclusiones.....	128
6.2. Recomendaciones.....	130
6.3. Futuros estudios.....	132
REFERENCIAS	133

ÍNDICE DE FIGURAS

Figura 1 Modelo Básico.	9
Figura 2 Núcleo de la arquitectura SDN.	11
Figura 3 Arquitectura SDN del RFC 7426.	13
Figura 4 Componentes de un switch OpenFlow.	18
Figura 5 Diagrama de flujo del tratamiento de paquetes.	20
Figura 6 Flujo de paquetes a través del procesamiento de canalización.	22
Figura 7 Componentes de Mininet-Wifi.	36
Figura 8 Componentes y conexiones en una red simulada de dos hosts en Mininet-Wifi.	37
Figura 9 Proceso de autenticación y autorización del servicio Radius.	39
Figura 10 Arquitectura general de red EDUROAM.	47
Figura 11 Access Point Zodiac WX.	50
Figura 12 Tarjeta de Captura AirPcap Nx.	51
Figura 13 Prueba de funcionamiento de red simple en Mininet-wifi.	55
Figura 14 Inicio de controlador ryu para simulación.	60
Figura 15 Simulación gráfica de red con estaciones y puntos de acceso.	61
Figura 16 Instalación de flujos en el controlador.	62
Figura 17 Pruebas de conectividad entre estaciones por el controlador.	62
Figura 18 Topología del Testbed.	63
Figura 19 Archivo defaults.cfg modificado.	65
Figura 20 Salida del CLI de ryu, en la cual se observa la instalación y desinstalación de flujos en los respectivos dispositivos de red.	66
Figura 21 Usuarios creados para el acceso a la red inalámbrica.	67
Figura 22 Clientes validados como suplicantes.	68
Figura 23 Información de autenticación para Eduroam proporcionada por el CA.	69
Figura 24 Configuración de TTLS en el archivo eap.	69
Figura 25 Mensaje de Access-Accept por parte del servidor Radius.	70
Figura 26 Configuración para la generación de certificados y CAs.	71
Figura 27 CA de producción.	72
Figura 28 Ventana de autorización para la configuración de un Zodiac WX.	73
Figura 29 Configuración de IP para las interfaces cableadas.	73
Figura 30 Script de inicialización de Open vSwitch.	74
Figura 31 Flujos OpenFlow instalados en el Zodiac WX.	75
Figura 32 Configuración de seguridad inalámbrica para la interfaz de 5 GHz.	76
Figura 33 Autenticación exitosa del usuario por parte de FreeRadius.	77
Figura 34 Instalación de nuevos flujos por ryu dado el ingreso de un dispositivo	

nuevo a la red.....	77
Figura 35 Paquetes 802.11 capturados con las tarjetas AirPcap NX.	80
Figura 36 Acrylic Wi-Fi Heatmaps realizando un survey.	82
Figura 37 Mapa de calor para el SSID Eduroam en 2.4 GHz.	83
Figura 38 Filtrado de direcciones MAC en 2.4 GHz de los APs relevantes y escala de potencia recibida en dBm.	84
Figura 39 Mapa de calor para el SSID Eduroam en 5 GHz.	85
Figura 40 Filtrado de direcciones MAC en 5 GHz de los APs relevantes	85
Figura 41 Mapa de calor para la red inalámbrica del Testbed en 2.4 GHz.	87
Figura 42 Mapa de calor para la red inalámbrica del Testbed en 2.4 GHz.	88
Figura 43 Izquierda: Mapa de calor para la red Eduroam en 2.4 GHz. Derecha: Mapa de calor para el Testbed en 2.4 GHz.	89
Figura 44 Izquierda: Mapa de calor para la red Eduroam en 5 GHz. Derecha: Mapa de calor para el Testbed en 5 GHz.	90
Figura 45 Parámetros configurados en D-ITG.	94
Figura 46 Panel de control de AirPcap, donde se puede seleccionar el canal en el cual se capturará el tráfico.....	96
Figura 47 Wireshark mostrando parte del tráfico capturado en una de las pruebas.....	97
Figura 48 Recorrido para las pruebas de roaming entre los Access Points.	98
Figura 49 Variación de potencia recibida durante movimiento del cliente inalámbrico en 2.4 GHz.	99
Figura 50 Variación de potencia recibida durante movimiento del cliente inalámbrico en 5 GHz.	101
Figura 51 Resultados de comparación en delay de redes implementadas en 2.4 GHz.	104
Figura 52 Resultados de comparación en jitter de redes implementadas en 2.4 GHz.	105
Figura 53 Resultados de comparación en bitrate de redes implementadas en 2.4 GHz.	106
Figura 54 Resultados de comparación en pp de redes implementadas en 2.4 GHz. .	107
Figura 55 Resultados de comparación en ta de redes implementadas en 2.4 GHz. ..	108
Figura 56 Resultados de comparación en delay con red universitaria en 2.4GHz.....	109
Figura 57 Resultados de comparación en jitter con red universitaria en 2.4GHz.	110
Figura 58 Resultados de comparación en bitrate con red universitaria en 2.4GHz. ...	112
Figura 59 Resultados de comparación en pp con red universitaria en 2.4GHz.	113
Figura 60 Resultados de comparación en ta con red universitaria en 2.4GHz.	114
Figura 61 Resultados de comparación en tr con red universitaria en 2.4GHz.	115
Figura 62 Resultados de comparación en delay de redes implementadas en 5 GHz.	116
Figura 63 Resultados de comparación en jitter de redes implementadas en 5 GHz. .	117
Figura 64 Resultados de comparación en bitrate de redes implementadas en 5 GHz.	118

Figura 65	Resultados de comparación en pp de redes implementadas en 5 GHz.	119
Figura 66	Resultados de comparación en ta de redes implementadas en 5 GHz.	120
Figura 67	Resultados de comparación en delay con red universitaria en 2.4GHz.....	122
Figura 68	Resultados de comparación en jitter con red universitaria en 5 GHz.	123
Figura 69	Resultados de comparación en bitrate con red universitaria en 5 GHz.	124
Figura 70	Resultados de comparación en pp con red universitaria en 5 GHz.	125
Figura 71	Resultados de comparación en ta con red universitaria en 5 GHz.	126
Figura 72	Resultados de comparación en tr con red universitaria en 5 GHz.	127

ÍNDICE DE TABLAS

Tabla 1	<i>Comparación de controladores SDN.....</i>	33
Tabla 2	<i>Escenarios de pruebas del Testbed.....</i>	92
Tabla 3	<i>Parámetros más relevantes configurados en la inyección de tráfico.....</i>	94
Tabla 4	<i>Comparación de redes implementadas en parámetro: Delay en 2.4 GHz</i>	103
Tabla 5	<i>Comparación de redes implementadas en parámetro: Jitter en 2.4 GHz.....</i>	104
Tabla 6	<i>Comparación de redes implementadas en parámetro: Bitrate en 2.4 GHz ...</i>	105
Tabla 7	<i>Comparación de redes implementadas en parámetro: pp en 2.4 GHz.....</i>	106
Tabla 8	<i>Comparación de redes implementadas en parámetro: ta en 2.4 GHz</i>	108
Tabla 9	<i>Comparación con red universitaria en parámetro: Delay en 2.4 GHz</i>	109
Tabla 10	<i>Comparación con red universitaria en parámetro: Jitter en 2.4 GHz.....</i>	110
Tabla 11	<i>Comparación con red universitaria en parámetro: Bitrate en 2.4 GHz</i>	111
Tabla 12	<i>Comparación con red universitaria en parámetro: pp en 2.4 GHz</i>	112
Tabla 13	<i>Comparación con red universitaria en parámetro: ta en 2.4 GHz</i>	113
Tabla 14	<i>Comparación con red universitaria en parámetro: tr en 2.4 GHz</i>	114
Tabla 15	<i>Comparación de redes implementadas en parámetro: Delay en 5 GHz</i>	116
Tabla 16	<i>Comparación de redes implementadas en parámetro: Jitter en 5 GHz.....</i>	117
Tabla 17	<i>Comparación de redes implementadas en parámetro: Bitrate en 5 GHz</i>	118
Tabla 18	<i>Comparación de redes implementadas en parámetro: pp en 5 GHz.....</i>	119
Tabla 19	<i>Comparación de redes implementadas en parámetro: ta en 5 GHz.....</i>	120
Tabla 20	<i>Comparación con red universitaria en parámetro: Delay en 5 GHz</i>	121
Tabla 21	<i>Comparación con red universitaria en parámetro: Jitter en 5 GHz.....</i>	122
Tabla 22	<i>Comparación con red universitaria en parámetro: Bitrate en 5 GHz.....</i>	124
Tabla 23	<i>Comparación con red universitaria en parámetro: pp en 5 GHz</i>	125
Tabla 24	<i>Comparación con red universitaria en parámetro: ta en 5 GHz</i>	126
Tabla 25	<i>Comparación con red universitaria en parámetro: tr en 5 GHz</i>	127

RESUMEN

El propósito de este proyecto es el diseño e implementación de un escenario de pruebas en cual se pueda estudiar la utilidad de las redes definidas por software, mediante el protocolo OpenFlow, para intentar mejorar el tiempo de roaming y parámetros de calidad de servicio asociados al roaming en redes inalámbricas basadas en Wi-Fi. Se implementó un Testbed con cuatro escenarios, en los cuales mediante el uso de inyecciones de tráfico se pudo caracterizar la red trabajando en un ambiente de roaming en términos de parámetros relevantes como el tiempo de roaming, tiempo de autenticación, delay, jitter, bitrate y pérdida de paquetes; realizando las respectivas comparativas entre los diferentes escenarios. Además de esto se comparó el desempeño del Testbed con el de una porción de la red inalámbrica actualmente implementada en la Universidad de las Fuerzas Armadas – ESPE, teniendo sumo cuidado en replicar esta con la mayor fidelidad posible. Mediante el análisis de los datos obtenidos durante las pruebas se pudo observar resultados prometedores, significativamente mejorando el tiempo de roaming y ciertas métricas como delay y pérdida de paquetes respecto a la red de la Universidad, mientras que otras métricas como el tiempo de autenticación mostraron un peor desempeño en la implementación con OpenFlow.

PALABRAS CLAVE:

- **SDN:** Redes Definidas por Software
- **OPENFLOW:** Protocolo abierto de SDN
- **ROAMING:** Movilidad inalámbrica
- **WI-FI:** Estándar de redes inalámbricas
- **RYU:** Controlador OpenFlow

ABSTRACT

The purpose of this project is the design and implementation of a Testbed in which the usefulness of software defined networking, by means of the OpenFlow protocol, can be studied, in order to try and improve roaming time and parameters associated with quality of service in Wi-Fi based wireless networks. A Testbed with four different scenarios was implemented, in which through the use of traffic injections we could characterize the network as it works in a roaming environment in terms of roaming time, authentication time, delay, jitter, bitrate and packet loss; drawing comparisons between the different scenarios. Besides this we benchmarked the Testbed against a section of the wireless network currently operating at Universidad de las Fuerzas Armadas – ESPE, taking great care in replicating this network with as much fidelity as possible. Through the analysis of obtained data during the testing phase we were able to observe promising results, significantly increasing roaming time and certain metrics such as delay and packet loss when compared to the University network, while other metrics such as authentication time showed worse performance on the OpenFlow implementation.

KEYWORDS:

- **SDN:** Software Defined Networks
- **OPENFLOW:** SDN open protocol
- **ROAMING:** Wireless Mobility
- **WI-FI:** Wireless network standard
- **RYU:** OpenFlow Controller

CAPÍTULO I: ANTECEDENTES Y JUSTIFICACIÓN

1.1. Antecedentes

La arquitectura de red tradicional no ha cambiado en los últimos años, sin embargo, el enfoque del Internet y los servicios que puede ofrecer un dominio interno han cambiado progresivamente.

La idea del cambio de paradigma y optar por redes programables no es nueva, las contribuciones nacen bajo la idea del proyecto SOFTNET. La red SOFTNET cuenta con una arquitectura de red multisalto, en la cual el campo de datos de cada paquete era incluido mediante comandos que los diferentes nodos iban estableciendo cuando los recibían, permitiendo que la red sea modificable y actúe de forma dinámica en tiempo real (Zander & Forchheimer, 1988).

Impulsada por la idea de SOFTNET se crean las conocidas como Redes Activas, las cuales permitían la programabilidad de la red a través de una interfaz. Esta arquitectura de red contaba con pequeños bloques de código en los paquetes transmitidos, los cuales eran ejecutados por los nodos de comunicación intermedia. (University of Kentucky, 1999)

Tanto SOFTNET como las Redes Activas presentan nuevas e innovadoras formas sobre la arquitectura de redes, con una idea dinámica y basada en los datos que transportaban. Su mecanismo era similar ya que, añaden líneas de código a la estructura del paquete de datos y permitían que los nodos de comunicación intermedia los

ejecutaran según el caso. Sin embargo, no incorporan un elemento de software de control de conmutación como lo establecen las Redes Definidas por Software (SDN - Software Defined Networking).

A principios de los años 2000, las redes seguían en un crecimiento acelerado debido a la facilidad de conectividad a Internet con la implementación de protocolos como ADSL (Asymmetric Digital Subscriber Line), permitiendo el uso de diversos servicios como: correo electrónico, teleconferencia o entretenimiento multimedia. Para finales de los años 2000 la complejidad de las redes aumento y era necesario contar con servicios más dinámicos, debido a la implementación de mecanismos de control como ACLs (listas de control de acceso), cortafuegos, ingeniería de tráfico o VLANs (Virtual Local Area Network).

En el año 2007 se desarrolla OpenFlow a través de la empresa Nicira por parte de profesores y doctores de la Universidad de Standford y a partir de esta idea se establece el nacimiento de las SDN. Posteriormente se funda la ONF (Open Networking Foundation), organización cuyo objetivo era fomentar el uso de OpenFlow como protocolo de comunicación de código abierto e implementar redes SDN más allá de las Universidades. En la actualidad Nicira fue adquirida por la empresa VmWare (compañía líder en virtualización de servidores) dando un paso para la implementación de virtualización de redes en su gama de productos.

1.2. Justificación

La gestión de redes tiene el objetivo de planificar, organizar y controlar los dispositivos de su arquitectura, con el fin de garantizar sus servicios, logrando así que la red esté disponible el mayor tiempo posible mediante la corrección y detección de fallos. La administración y configuración de los equipos intermedios de red son realizados usualmente de manera individual y por medio de interfaces gráficas o línea de comandos. La comunicación con los equipos varía entre las diferentes marcas e incluso de productos del mismo proveedor por motivos de líneas comerciales.

En los últimos años el avance en la gestión y almacenamiento de la información con la aparición de nuevas tecnologías como cloud computing, mayor presencia de data centers, permitieron incrementar a las redes en tamaño y velocidad, sin embargo, el proceso de enrutamiento sigue presentando fallas, haciendo necesario implementar un plano de control y su interacción con el plano de datos. Otro de los problemas que se presentan en las redes tradicionales es su gestión en el volumen de tráfico, hoy en día los dispositivos inalámbricos cuentan con una gran variedad de tecnologías como GPS, contenidos multimedia en alta definición, telefonía IP, sensores electrónicos, además de la necesidad de movilidad, seguridad informática y Big Data. Debido a estos grandes cambios de las redes tradicionales es necesario que toda organización con gran influencia en TICs (Tecnologías de la Información y la Comunicación) implementen nuevas soluciones que refuercen su funcionamiento (Yáñez & Gallegos, Implementación de un prototipo de Red Definida por Software para el Hotspot-Espeak mediante un controlador basado en OpenFlow, 2016). Las redes definidas por software buscan implementar con

mayor facilidad un servicio de red, de forma eficiente, escalable y económica. Esta interfaz programada nos permite evitar las diferentes interfaces de administración dependiendo de la marca o línea comercial de los dispositivos de red, ya que todos trabajaran bajo la directiva de OpenFlow, cuyas instrucciones de funcionamiento son proporcionadas por los controladores en un lenguaje universal, en lugar de múltiples protocolos de dispositivos de fabricantes específicos (Guano, Prototipo de una SDN utilizando herramientas Open-source, 2017).

En la actualidad las redes inalámbricas cuentan con nuevas tecnologías para realizar una gestión central, para ello existen los Wireless LAN Controller (WLC), pero la implementación de esta nueva tecnología elevaría los costos de la red, además que trabajaría exclusivamente para equipos de su misma marca o línea de trabajo, debido a que son compatibles en capas superiores del modelo OSI o a su vez necesitan de protocolos propietarios como en el caso del Lightweight Access Point Protocol (LWAPP) exclusivo para equipos de la marca CISCO (CISCO, 2016). Por el contrario, las SDN trabajan en capas inferiores, permitiendo que, los equipos acepten el protocolo OpenFlow de código abierto y ser gestionado por el controlador central (implementación con menor costo) de la red sin considerar la marca del equipo de enrutamiento. Se debe considerar que actualmente las redes inalámbricas Wi-Fi, deben ofrecer servicio a grandes áreas de cobertura sin perder conectividad y una de sus soluciones es con implementación de técnicas como roaming (Minghui, Xuemin, & Jon, 2004).

Se debe considerar que el desarrollo de estos proyectos en la Universidad de las Fuerzas Armadas – ESPE está pensado para el beneficio de esta, valiéndose de los

recursos de investigación que la Universidad posee y estableciendo una línea base para una posible futura actualización de la infraestructura de red utilizando SDN.

1.3. Objetivos

1.3.1. Objetivo General

Comparar el desempeño de la arquitectura de red inalámbrica de datos de la Universidad de las Fuerzas Armadas – ESPE (Laboratorios de electrónica) y una SDN implementada en el mismo escenario, en términos de parámetros de calidad de servicio de redes inalámbricas (roaming, jitter, delay, throughput y paquetes perdidos) y el tiempo de roaming.

1.3.2. Objetivos específicos

- Recopilar, estudiar y analizar información sobre la infraestructura actual de la red inalámbrica de la ESPE y trabajos desarrollados sobre SDN en escenarios de roaming.
- Implementar un escenario de pruebas de SDN basado en la infraestructura de la red inalámbrica de la ESPE y realiza pruebas de funcionamiento.
- Realiza pruebas comparativas de roaming y parámetros de calidad de servicio en la arquitectura de red de datos y la red SDN en el escenario establecido, utilizando los mismos equipos terminales de usuario.
- Recolectar y analizar datos comparativos en los escenarios de infraestructura de red: SDN, SDN con Radius, WPA (Wi-Fi Protected

Access) con Radius y WPA para conocer el comportamiento de las diferentes arquitecturas en el mismo escenario de red.

CAPÍTULO II: FUNDAMENTOS TEÓRICOS

2.1. Redes Definidas por Software

Las redes tradicionales no son óptimas para satisfacer los requerimientos de tráfico en la actualidad debido a tecnologías como: calidad de servicio, políticas de control de acceso, servicios en la nube, necesidad de elevados anchos de banda en dispositivos móviles o la virtualización de servicios, esto se debe a su estructura cerrada y estática, que no permite ser configurado en tiempo real bajo la demanda de aplicaciones. En ese contexto se necesita una solución que pueda transformar la arquitectura y gestión de las redes que se utilizan hoy en día. (García, Rodríguez, Calderón, & Casmartíño, 2014)

Las Redes Definidas por Software (Software Defined Networking - SDN) son arquitecturas de red en la cual el plano de control se desprende del plano de datos. Su objetivo es permitir a los administradores de red una gestión de servicios centralizada, pudiendo definir el comportamiento de la red desde un elemento central (Controlador SDN) con el software apropiado.

En una SDN, se evita la configuración individual de los equipos intermedios de red (switches o routers), permitiendo cambiar reglas para aceptar o bloquear tráfico en los dispositivos, decidir el reenvío de paquetes y permitir la construcción de tablas de enrutamiento. Por medio de estas funciones se facilita la implementación de servicios y aplicaciones de forma flexible, escalable, eficiente y económica, por sus características

programables y automatizables, ideales para altas demandas de ancho de banda y naturalezas de red dinámicas. (Serrano, 2015)

Una de las propuestas de la SDN es permitir la interoperabilidad en equipos de red, al habilitar una interfaz abierta entre equipos de terceros, sin importar su fabricante ni versiones de software, siempre y cuando cuenten con el protocolo de red que permita la comunicación entre equipos SDN, conocido como OpenFlow. De esta forma los equipos de una red SND no deben entender varios protocolos estándares, sino aceptar las instrucciones que envía el controlador SDN.

2.2. Estándares y Normas de la arquitectura SDN

Las normas y estándares son acuerdos documentados que especifican criterios o técnicas precisas que son utilizadas como guías para el desarrollo de equipos, productos, servicios o procesos, con el objetivo de que cumplan con su propósito establecido. Es por tanto necesario conocer los acuerdos documentales más relevantes que definen las SDN.

2.2.1. Open Networking Foundation

La Open Networking Foundation (ONF) es una de las instituciones que busca la estandarización de la arquitectura de las redes SDN evitando detalles tecnológicos. La ONF recomienda documentos que detallan potencialmente, la arquitectura en áreas específicas como:

- TR-522, SDN Architecture for Transport Networks (Open Networking Foundation, 2016).
- TR-518, Relationship of SDN and NFV (Open Networking Foundation, 2015).
- TR-523, Intent NBI – Definition and Principles (Open Networking Foundation, 2016).

La arquitectura descrita en el documento SDN Architecture (Open Networking Foundation, 2016), define un diseño que busca ser consistente, útil y abierto.

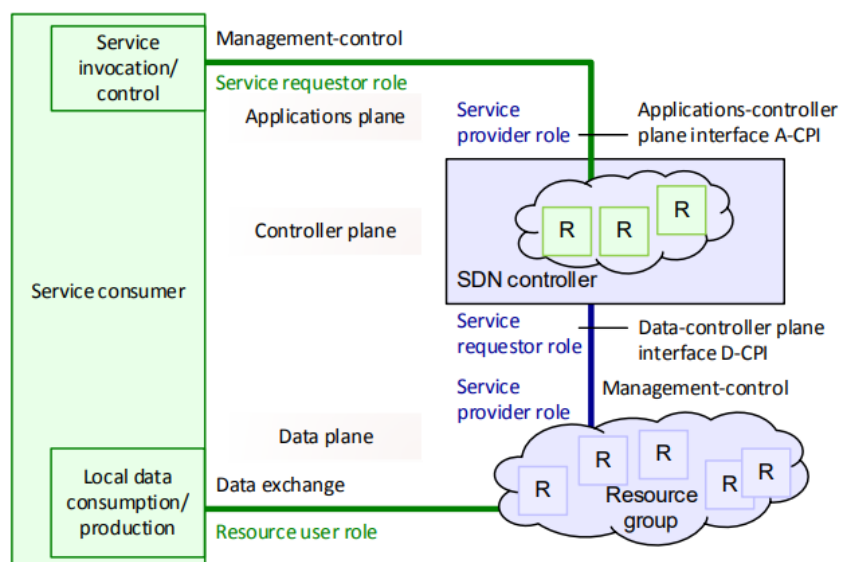


Figura 1 Modelo Básico.

Fuente: (Open Networking Foundation, 2016)

El modelo de la Figura 1 muestra un modelo básico de un cliente (cuadro en verde), que intercambia tráfico y operaciones de control de gestión de un proveedor SDN (cuadro en azul). Se puede observar que, aunque se utilizan recursos de envío R del cliente, el controlador es quien administra su servicio por el plano de control.

Las redes SDN se modelan como un conjunto de servicios cliente-servidor entre el controlador SDN y los equipos conectados a la red. La diferencia de un servidor con un controlador SDN, es que, el controlador puede solicitar servicios de cualquier cantidad de servidores en la red. El documento especifica que la visión tradicional de SDN se la realiza en estructura de planos. Un plano de datos encargado de procesar el tráfico de usuarios, un plano de control que aloja instancias del controlador SDN y el plano de aplicación en el cual se consumen los servicios solicitados por los usuarios.

El controlador SDN cumple las necesidades de los clientes virtualizando y organizando sus recursos subyacentes. A medida que las condiciones y el entorno de la red cambian y las demandas de los clientes cambian, el controlador es responsable de establecer y actualizar continuamente el estado de los servicios de red para tener una configuración óptima basada en políticas preestablecidas.

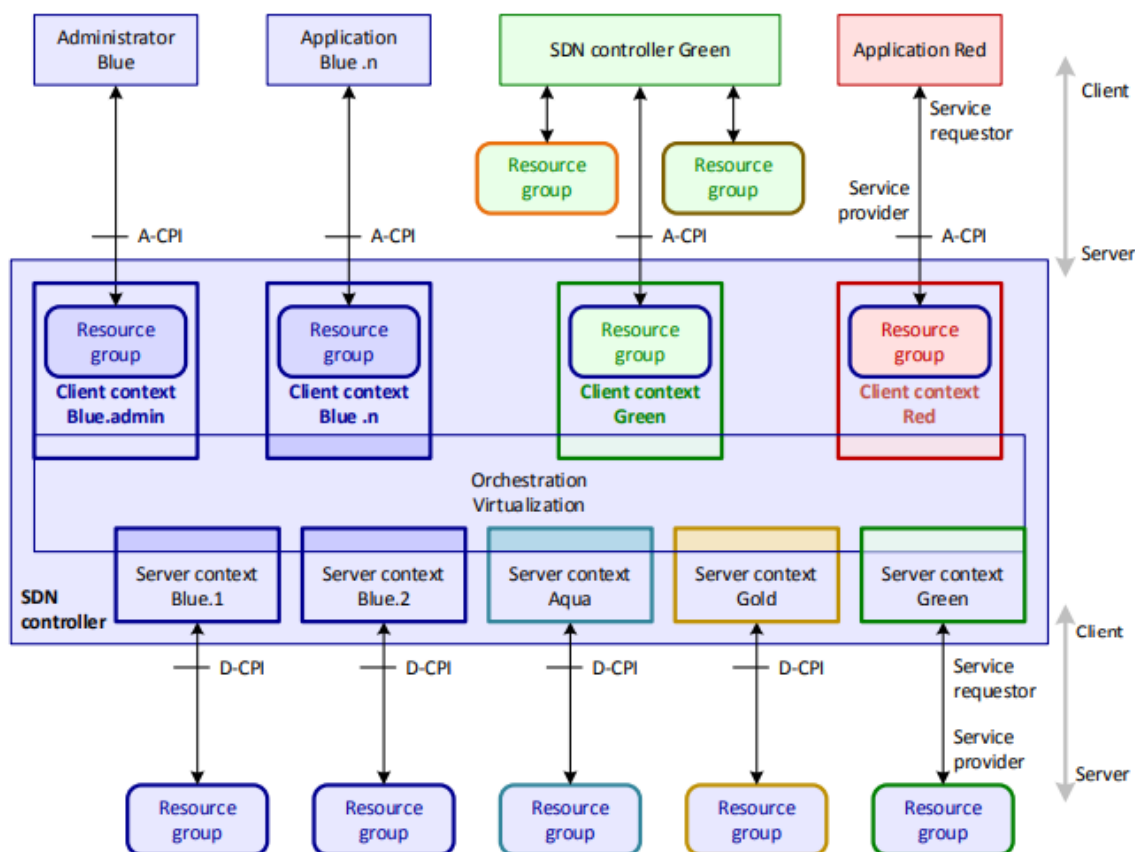


Figura 2 Núcleo de la arquitectura SDN.

Fuente: (Open Networking Foundation, 2016)

Como se puede visualizar en la Figura 2 el controlador expone sus servicios y recursos a los clientes a través de las interfaces del controlador de aplicaciones (A-CPI applications-controller plane interfaces), y consume los servicios y recursos subyacentes a través de las interfaces del controlador de datos (D-CPI data-controller plane interfaces).

Los procesos de administración y control actúan como uno en las SDN, en el cual se establece que el rol de administrador difiere del de las aplicaciones ordinarias por tener

un mayor alcance y privilegio. El administrador cuenta con los privilegios para configurar el controlador junto con los contextos del cliente y servidor.

Los controladores SDN pueden asociarse con otros controladores y entidades de administración y control que no sean SDN en arreglos jerárquicos o pares, dentro o entre dominios de red. De esta manera las SDN permiten abarcar desde la negociación del servicio del usuario final hasta el aprovisionamiento de servicios en todo el mundo y el control preciso de elementos de red individuales.

2.2.2. RFC 7426 – SDN: Layers and Architecture Terminology

Este documento científico contiene una descripción más técnica sobre la arquitectura de las SDN, definiendo un conjunto de planos y capas para su explicación.

La arquitectura SDN tiene el concepto de separación de una entidad controlada y una entidad controladora a través de una interfaz definida. Las interfaces pueden establecerse a través de la definición de un protocolo, ya sean de procesos locales (IPC) o un protocolo remoto que puede estandarizarse como abierto o patentado.

El modelo jerárquico que propone este documento científico se muestra a continuación (Internet Research Task Force - IRTF, 2015).

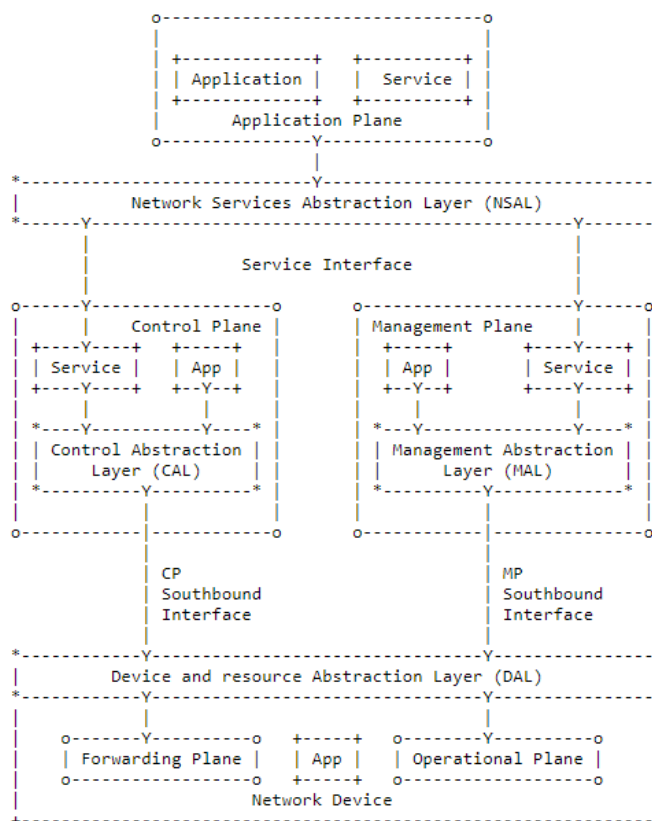


Figura 3 Arquitectura SDN del RFC 7426.

Fuente: (Internet Research Task Force - IRTF, 2015)

Como se puede observar en la Figura 3, la arquitectura de red SDN abarca varios planos, teniendo los siguientes:

- **Plano de reenvío (Forwarding Plane):** Es el responsable de manejar los paquetes en la ruta de datos, en función de las instrucciones recibidas por el plano de control, sus principales acciones se centran en el reenvío, descarte y cambio de paquetes. Este es el punto de finalización de los servicios y aplicaciones del plano de control, también conocido como el plano de datos (data plane) o ruta de datos (data path).

- Plano Operacional (Operational Plane): Es el encargado de gestionar que los dispositivos de red se encuentren en un estado operacional, reportando su estado (activo o inactivo), la cantidad y estado de puertos de comunicación, entre otros. Este plano es el punto de finalización para servicios y aplicaciones del plano de gestión. Algunas organizaciones proponen al plano operacional como un conjunto de funciones que cumple el plano de reenvío.
- Plano de control (Control Plane): En este plano se toman decisiones de cómo deben ser enrutados los paquetes por los dispositivos que componen la red y asignarles estas tareas para su ejecución. Se enfoca principalmente en el plano de reenvío y se ayuda del plano operacional para conocer el estado actual de un equipo y sus puertos, así como de la capacidad de estos. Su principal objetivo es ajustar las tablas de reenvío que residen en el plano de reenvío.
- Plano de administración (Management Plane): Es el plano de responsable de supervisar, configurar y mantener el estado de los dispositivos de red, enfocándose principalmente en el plano operacional.
- Plano de aplicación (Application Plane): En este plano residen las aplicaciones y servicios que definen el comportamiento de la red. Hay que considerar que existen aplicaciones modulares y distribuidas, por lo que, a menudo pueden abarcar más de un solo plano de la arquitectura.

Adicionalmente se consideran cuatro capas de abstracción en la arquitectura definida por el RFC 7426 y son:

- La capa de dispositivo y abstracción de recursos (The Device and resource Abstraction Layer - DAL): Es la responsable de abstraer los recursos del plano de reenvío y el plano operacional de los dispositivos hacia los planos de gestión y control.
- La capa de abstracción de control (The Control Abstraction Layer - CAL): Esta capa se encarga de abstraer la interfaz de control, hacia el sur de la arquitectura y el DAL de las aplicaciones y servicios del plano de control.
- La capa de abstracción de administración (The Management Abstraction Layer - MAL): Abstrae la interfaz del plano de administración en dirección sur y el DAL de las aplicaciones y servicios del plano de administración.
- La capa de abstracción de servicios de red (The Network Services Abstraction - NSAL): Proporciona la abstracción de servicio para el uso de aplicaciones y otros servicios.

2.2.2.1. Plano de control

El plano de control es el responsable de la configuración del plano de reenvío, manejando los paquetes que crucen la red utilizando la interfaz de control hacia el sur (Control-Plane Southbound Interface - CPSI) con DAL como su punto de referencia. Las principales funciones del plano de control son:

- Descubrimiento y actualización de topología de red.
- Selección e instrucción de ruta de tráfico de paquetes.
- Mecanismos de conmutación por error de ruta. (Path failover mechanisms)

Las principales características con las que debe contar el plano de control es una interfaz con baja latencia y gran ancho de banda. Las aplicaciones de control como servicio de LAN privada virtual, túneles de servicio o servicios de topología usan la CAL para controlar el dispositivo de red sin proporcionar ningún servicio a las capas superiores.

2.2.2.2. Plano de administración

Este plano suele establecerse en un solo punto centralizado de la red y su objetivo es el de garantizar que la red en su conjunto funcione en sus óptimas condiciones, comunicándose con el plano operativo de los equipos por medio de la interfaz de administración hacia el sur (Management-Plane Southbound Interface -MPSI) con DAL como punto de referencia.

Las principales funciones del plano de administración están enfocadas en una vista general de la arquitectura para el uso del administrador de la red. Adicionalmente se han realizado algoritmos de automatización para las funcionalidades del plano de administración (Fault, Configuration, Accounting, Performance, Security - FCAPS), incluyendo la Gestión de fallas y monitoreo y la Gestión de la configuración.

A diferencia del plano de control, la interfaz del plano de administración (Management-Plane Southbound Interface - MPSI) no necesita características de tiempo crítico con bajas latencias. Dicha interfaz está enfocada hacia la interacción con administradores y la mensajería debe ser más frecuente que en el CPSI.

2.2.2.3. Plano de aplicación

El plano de administración lo forman los servicios y aplicaciones que se usan en los planos de control y administración. Estas aplicaciones pueden servir para el descubrimiento de topología de red, aprovisionamiento de red, reservación de ruta, entre otros.

2.3. OpenFlow

El protocolo OpenFlow permite a un servidor de software establecer el camino de reenvío de paquetes dentro de los switches de la red. En un entorno de red SDN, los switches que manejan OpenFlow realizan el reenvío de paquetes de red, pero las decisiones de conmutación o enrutamiento las realiza el controlador SDN a través del protocolo OpenFlow, tomando decisiones de envíos de paquetes, modificación de tablas de reenvío y define los mensajes de estado como paquetes recibidos obteniendo estadísticas del tráfico. Si un dispositivo de red recibe un paquete y conoce como enrutarlo lo realizara, caso contrario, enviara el paquete al controlador SDN para tomar la decisión del reenvío y establecer su decisión al conmutador para su uso futuro.

El objetivo principal de OpenFlow es permitir la interoperabilidad entre dispositivos de red, permitiendo infraestructuras de red más versátiles y escalables. La ONF define los parámetros más importantes para el funcionamiento del protocolo OpenFlow en su documento OpenFlow Switch Specification. (Open Networking Foundation, 2015)

2.3.1. Switch OpenFlow

Los componentes principales de un switch OpenFlow son:

- El canal de comunicación.
- Una o más tablas de flujos.
- Una Tabla de grupo.
- Puerto de comunicación.

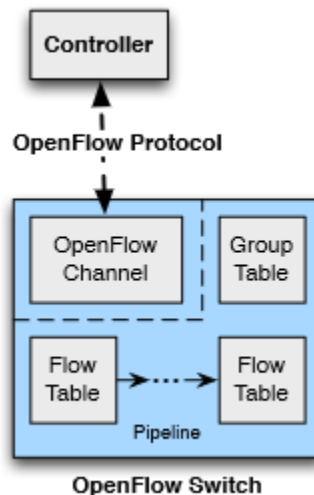


Figura 4 Componentes de un switch OpenFlow.

Fuente: (Open Networking Foundation, 2015)

Por medio del controlador el switch puede agregar, actualizar y eliminar entradas de flujo de las tablas de flujo, de forma reactiva o proactiva. Se debe tomar en cuenta que cada flujo del switch genera un conjunto de entradas de flujo; por su parte cada entrada de flujo consta de contadores, campos de coincidencia e instrucciones aplicables para los paquetes coincidentes.

La etapa de coincidencia inicia en la búsqueda en las tablas de flujos. Si se detecta una coincidencia, se ejecutan las instrucciones a la entrada de flujo específica. En caso de no existir una coincidencia en las tablas de flujo, los paquetes serán descartados, aunque pueden tener otra acción asociada.

Las instrucciones asociadas a las entradas de flujo contienen acciones como reenvió, modificación o procesamiento por tablas de grupos, e incluso pueden modificar el procesamiento de la canalización para posteriores tablas y permitir que la información sea enviada en forma de metadatos para la comunicación entre las mismas.

El procesamiento del canal de tabla finaliza cuando el conjunto de instrucciones no se encuentra asociada a otra tabla de flujo y el paquete es modificado y reenviado. De esta forma las entradas de flujo pueden ser enviadas a un puerto, ya sea este físico o lógico. Las acciones también pueden dirigir al flujo a un grupo, donde su conjunto de acciones se enfoca en la inundación, así como un reenvió más complejo de flujos.

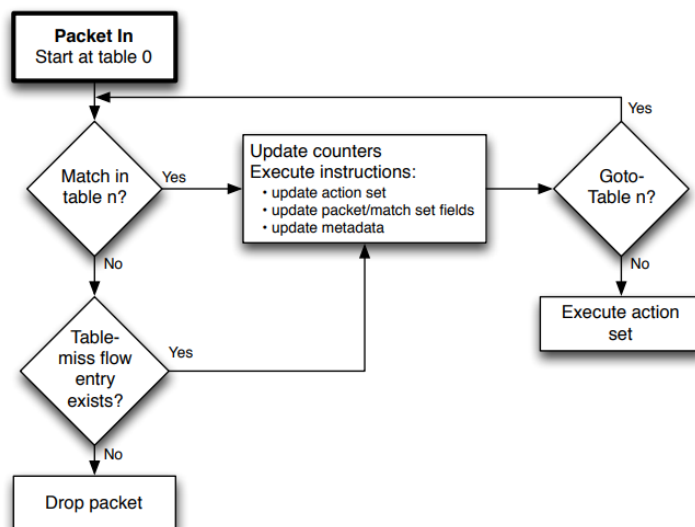


Figura 5 Diagrama de flujo del tratamiento de paquetes.

Fuente: (Open Networking Foundation, 2015)

2.3.2. Puertos OpenFlow

Los puertos OpenFlow definen las interfaces de red por los cuales se procesan los paquetes de comunicación entre el Controlador y el resto de la red. Un conmutador OpenFlow hace que una serie de puertos OpenFlow estén disponibles para el procesamiento de datos. Para establecer una comunicación OpenFlow entre dos conmutadores, es necesario la disposición de un puerto OpenFlow de salida en un switch y un puerto OpenFlow de entrada en el segundo switch.

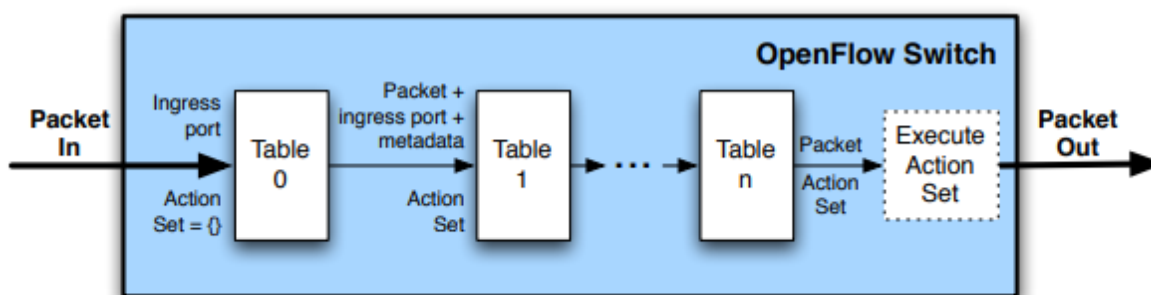
Un switch OpenFlow admite tres tipos de puertos, los físicos, lógicos y reservados. Adicionalmente hay puertos conocidos como puertos estándar los cuales definen puertos físicos, lógicos y el puerto reservado LOCAL, los cuales pueden ser usados en grupos, tienen contadores de puertos y cuentan con estado y configuración. Los diferentes puertos OpenFlow se definen de la siguiente manera:

- Puertos Físicos: Son los puertos definidos en el switch que corresponden a interfaces de hardware.
- Puertos Lógicos: Son puertos definidos por el conmutador que no corresponden directamente a una interfaz de hardware. Los puertos lógicos son abstracciones de nivel superior que pueden definirse en el conmutador utilizando métodos que no son OpenFlow y pueden ser asignados a varios puertos físicos.
- Puertos Reservados: Estos puertos especifican acciones de reenvío genéricas, como envíos al controlador, inundación o el reenvío utilizando métodos diferentes a OpenFlow, entre los puertos reservados tenemos:
 - ALL: Especifica todos los puertos que el switch tiene a disposición para reenviar un paquete. Solo puede ser usado como puerto de salida.
 - CONTROLLER: Representa el canal de control hacia el controlador OpenFlow. Puede ser utilizado como puerto de entrada o de salida.
 - TABLE: Es el inicio de canalización de OpenFlow. Solo puede ser utilizado en acciones de salida enviando el paquete a la primera tabla de flujo.
 - IN_PORT: Especifica el puerto de ingreso de paquetes. Envía el paquete a través de su puerto de entrada, es decir solo se puede usar como puerto de salida.
 - ANY: Valor usado cuando no se especifica ningún puerto en OpenFlow. No se puede usar como puerto de entrada o salida.
 - LOCAL (Opcional): Representa la pila de red local y de administración del switch. Puede ser usado como puerta de entrada o de salida.

- NORMAL (Opcional): Especifica el canal tradicional no OpenFlow de conmutador. Puede usarse como puerto de salida únicamente.
- FLOOD (Opcional): Representa el envío de inundación por canal tradicional y solo puede ser usado como puerto de salida.

2.3.3. Tablas OpenFlow

La canalización OpenFlow contiene una o más tablas de flujo con múltiples entradas de flujo, el proceso de canalización define como interactúan los paquetes con esas tablas de flujo. Un switch OpenFlow necesita al menos de una tabla de flujo para su funcionamiento.



(a) Packets are matched against multiple tables in the pipeline

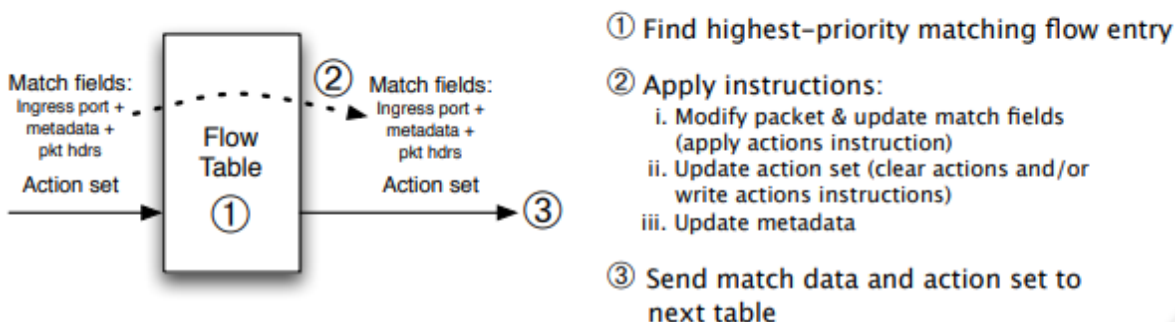


Figura 6 Flujo de paquetes a través del procesamiento de canalización.

Fuente: (Open Networking Foundation, 2015)

El procesamiento de canalización inicia en la primera tabla de flujo, comparándose con la entrada de flujo de la tabla 0, en caso de no existir coincidencia otras tablas de flujo pueden ser utilizadas en el proceso. En el caso de encontrar una tabla de flujo se ejecuta el conjunto de instrucciones asociadas a la entrada incluyendo el envío del flujo a una tabla posterior. En el caso de que el flujo coincidente no dirige los paquetes a otra tabla, se procesa el conjunto de acciones y generalmente se envía el paquete.

Si un paquete no tiene coincidencias con una entrada de flujo, esto se considera como un error de tabla. En ese caso las opciones más comunes son la eliminación, el paso del flujo a otra tabla o reenviarlos al controlador a través del canal de control. Una tabla de flujo cuenta con las siguientes entradas de flujo:

- Match fields: Son los campos de coincidencia de los paquetes, estos pueden ser el puerto de ingreso y los encabezados de los paquetes, en algunos casos puede ser los metadatos especificados por una tabla anterior.
- Priority: Coincidencia de precedencia de la entrada de flujo.
- Counters: Se actualizan cuando los paquetes coinciden.
- Instructions: Sirven para la modificación del conjunto de acciones o el procesamiento de canalización.
- Timeouts: Cantidad de tiempo máxima de espera para que el switch expire el flujo.
- Cookie: Número de identificación para filtrar el flujo.
- Flag: Modifican la forma de gestión de las entradas de flujo.

Una entrada de tabla de flujo es identificada por sus campos de coincidencia y prioridad creando en conjunto una identificación única para la entrada de flujo. Se debe

considerar que cada tabla de flujo debe admitir la entrada de flujo de pérdida de tabla para que se procesen todas las fallas de la tabla. Las acciones que se pueden definir son enviar paquetes al controlador, eliminar paquetes o dirigirlo a una tabla posterior. Adicionalmente debe admitir el envío de paquetes al controlador utilizando el puerto CONTROLADOR y la eliminación de paquetes por medio de la instrucción Borrar acciones. Si la entrada de flujo de pérdida de tabla no existe, los paquetes que no coinciden con las entradas de flujo se eliminan.

2.3.4. Tabla de Medidores

Estas tablas consisten en entradas por medidores permitiendo a OpenFlow implementar operaciones de QoS simples desde limitación de velocidad hasta DiffServ. Los medidores miden y controlan la velocidad del agregado de todas las entradas de flujo a las que está conectado. Cada entrada del medidor se identifica por su identificador de medidor y contiene:

- Meter identifier: Entero sin signo de 32 bits que identifica al medidor de forma única.
- Meter bands: Lista desordenada de bandas de medidores en el que se especifica la velocidad de la banda y la forma de procesar el paquete.
 - Band Type: define cómo se procesan los paquetes.
 - Rate: Es usada por el medidor para seleccionar la banda del medidor, definiendo la tasa más baja a la que puede aplicar la banda.

- Burst: Define la granularidad de la banda del medidor.
- Counters: Contador de los paquetes que son procesados por una banda.
- Type Specific Arguments: Argumentos adicionales para algunas bandas.
- Counters: Son variables que aumentan cuando los paquetes son procesados por un medidor.

2.3.5. Instrucciones

Cada entrada de flujo contiene un conjunto de instrucciones que son ejecutadas cuando el paquete coincide con la entrada. Las instrucciones tienen el objetivo de realizar cambios en el paquete, el conjunto de acciones o el procesamiento de canalización. Los tipos de instrucciones más importantes son:

- Meter meter_id: Identificador de medidor.
- Apply-Actions: Ejecuta las acciones específicas inmediatamente sin cambiar el conjunto de acciones.
- Clear-Actions: Borra las acciones definidas en el conjunto de acciones.
- Write-Actions: En el caso de existir una acción en el conjunto actual se debe sobrescribirla, de lo contrario se agrega.
- Write-Metadata: Escribe el valor de metadatos enmascarados en el campo de metadatos.
- Goto-Table next-table-id: Define la siguiente tabla en el proceso del paquete.

Se debe considerar que el conjunto de instrucciones asociada a una entrada de flujo cuenta con un máximo por cada tipo de conjunto. Las instrucciones siguen el orden especificado en el listado.

2.3.6. Set de acciones

El conjunto de acciones son las actividades relacionadas con los flujos una vez detectada una coincidencia en la entrada de flujo. Una entrada de flujo puede modificar el conjunto de acciones a través de las instrucciones de Write-Action o Clear-Action asociada a una coincidencia particular. Cuando una acción es agregada en el conjunto de acciones y existe una acción del mismo tipo, se sobrescribe con la acción posterior, por tanto, en el caso de necesitar múltiples acciones del mismo tipo, se debe usar la instrucción Apply-Actions. Las acciones que forman un conjunto de acciones se ejecutan en el siguiente orden especificado:

1. Copy TTL inwards: Aplica las acciones de TTL al paquete en entrada.
2. Pop: Aplica todas las opciones POP de etiqueta del paquete.
3. Push-MPLS: Aplica la acción de push de la etiqueta MPLS.
4. Push-PBB: Aplica la acción de push de la etiqueta PBB.
5. Push-VLAN: Aplica la acción de push de la etiqueta de VLAN.
6. Copy TTL outwards: Aplica las acciones de TTL al paquete en salida.
7. Decrement TTL: Aplica la acción de decrementar TTL al paquete.
8. Set: Aplica todas las acciones de campo de conjunto.
9. Qos: Aplica todas las acciones de QoS.
10. Group: Aplica las acciones de grupo en caso de existir.

11. Output: Reenvía el paquete por el puerto especificado por la acción de salida.

Si no se especificaron acciones de salida, ni acciones de grupo en un conjunto de acciones, el paquete se descarta, de igual manera ocurre si se especifica un puerto de salida no existente. El listado de acciones se muestra a continuación:

- Output port_no: Esta acción reenvía un paquete a un puerto específico OpenFlow, ya sea físicos, lógicos o reservados.
- Group_ip: Envía al paquete a ser procesado por un grupo específico.
- Drop: No existe una acción definida para Drod. En su lugar, los paquetes cuyos conjuntos de acciones no tienen una acción de salida y ninguna acción de grupo se deben eliminar.
- Set-Queue_id: Establece el ID de cola para un paquete. El comportamiento de reenvío es dictado por la configuración de la cola y se usa para proporcionar soporte básico de Calidad de Servicio (QoS).
- Push-Tag/Pop Tag Ethertype: Inserta etiquetas para ayudar en la integración con las redes existentes en temas como soporte a VLANs, MPLS o PBB.
- Type Value: Las diversas acciones de Set-Field se identifican por su tipo de campo y modifican los valores de los respectivos campos de encabezado en el paquete.
- TTL: Modifica los valores de TTL en IPv4, el Hop Limit en IPv6 o el MPLS TTL.

2.3.7. Canal OpenFlow

El canal OpenFlow se define como la interfaz que conecta cada switch a su controlador. A través de este canal el controlador configura y administra al equipo de red, el controlador recibe eventos y envía paquetes por la red. El canal OpenFlow por defecto se encuentra cifrado mediante protocolo TLS, pero puede ejecutarse directamente sobre TCP.

Los mensajes de comunicación entre controlador y conmutador son iniciados por el controlador y no siempre se espera una respuesta del conmutador. Los mensajes de comunicación pueden ser los siguientes:

- Características: El controlador envía este mensaje para solicitar la identidad y capacidades básicas del switch mediante esta solicitud. El switch debe responder con sus características de identificación, esto ocurre cuando se establece el canal OpenFlow.
- Configuración: El controlador utiliza estos mensajes para consultar y configurar al switch.
- Estado de modificación: Es utilizado para agregar, eliminar o modificar las entradas de flujo en las tablas OpenFlow y establece las características del puerto de comunicación.
- Estado de lectura: Su objetivo es recopilar información del switch en tiempo real.
- Salida de paquetes: Este mensaje es utilizado por el controlador para enviar paquetes desde un puerto específico del switch y para reenviar los paquetes

recibidos a través de mensajes de entrada de paquetes. Este mensaje debe contener además la lista de acciones que se aplicaran en el orden específico.

- Barrera: Los mensajes de solicitud/respuesta de barrera son usados para garantizar que se cumplieron las dependencias de los mensajes o para recibir notificaciones de operaciones finalizadas.
- Solicitud de función: Por medio de estos mensajes el controlador establece la función del canal OpenFlow o consultar la función.
- Configuración asíncrona: Se usa para establecer un filtro adicional en los mensajes asíncronos que desea recibir el controlador en su canal OpenFlow o para consultar el filtro.

La clasificación de los mensajes entre conmutador y switch se dividen en dos grupos: síncronos y asíncronos. Los mensajes asíncronos se envían desde el switch sin que el controlador los solicite para indicar la llegada de un paquete o por algún cambio en el estado del switch. Los principales mensajes asíncronos son:

- Packet-in: Este mensaje transfiere la potestad de control del paquete al controlador.
- Flow-Removed: Por medio de este mensaje en switch informa al controlador sobre la eliminación de una entrada de flujo de su tabla.
- Port status: Se informa al controlador sobre el cambio de puerto en una comunicación.
- Role-status: En caso de existir más de un controlador en la red, este mensaje especifica el cambio de rol de un controlador en la red.

- Controller-Status: Se informa al controlador por el cambio de estado de un canal OpenFlow.
- Flow-Monitor: Informa sobre un cambio en una tabla de flujo al controlador.

Por su parte, los mensajes simétricos se envían sin solicitud en ambos sentidos, los principales mensajes son:

- Hello: Se utilizan para iniciar una conexión entre el conmutador y el controlador.
- Echo: Este mensaje se utiliza para verificar la estabilidad de conexión entre el conmutador y controlador y usualmente utilizado para medir parámetros de latencia y ancho de banda.
- Error: Se utilizan para reportar problemas de conexión, ya sea por parte del controlador o del conmutador.
- Experimenter: Área destinada para futuras versiones de OpenFlow.

2.3.8. Manejo de Mensajes

Una de las características del protocolo OpenFlow es la entrega de mensajes confiables, pero no proporciona mensajes de acuse de recibo o el procesamiento ordenado de mensajes recibidos. El manejo de mensajes mostrados a continuación ocurre en conexiones principales y conexiones auxiliares que utilicen transporte confiable.

- Entrega de mensajes: Se garantiza la entrega de mensajes por el canal OpenFlow a menos que este tenga fallas por completo y en este caso el controlador no aceptara mensajes que provengan del conmutador con fallas.

- **Procesamiento de mensajes:** En el protocolo OpenFlow el conmutador debe procesar todos los paquetes que provengan del controlador sin excepción. En caso de poder hacerlo, el conmutador debe enviar un mensaje de error como notificación al controlador.
- **Agrupaciones de mensajes:** Los mensajes pueden ser tratados en agrupaciones por parte del controlador, haciendo que se procesen juntos y en caso de existir un error ninguno de los mensajes de la agrupación se aplica en el conmutador.
- **Pedido de mensajes:** En ausencia de mensajes de barrera, los conmutadores pueden reordenar arbitrariamente los mensajes para maximizar el rendimiento; por lo tanto, los controladores no deben depender de un orden de procesamiento específico.

2.3.9. Cifrado

Por defecto el mecanismo de seguridad de OpenFlow es TLS (Transport Layer Security – Seguridad de capa de transporte). TLS es usado para establecer la autenticación y cifrado de la conexión. Para efectos prácticos de seguridad se recomienda el uso de la versión 1.2 o superior. El conmutador y el controlador se autentican mutuamente intercambiando certificados firmados por una clave privada por el puerto predeterminado TCP 6653. Otra alternativa es el uso de certificados autofirmados o usar intercambio de claves previamente compartidas para la autenticación. Opcionalmente se pueden comunicar utilizando únicamente el protocolo TCP.

2.4. Controladores

Uno de los componentes principales en la estructura básica de las SDN es el controlador. Los switches delegan toda la inteligencia de red al controlador (plano de control) y se convierten en unidades de conmutación de tráfico. Se puede considerar al controlador como el sistema operativo de la red que gestiona la comunicación entre las aplicaciones y los dispositivos. Las características que definen a los controladores son:

- Gestión del estado de red a través de una base de datos que contiene el estado de la red, configuraciones temporales y topología.
- Un modelo de datos que relaciona los recursos gestionados, las políticas de enrutamiento y otros servicios prestados por el controlador, usualmente realizado por el lenguaje Yang.
- Mecanismos de descubrimiento de dispositivos, topología y servicio.
- Un conjunto de aplicaciones a menudo RESTful exponiendo servicios a las aplicaciones de gestión, facilitando la interacción entre controlador y aplicaciones.

Los controles de código abierto más representativos son: Beacon (Atlassian Confluence Open Source Project, 2013), Floodlight (Big Switch Networks, s.f.), NOX (International Computer Science Institute, 2014), POX (The POX network software platform , s.f.), Ryu (Ryu SDN Framework Community, 2017) y OpenDayLight (TheLinuxFoundationProjects, s.f.), cuyas características se describen en la Tabla 1.

Tabla 1
Comparación de controladores SDN

Parámetro/Controlador	Beacon	Floodlight	NOX	POX	Ryu	ODL
Versión OpenFlow	V1.0	V1.0 y V1.3	V1.0	V1.0	V1.0, V1.2, V1.3, V1.4, y V1.5	V1.0, V1.2 y V1.3,
Virtualización	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch
Lenguaje de programación	Java	Java	C++	Python	Python	Java
REST API	No	Si	No	No	Si	Si
Interfaz Gráfica	Web	Web	Python+, QT4	Python+, QT4, Web	Web	Web
Soporte en Plataformas	Linux, Mac OS, Windows y Android	Linux, Mac OS, Windows	Linux	Linux, Mac OS, Windows	Linux	Linux, Mac OS, Windows
Soporte de OpenStack	No	Si	No	No	Si	Si
Multiprocesos	Si	Si	Si	No	Si	Si
Código Abierto	Si	Si	Si	Si	Si	Si
Documentación	Alta	Alta	Media	Baja	Media	Media

Fuente: (García, Rodríguez, Calderón, & Casmartíno, Controladores SDN, elementos para su selección y evaluación, 2014)

Dentro de los parámetros de selección de funcionamiento multiplataforma, soporte de versiones en OpenFlow y escalabilidad, los controladores más destacados son: Ryu y OpenDayLight. Por preferencia en lenguaje de programación el controlador elegido es Ryu.

2.4.1. Controlador Ryu

Ryu es un controlador OpenFlow que proporciona componentes de software con APIs definidas permitiendo la creación de nuevas aplicaciones de gestión y control de redes. El controlador es compatible con protocolos como OpenFlow, Netconf, OF-config y OpenFlow 1.0, 1.2, 1.3, 1.4 y 1.5. El código es de licencia gratuita desarrollado en Apache 2.0 escrito en Python (Nippon Telegraph and Telephone Corporation, s.f.). Las características claves del controlador son:

- Cuenta con la capacidad para escuchar eventos realizados de forma asincrónica como, por ejemplo: PACKET IN, FLOW REMOVED.
- Cuenta con la capacidad para analizar los paquetes entrantes como, por ejemplo: ARP, ICMP, TCP y fabricarlos para su envío por la red.
- Cuenta con la capacidad para crear y enviar mensajes OpenFlow/SDN como, por ejemplo: PACKET_OUT, FLOW_MOD, STATS_REQUEST a los planos de programación.

El código del controlador principal se encuentra organizado bajo la carpeta /ryu y sus componentes principales son:

- app/: Contiene un conjunto de aplicaciones para el northbound del controlador.
- base/: Contiene la clase base para aplicaciones RYU.
- controller/: Contiene archivos para las funciones de OpenFlow como paquetes de conmutación, generación de flujos, manejo de eventos de red, etc.

- `lib/`: Contiene conjuntos de bibliotecas para el análisis de encabezados de protocolos. Además, incluye analizadores de NetFlow y sFlow.
- `ofproto/`: Contiene información de OpenFlow y analizadores para las diferentes versiones.
- `topology/`: Contiene código que realiza procesos de descubrimiento de la topología relacionado con el conmutador OpenFlow. Internamente utiliza el protocolo LLDP.

Para su correcto funcionamiento, el controlador Ryu crea una instancia llamando el script conocido como `ryu-manager`, a través del módulo `simpleswitch2.py`.

El módulo inicia el controlador e inserta reglas en los conmutadores para habilitarlos como conmutadores simples.

2.5. Mininet-WiFi

Mininet-Wifi es una modificación del emulador de redes SDN Mininet, agregando estaciones WiFi virtualizados con parámetros como posición y relación de movimiento, además, puntos de acceso basado en controladores de Linux estándar y el controlador de simulación `80211_hwsim`. La arquitectura y componentes de software Mininet -WiFi se define en la Figura 7.

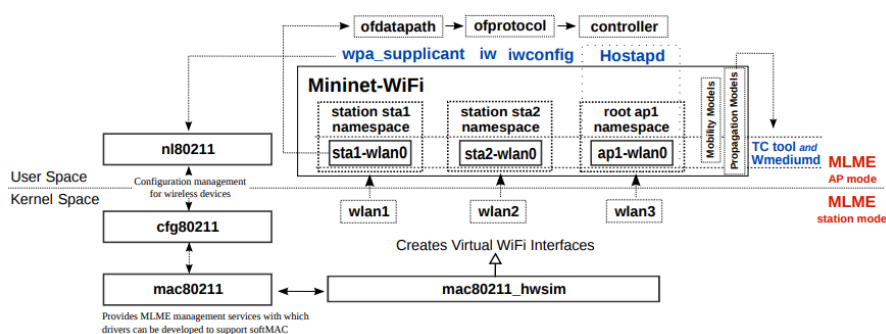


Figura 7 Componentes de Mininet-Wifi.

Fuente: (Guano, Prototipo de una SDN utilizando herramientas Open-source, 2017)

El módulo `mac80211_hwsim` es el responsable de la creación de interfaces virtuales de WiFi necesarios para simular estaciones inalámbricas y puntos de acceso. A continuación, en el espacio de Kernel, se establece el MLME que es la Entidad de administración de subcapa de control de acceso de medios a través de los módulos `iw` e `iwconfig` en las estaciones, mientras que del lado de los AP el módulo `hostapd` es el encargado de esta tarea.

Otro de los módulos fundamentales es TC (Control de Tráfico), la cual es utilizada para configurar los paquetes de kernel de Linux, en parámetros como: velocidad, retardo, latencia y pérdida de paquetes (INFORMATION & NETWORKING TECHNOLOGIES RESEARCH & INNOVATION GROUP (INTRIG), 2018).

En la Figura 8 se muestran los componentes y conexiones para una topología simple con dos estaciones en Mininet-Wifi. Adicionalmente en el ejemplo se establecen conexiones con puntos de acceso a través de enlaces cableados.

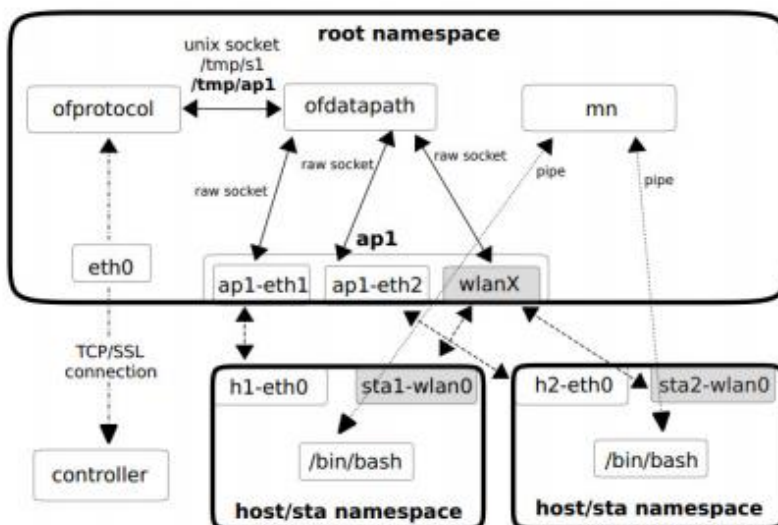


Figura 8 Componentes y conexiones en una red simulada de dos hosts en Mininet-Wifi

Fuente: (Fontes, Afzal, Brito, Santos, & Rothenberg, 2015)

Dentro de la arquitectura simulada, se agregan interfaces WiFi en estaciones inalámbricas conectándose a un punto de acceso a través de la interfaz conocida como wlanX, el cual se encuentra conectado a un conmutador OpenFlow con las capacidades representadas por ap1. Para la simulación se utiliza el lenguaje de programación Python, el cual utiliza las siguientes clases para establecer la arquitectura de red:

- UserAP(): Establece el espacio de usuario AP.
- OVSAP(): Establece un AP del tipo vSwitch.
- addHost(): Agrega un host a la topología de red.
- addAccessPoint(): Agrega un punto de acceso a la topología de red.
- addStation(): Agrega una estación (cliente inalámbrico) a la topología de red.
- addSwitch(): Agrega un switch a la topología de red.

- `addLink()`: Agrega un enlace con sentido bidireccional a la topología de red, por defecto los enlaces son bidireccionales a menos que se establezca lo contrario.
- `plotGraph()`: Establece una clase para presentar la topología en GUI.
- `startMobility`: Útil para establecer modelos de movilidad en usuarios inalámbricos.

2.6. Servidor Radius

El servidor Radius (Remote Authentication Dial-Up Server) que significa Servidor de Autenticación Remota para sistemas de acceso telefónico a redes, es un protocolo de acceso que cumple con las normas AAA (Authentication, Authorization and Accounting - Autenticación, autorización y contabilidad), lo que permite una arquitectura de red que defina parámetros de identificación, servicios permitidos y gestión de recursos (Paredes, Urbina, & Espinosa).

Este protocolo utiliza un modelo de cliente – servidor, en el cual el cliente es el encargado de realizar y confirmar el envío de solicitudes de acceso y el servidor Radius el encargado de verificar las credenciales de acceso enviadas por el cliente. Adicionalmente el servidor es el encargado de enviar al NAS (Network Access Server) los parámetros de conexión necesarias para prestar el servicio.

2.6.1. Autenticación, Autorización y Contabilidad

La autenticación es el proceso mediante el cual un cliente de red demuestra su identidad a un sistema, red u otro cliente. La identificación se realiza a través de un token, contraseñas, hashes las cuales fueron previamente almacenadas en el servidor AAA.

Posterior a la etapa de autenticación el cliente del servicio identifica los permisos a los activos y recursos de la red definidos mediante políticas de seguridad.

La contabilidad hace referencia al proceso de registrar los eventos realizados por cada cliente del servicio AAA por medio de logs informativos.

2.6.2. Funcionamiento

El proceso del servicio Radius inicia con el envío de un pedido (Solicitud de acceso) por parte del usuario, desde el NAS al servidor Radius, posteriormente se espera la correspondiente respuesta (Aceptación de acceso o Rechazo de acceso) por parte del servidor, como se puede observar en la Figura 9.

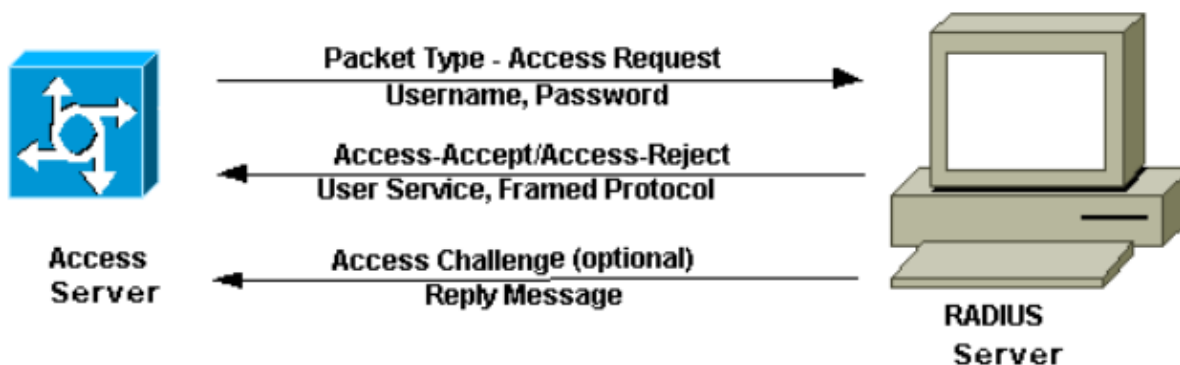


Figura 9 Proceso de autenticación y autorización del servicio Radius.

Fuente: (CISCO)

El paquete conocido como Access-request contiene información como el nombre del usuario, las credenciales de acceso encriptadas, la dirección IP del NAS y el puerto de comunicación establecido. El servicio Radius este permanece a la escucha de solicitudes de autenticación por parte de los clientes. Para el proceso de comunicación

se utiliza el protocolo UDP escuchando de forma pasiva el puerto 1812 para establecer la autenticación por el puerto 1813. Radius puede soportar múltiples métodos de autenticación de usuario como pueden ser: PPP, PAP o GRIETA, entre otros.

Una vez el servidor Radius escucha y recibe el pedido de acceso enviado desde el NAS, realiza la búsqueda en su base de datos sobre la solicitud de acceso del usuario. Si el usuario no existe dentro de la base de datos, el servidor puede tomar la decisión de cargar un perfil predeterminado o enviar el mensaje de Access-Reject (Acceso denegado). En caso de existir una coincidencia en la base de datos en servidor envía un mensaje de Access-Accept (Acceso permitido). En el servicio Radius las etapas de autenticación y autorización se encuentran unidas. En el mensaje de acceso permitido el servidor envía una lista para usarse en la sesión establecida como: tipo de servicio, la dirección IP del usuario, la lista de acceso y la ruta estática para la tabla de ruteo del NAS.

La función de contabilidad del servicio Radius es independiente de la autorización y autenticación. Esta función permite establecer los parámetros de tiempo, paquetes o bytes que se pueden usar en la sesión establecida (CISCO).

2.7. Redes Inalámbricas de Área Local (WLAN)

Las Redes Inalámbricas de área local (WLAN) son redes con acceso inalámbrico con rango típico para sus usuarios de hasta 100 metros, usados para ofrecer conectividad en entornos de oficina, hogares o estaciones de estudio, permitiendo que

sus usuarios tengan la capacidad de movilidad sin perder conexión. Se encuentran definidas en el estándar 802.11 de la IEEE y son comercializados bajo la demonización redes Wi-Fi.

El estándar 802.11 son especificaciones que definen las dos primeras capas del modelo OSI, control de acceso al medio (MAC) y capa física (PHY) enfocado en la implementación de redes inalámbricas especialmente en las bandas de frecuencias de 2,4 GHz y 5 GHz (Salazar).

En la estándar se especifican varios métodos de codificación y transmisión en la capa física como: Espectro Ensanchado por Salto de Frecuencia (Frequency Hopping Spread Spectrum - FHSS), Espectro Ensanchado por Secuencia Directa (Direct Sequence Spread Spectrum - DSSS) y Multiplexación por División de Frecuencias Ortogonales (Orthogonal Frequency Division Multiplexing - OFDM), debido a esto se especifican las siguientes versiones del estándar a lo largo de la IEEE 802.11.

- IEEE 802.11: Cuenta con una velocidad original de 2 Mbps por medio del método de transmisión FHSS y en la banda de frecuencia de 2,4 GHz. Bajo condiciones no ideales la velocidad de transmisión era menor de 1 Mbps.
- 802.11b: En esta versión del estándar se establece una estandarización de la capa física con el método de transmisión DSSS permitiendo velocidades adicionales de 5,5 Mbps y 11 Mbps, usando la banda de frecuencias de 2,4 GHz.
- 802.11a: Puede ofrecer velocidades de hasta 54 Mbps usando el método OFDM en la banda de 5 GHz, permitiendo el envío de subportadoras en paralelo.

- 802.11g: En este estándar de la IEEE se puede alcanzar velocidades de hasta 54 Mbps por medio del uso de las bandas de 2,4 y 5 GHz a través del método OFDM. Es compatible con 802.11b utilizando DSSS.
- 802.11n: El objetivo principal de esta versión del protocolo es mejorar el servicio en parámetros de distancia, además de un aumento significativo en velocidades de transmisión de hasta 600 Mbps, añadiendo tecnologías como MIMO (múltiples entradas múltiples salidas) y canales de 40 MHz. El estándar 802.11n puede funcionar en las bandas de 2,4 Ghz y 5 GHz.
- 802.11ac: Este estándar ofrece tasas de velocidad teorías de hasta 1,3 Gbps y trabaja en las bandas de 2,4 GHz y 5GHz. Incorpora tecnologías como formación de haz, banda ancha y uso de múltiples antenas de comunicación.

2.7.1. Seguridad en Redes Inalámbricas WLAN

En las arquitecturas de red inalámbricas, se utiliza un punto de acceso también conocido como AP (Access Point) para interconectar a todos los clientes. Por lo general el AP se encuentra conectado a equipos intermedios de red por medios cableados, transmitiendo datos entre red cableada y red inalámbrica.

En el caso de la red inalámbrica el acceso no debe ser por medio de un punto físico como es el caso de las cableadas, es decir, cualquier persona con credenciales de acceso que se encuentre en el área de cobertura del punto de acceso puede tener conectividad o ver el contenido de los paquetes que circulan por el medio de transmisión que en este caso es el aire. Los principales protocolos de seguridad en redes inalámbricas son (Panda Software, 2005):

- WEP (Wired Equivalent Privacy): Establece una clave estática de 64 o 128 bits por medio del algoritmo RC4, actualmente presenta fallos graves de seguridad.
- WPA (Wi-Fi Protected Access): Utiliza TKIP (Temporal Key Integrity Protocol) como protocolo de cifrado y fue creado para corregir los múltiples fallos de WEP. En la práctica WPA sigue usando RC4 como algoritmo de cifrado de 128 bits, pero usa TKIP para cambiar claves de forma dinámica.
- WPA2: Utiliza el protocolo de cifrado AES y requiere la configuración de una clave precompartida PSK entre el punto de acceso y el usuario.

2.7.2. Autenticación 802.1X (Seguridad Empresarial)

El proceso de autenticación 802.1x es independiente del proceso de autenticación 802.11 utilizado, esto se debe a que 802.1x cuenta con múltiples protocolos de autenticación y gestión de claves y utiliza los protocolos 802.11 para establecer las comunicaciones entre cliente y punto de acceso. El proceso de autenticación se basa en 4 pasos y son:

1. El cliente envía una “solicitud de acceso” al AP y este solicita la identidad del cliente.
2. El cliente que hizo la solicitud responde con un paquete de identidad que posteriormente será enviado al servidor de autenticación.
3. En caso de tener un registro, el servidor de autenticación envía un paquete de aceptación al AP.
4. El punto de acceso habilita el puerto de comunicación al cliente y le permite el envío de datos.

En 802.1x se utiliza una gran variedad de algoritmos de autenticación entre los que pueden estar: EAP-TLS, EAP-TTLS, EAP y LEAP, los cuales utilizan un servidor Radius para su identificación.

2.7.2.1. Tipos de Autenticación

Los métodos de autenticación más utilizados en las redes inalámbricas son (INTEL, 2018):

- TLS (Seguridad en capas de Transporte - Transport Layer Security): Método de autenticación basado en el protocolo EAP (Protocolo de Autenticación Ampliable – Extensible Authentication Protocol). EAP-TLS basa su método en la utilización de certificados de seguridad que usan contraseñas. Adicionalmente permite que el servidor negocie con el cliente un algoritmo de codificación y sus respectivas llaves antes de iniciar la transmisión de datos. En este método se utilizan certificados mutuos entre el cliente y servidor.
- TTLS (Seguridad de nivel de transporte de túnel – Tunneled Transport Layer Security): En este método el cliente inalámbrico utiliza EAP-TLS para validar el servidor de autenticación y crear un canal TLS codificado. Actualmente TTLS permite el uso de cualquier método definido por EAP, PAP, CHAP, MS-CHAP o MS-CHAPv2. En este método solo se requiere certificados del lado del servidor debido al establecimiento de un canal codificado de comunicación.
- PEAP (Protected Extensible Authentication Protocol Protegido - Protected Extensible Authentication Protocol): Este método de transporte seguro de datos de autenticación establece túneles de comunicación entre cliente y servidor al igual

que TTLS. PEAP realiza el proceso de autenticación con certificados utilizados solo por parte del servidor.

- LEAP (Protocolo de autenticación extensible ligero - Lightweight Extensible Authentication Protocol): Este es un método parecido a EAP usado en redes WLAN de Cisco. Por medio de LEAP se encripta el canal de comunicación por medio de claves WEP dinámicas y admite autenticación mutua.

2.7.2.2. Protocolo de Autenticación

Los protocolos principales de autenticación para redes inalámbricas son:

- PAP (Protocolo de autenticación de contraseña – Password Authentication Protocol): Es un protocolo de autenticación simple, el cual establece un enlace de dos vías para enviar claves en texto plano. Solo se encuentra disponible para autenticación del tipo TTLS.
- CHAP (Protocolo de autenticación por desafío mutuo – Challenge Handshake Authentication Protocol): Establece un enlace de tres vías en el cual se realizan procesos periódicos de verificación de la identidad del cliente.
- MS-CHAP (Protocolo de autenticación por desafío mutuo de Microsoft - Microsoft Challenge Handshake Authentication Protocol): Este protocolo es la versión de Microsoft del protocolo CHAP, el cual es irreversible.
- MS-CHAPv2: Es la versión siguiente a MS-CHAP, la cual cuenta con la función de cambio de contraseña. Esto permite que el cliente modifique su clave si el servidor Radius informa su vencimiento.

2.8. EDUROAM

EDUROAM (EDUcation ROAMing) es un proyecto enfocado en el acceso a Internet para los usuarios de la comunidad académica y científica asociadas. Actualmente se encuentra disponible en 89 países y 5331 instituciones alrededor del mundo. En Ecuador son 121 las instituciones que pertenecen al proyecto (CEDIA, s.f.).

La conexión a Eduroam se realiza habitualmente por medio de un punto de acceso inalámbrico, que utiliza el SSID “eduroam”. Se basa en 802.1x y una arquitectura de red con jerarquía vinculada con servidores Radius que contienen la base de datos de usuarios. Las instituciones vinculadas al proyecto deben contar con una red con el servicio Radius operativa y aceptar los términos y condiciones de Eduroam, configurando la arquitectura de la siguiente manera:

1. Configurar el servidor Radius conectado a su servidor de identidad institucional (LDAP).
2. Conecte sus puntos de acceso como clientes del servidor Radius.
3. Federar el servidor Radius.

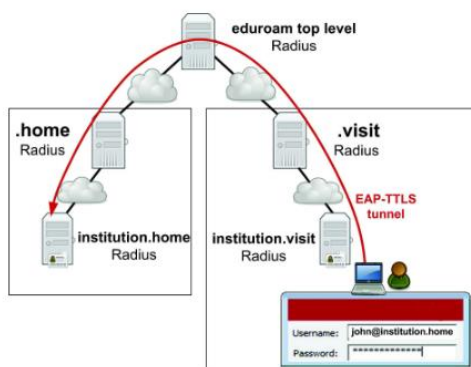


Figura 10 Arquitectura general de red EDUROAM.

Fuente: (EDUROAM, s.f.)

Eduroam proporciona un cifrado de extremo a extremo garantizando que las credenciales de acceso de los usuarios sean privadas y solo se encuentren disponibles para la institución de origen (EDUROAM, s.f.).

Los protocolos de autenticación permitidos en EDUROAM son: EAP-PEAP-MSCHAPv2 y el cifrado de red usado es WPA2-AES.

CAPÍTULO III: IMPLEMENTACIÓN

3.1. Consideraciones para el diseño del Testbed

Para el diseño del Testbed se debe considerar que se quiere lograr una comparación lo más justa posible entre el Testbed y la red inalámbrica actual de los Laboratorios de Eléctrica y Electrónica, tomando en cuenta los factores que puedan influir en el proceso de roaming inalámbrico y que estén presentes en ambas redes. Estos factores son:

- Versión del estándar IEEE 802.11: Existe una gran cantidad de revisiones del estándar IEEE 802.11 que traen cambios significativos en cuanto a capa física y MAC, entre otros factores que pueden afectar el tiempo de roaming (Banerji & Chowdhury, 2003). Debido a este sinnúmero de factores, la solución es utilizar equipos que soporten el mismo estándar que los encontrados en la red actual.
- Método de autenticación: La red con la que se realizaron las comparaciones es la red Eduroam de la Universidad de las Fuerzas Armadas ESPE, la cual utiliza un sistema de autenticación WPA2-EAP con un servidor Radius, por lo que este sistema debe ser replicado en el Testbed.
- Niveles de potencia: La potencia de transmisión de los Access Points tiene un gran impacto en cómo y cuándo una STA (Estación Inalámbrica) decide realizar el

proceso de roaming, por lo que estos también deberán ser lo más similares posible entre los dos escenarios.

- Estación móvil: La tarjeta de interfaz de red (NIC) de una estación móvil inalámbrica (STA) incluye un algoritmo que decide bajo qué circunstancias hacer roaming, y esta varía entre fabricantes (Wi-Fi Alliance, s.f.), por lo que para una comparación justa se debe utilizar la misma STA como el cliente que hará roaming en todas las pruebas.

Otras características como la existencia de un controlador de red inalámbrica (Wireless LAN Controller) en la red actual o un controlador OpenFlow en el Testbed son inherentes de cada arquitectura de red a ser estudiada, y son las que darán las diferencias que permiten este estudio comparativo, por lo que estas no serán tomadas en cuenta para evaluar la igualdad de condiciones.

Adicionalmente, aspectos como el retardo entre los Access Points y el Servidor Radius que varían considerablemente entre la red actual y el Testbed debido a que la red de distribución de la Universidad es mucho más compleja y desconocida no podrán ser implementados directamente en el Testbed, por lo que se deberá considerar estas diferencias en el análisis de resultados.

3.2. Hardware y Software utilizado

Teniendo en cuenta las consideraciones anteriores, los equipos que se utilizaron para la implementación del Testbed son:

- 2x Northbound Networks Zodiac WX OpenFlow Access Points: En base a las características mencionadas en la sección previa y el hecho de que los Access Points utilizados en la red de la Universidad son los Cisco AIR-CAP36021-A-K9, se eligió este modelo para la implementación de la red inalámbrica. Soportan los estándares IEEE 802.11ac, WPA2-EAP, además de estar garantizados para soportar OpenFlow 1.3.



Figura 11 Access Point Zodiac WX

Fuente: (Northbound Networks, s.f.)

- 1x Switch 3Com SuperStack 3 4250T: Switch de capa 2 con interfaces FastEthernet 10/100, suficientemente adecuadas para la implementación del Testbed debido a que no se tomará en cuenta la velocidad de la red debido a la dificultad de replicar el throughput de la red universitaria.
- 1x Computador Dell Inspiron 13 7359: El computador que actuará como servidor FreeRADIUS y controlador SDN, además de ejecutar otros softwares necesarios para la medición de datos.

- 1x Adaptador USB a Ethernet TP-Link UE300: Adaptador GigabitEthernet para el servidor que trabajará a 100 Mbps debido a la velocidad de las interfaces del Switch.

Adicional al hardware que forma parte del Testbed, se utilizaron otros equipos para la toma de datos:

- 2x Tarjetas de Captura AirPcap Nx: Sniffers que permiten capturar todos los paquetes, incluyendo los de control y gestión, de un canal Wi-Fi a la vez, ya sea de 2.4 o 5 GHz.



Figura 12 Tarjeta de Captura AirPcap Nx.

Fuente: (Riverbed Technology, Inc., s.f.)

- 1x Computador para la captura de datos: El computador en el cual se conectarán las tarjetas de captura y se podrá monitorear el tráfico a través del software Wireshark.
- Adaptador Wi-Fi Dell Wireless 1820 802.11ac: El adaptador Wi-Fi utilizado en el computador que actuará como cliente inalámbrico y realizará el proceso de

roaming. Debe ser el mismo en todas las pruebas para obtener resultados consistentes, debido a que cada fabricante implementa su propio algoritmo para controlar bajo qué condiciones se hará roaming.

Una vez detallados los equipos físicos a ser utilizados en el Testbed se instalaron varios tipos de software libre y código abierto en el computador que actuará como servidor, los cuales son:

- Ubuntu 18.04 LTS: El sistema operativo nativo sobre el cual se ejecutarán los servicios necesarios en el Testbed.
- FreeRADIUS 3.0.16: Este software es el que prestará el servicio AAA de autenticación WPA2-EAP a los Access Points para que puedan verificar y autenticar a los usuarios que requieran conectarse a la red inalámbrica.
- ryu 4.30: El controlador OpenFlow utilizado para la investigación, actúa como un framework para instalar diferentes aplicaciones de SDN.
 - SimpleSwitch 2.0: Una aplicación para ryu que instala flujos en los Access Points en tiempo real de acuerdo a las conexiones y tipos de paquetes que se envían (inside-openflow, 2016).

Adicionalmente, se utilizaron otras aplicaciones no relacionadas con el servidor para la obtención y procesamiento de los datos que serán usados en la sección de resultados. Estas aplicaciones incluyen:

- Wireshark 2.6.4: Una herramienta que usualmente permite la captura de tramas por medios cableados. Utilizado en conjunto con las tarjetas de captura AirPcap NX se puede capturar paquetería 802.11, en particular paquetes de gestión y control que usualmente no están al alcance del usuario.
- D-ITG 2.8.1 r1023: Un inyector de tráfico que permite configurar de manera granular los flujos de datos, así como obtener estadísticas de parámetros de QoS como delay, jitter, pérdida de paquetes y throughput.

3.3. Simulación en Mininet-Wifi

3.3.1. Instalación de Mininet-wifi

Los sistemas operativos compatibles para Mininet-wifi son las distribuciones UNIX/Linux como Ubuntu, Debian, Fedora, entre otros, siendo necesario los siguientes requisitos:

- Compatibilidad con software de *switch* (*Open vSwitch*).
- Compatibilidad con Python, bash, ping e iperf.
- Permisos de usuario root para ejecución de *software*.

Existen dos métodos de instalación del software, esto puede ser a través de Dockers o a través del repositorio Github. Para la instalación desde el repositorio de Github se utiliza los siguientes comandos:

```
$ sudo apt-get install git
```

```
$ git clone https://github.com/intrig-unicamp/mininet-wifi
```

```
$ cd mininet-wifi
```

```
$ sudo util/install.sh -WlInfv
```

Otra de las fuentes de instalación del software es a través de una máquina virtual preconfigurada cargada en la página de Github de Mininet-wifi, la cual es un Ubuntu 16.04 con todas las configuraciones y software ya instalados (Github, s.f.).

3.3.2. Pruebas de funcionamiento

Para realizar pruebas de la correcta instalación del software, se establece una simulación de red simple con 2 estaciones inalámbricas y 1 punto de acceso a través del siguiente comando:

```
$ sudo mn --wifi
```

```
wifi@wifi-VirtualBox: ~
wifi@wifi-VirtualBox:~$ sudo mn --wifi
*** Creating network
*** Adding controller
*** Adding stations:
sta1 sta2
*** Adding access points:
ap1
*** Configuring wifi nodes...

*** Adding link(s):
(sta1, ap1) (sta2, ap1)
*** Configuring nodes
*** Starting controller(s)
c0
*** Starting switches and/or access points
ap1 ...
*** Starting CLI:
mininet-wifi> sta1 ping sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.142 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.060 ms
^C
```

Figura 13 Prueba de funcionamiento de red simple en Mininet-wifi.

3.3.3. Python y Mininet-wifi

Mininet-wifi a través de scripts de Python permite la creación de simulación de topologías de red.

En la generación de estaciones inalámbricas se puede indicar su dirección MAC e IP, con el siguiente comando:

```
$ net.addStation ('sta1', mac='00:00:00:00:00:02', ip='10.0.0.2/8')
```

En la generación de puntos de acceso se especifica el ssid (*Service Set Identifier*), el modo 802.11, el canal de comunicación, la posición en el mapa y su rango de conexión, a través del siguiente comando:

```
$ ap1= net.addAccessPoint ('ap1',ssid='ssid-ap1', mode='g', channel='6',  
position='70,50,0', range=30)
```

Para la simulación del controlador especificaremos su dirección IP y puerto de uso, ya que se utilizará un controlador remoto a través de ryu, usando el siguiente comando:

```
$ c1=net.addController('c1', controller=RemoteController, ip='127.0.0.1', port=6633)
```

Para la generación de conexiones cableadas a través de los puntos de acceso y el controlador, se utiliza el siguiente comando:

```
$ net.addLink(ap1, ap2)
```

Existe la posibilidad de simular los modelos de propagación de información inalámbrica, usando el siguiente comando:

```
$ net.setPropagationModel(model='logDistance', exp=5)
```

Otra de las posibilidades de simulación es la de establecer modelos de movilidad entre las estaciones inalámbricas, definiendo la velocidad máxima y mínima, la posición máxima y mínima en el mapa y se establece un modelo de asociación con el punto de acceso más cercano, a través del siguiente comando:

```
$ net.setMobilityModel(time=0, model='RandomWayPoint', max_x=120, max_y=120,  
min_v=0.3, min_v=0.5, seed=1, associationControl='ssf')
```

La prueba de simulación no busca obtener resultados comparables de *roaming* o performance de red con la implementación de esta, su objetivo es determinar el

funcionamiento del controlador ryu en términos de control y conectividad. El código utilizado para la simulación en Python es:

```
#!/usr/bin/python

'Setting mechanism to optimize the use of APs'

from mininet.node import RemoteController

from mininet.log import setLogLevel, info

from mn_wifi.cli import CLI_wifi

from mn_wifi.net import Mininet_wifi

def topology():

    "Create a network."

    net = Mininet_wifi(controller=RemoteController)

    info("*** Creating nodes\n")

    net.addStation('sta1', mac='00:00:00:00:00:02', ip='10.0.0.2/8')

    net.addStation('sta2', mac='00:00:00:00:00:03', ip='10.0.0.3/8')

    net.addStation('sta3', mac='00:00:00:00:00:04', ip='10.0.0.4/8')

    net.addStation('sta4', mac='00:00:00:00:00:05', ip='10.0.0.5/8')

    net.addStation('sta5', mac='00:00:00:00:00:06', ip='10.0.0.6/8')
```

```
net.addStation('sta6', mac='00:00:00:00:00:07', ip='10.0.0.7/8')

net.addStation('sta7', mac='00:00:00:00:00:08', ip='10.0.0.8/8')

net.addStation('sta8', mac='00:00:00:00:00:09', ip='10.0.0.9/8')

net.addStation('sta9', mac='00:00:00:00:00:10', ip='10.0.0.10/8')

net.addStation('sta10', mac='00:00:00:00:00:11', ip='10.0.0.11/8')

ap1 = net.addAccessPoint('ap1', ssid='ssid-ap1', mode='g', channel='1',
                          position='50,50,0')

ap2 = net.addAccessPoint('ap2', ssid='ssid-ap2', mode='g', channel='6',
                          position='70,50,0', range=30)

ap3 = net.addAccessPoint('ap3', ssid='ssid-ap3', mode='g', channel='11',
                          position='90,50,0')

c1 = net.addController('c1', controller=RemoteController, ip='127.0.0.1', port=6633)

net.setPropagationModel(model="logDistance", exp=5)

info("*** Configuring wifi nodes\n")

net.configureWifiNodes()

info("*** Associating and Creating links\n")
```

```
net.addLink(ap1, ap2)
```

```
net.addLink(ap2, ap3)
```

```
net.plotGraph(max_x=120, max_y=120)
```

```
net.setMobilityModel(time=0, model='RandomWayPoint', max_x=120, max_y=120,  
                    min_v=0.3, max_v=0.5, seed=1, associationControl='ssf')
```

```
info("*** Starting network\n")
```

```
net.build()
```

```
c1.start()
```

```
ap1.start([c1])
```

```
ap2.start([c1])
```

```
ap3.start([c1])
```

```
info("*** Running CLI\n")
```

```
CLI_wifi(net)
```

```
info("*** Stopping network\n")
```

```
net.stop()
```

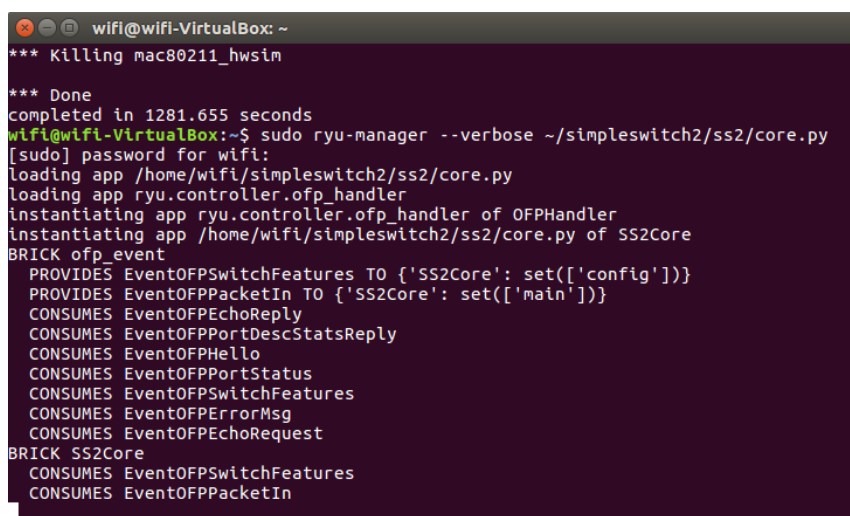
```
if __name__ == '__main__':
```

```
setLogLevel('info')
```

```
topology()
```

Para el primer paso de la simulación, se arranca el controlador ryu a través del siguiente comando:

```
$ sudo ryu-manager --verbose ~/simpleswitch2/ss2/core.py
```



```
wifi@wifi-VirtualBox: ~
*** Killing mac80211_hwsim
*** Done
completed in 1281.655 seconds
wifi@wifi-VirtualBox:~$ sudo ryu-manager --verbose ~/simpleswitch2/ss2/core.py
[sudo] password for wifi:
loading app /home/wifi/simpleswitch2/ss2/core.py
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app /home/wifi/simpleswitch2/ss2/core.py of SS2Core
BRICK ofp_event
PROVIDES EventOFPSwitchFeatures TO {'SS2Core': set(['config'])}
PROVIDES EventOFPPacketIn TO {'SS2Core': set(['main'])}
CONSUMES EventOFPEchoReply
CONSUMES EventOFPPortDescStatsReply
CONSUMES EventOFPHello
CONSUMES EventOFPPortStatus
CONSUMES EventOFPSwitchFeatures
CONSUMES EventOFPErrormsg
CONSUMES EventOFPEchoRequest
BRICK SS2Core
CONSUMES EventOFPSwitchFeatures
CONSUMES EventOFPPacketIn
```

Figura 14 Inicio de controlador ryu para simulación.

Como siguiente paso se realiza la simulación del script de Python denominado tesis.py, el cual simulara la presencia de 3 puntos de acceso y 10 estaciones de trabajo bajo el dominio del controlador ryu. El comando para iniciar la simulación es:

```
$ sudo python tesis.py
```

El resultado de la simulación será la gráfica de los equipos inalámbricos y la consola por línea de comandos del software Mininet-wifi.

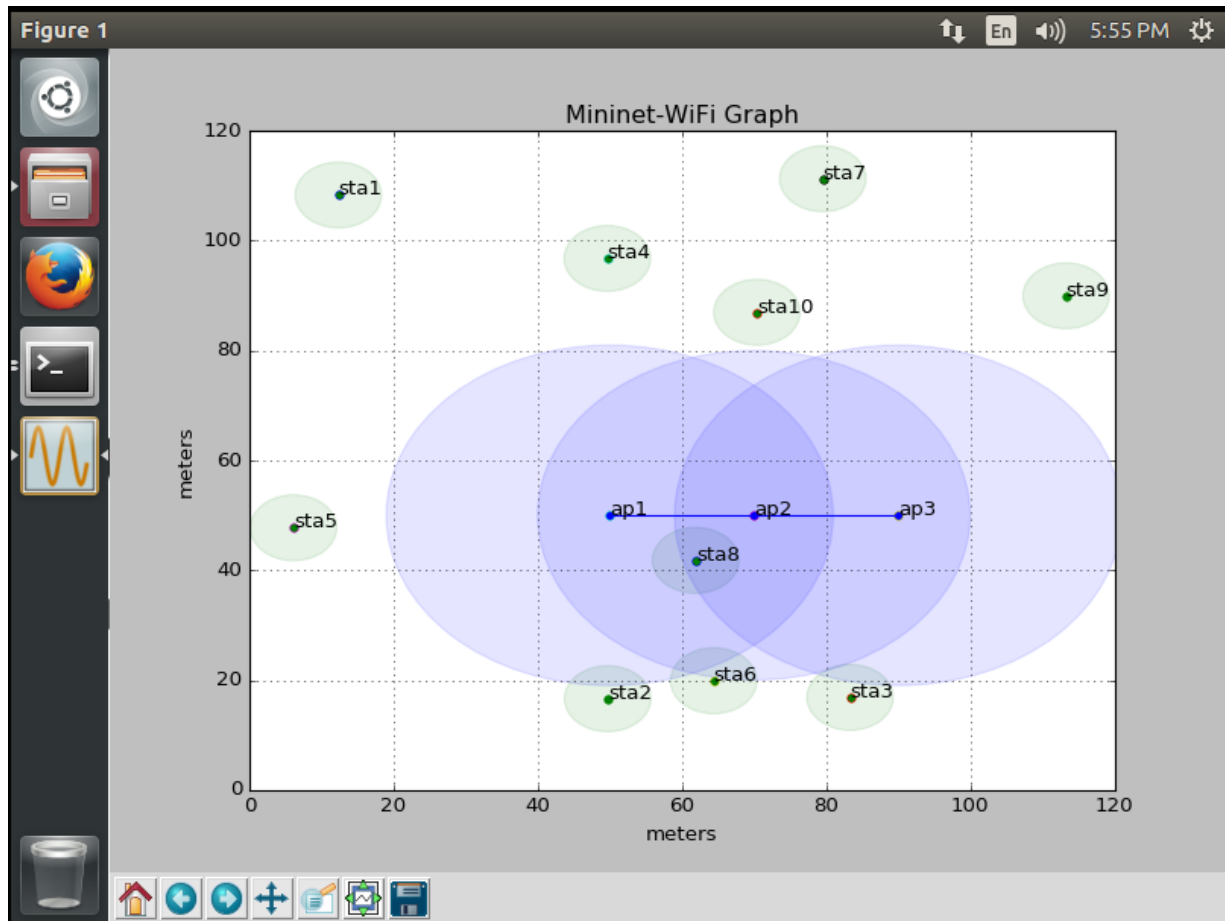


Figura 15 Simulación gráfica de red con estaciones y puntos de acceso.

En el controlador se inician los procesos de instalación de flujos para la administración de la red.

```

Terminal File Edit View Search Terminal Help
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 1152921504606846979, 2, 7a:37:3f:ab:69:9d after 0 hits
Unlearned 1152921504606846979, 2, 06:b1:eb:e3:1f:6a after 0 hits
Unlearned 1152921504606846978, 2, 06:b1:eb:e3:1f:6a after 0 hits
Unlearned 1152921504606846978, 2, 2e:38:86:cf:05:cd after 0 hits
Unlearned 1152921504606846977, 2, 2e:87:71:ff:21:6b after 0 hits
Unlearned 1152921504606846977, 2, 2e:38:86:cf:05:cd after 0 hits
Learned 1152921504606846977, 2, 00:00:00:00:00:09
Learned 1152921504606846979, 2, 00:00:00:00:00:09
Learned 1152921504606846978, 1, 00:00:00:00:00:09
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 1152921504606846977, 2, 00:00:00:00:00:09 after 1 hits
Unlearned 1152921504606846979, 2, 00:00:00:00:00:09 after 1 hits
Unlearned 1152921504606846978, 1, 00:00:00:00:00:09 after 1 hits
Learned 1152921504606846979, 2, 00:00:00:00:00:07
Learned 1152921504606846978, 1, 00:00:00:00:00:07
Learned 1152921504606846977, 2, 00:00:00:00:00:07
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 1152921504606846978, 1, 00:00:00:00:00:07 after 1 hits
Unlearned 1152921504606846979, 2, 00:00:00:00:00:07 after 1 hits
Unlearned 1152921504606846977, 2, 00:00:00:00:00:07 after 1 hits
Learned 1152921504606846979, 1, 00:00:00:00:00:04

```

Figura 16 Instalación de flujos en el controlador.

La prueba final se realiza a través de pruebas de conectividad (ping) entre la estación 2 asociada al punto de acceso 1 y la estación 6 asociada al punto de acceso 2, a través de los flujos instalados por el controlador.

```

wifi@wifi-VirtualBox: ~/mininet-wifi/examples
rtt min/avg/max/mdev = 2.068/2.642/4.374/0.753 ms
mininet-wifi> exit
*** Stopping network
*** Stopping 1 controllers
c1
*** Stopping 12 links
.....
*** Stopping switches/access points
ap1 ap2 ap3
*** Stopping nodes
sta1 sta2 sta3 sta4 sta5 sta6 sta7 sta8 sta9 sta10
*** Killing hostapd
*** Killing mac80211_hwsim

*** Done
wifi@wifi-VirtualBox:~/mininet-wifi/examples$ sudo python test.py
*** Creating nodes
*** Configuring wifi nodes
*** Associating and Creating links
*** Starting network
*** Configuring nodes
*** Running CLI
*** Starting CLI:
mininet-wifi> sta2 ping sta6
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data:
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=8.25 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=3.76 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=3.58 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=3.87 ms
64 bytes from 10.0.0.7: icmp_seq=5 ttl=64 time=4.06 ms
64 bytes from 10.0.0.7: icmp_seq=6 ttl=64 time=3.60 ms
64 bytes from 10.0.0.7: icmp_seq=7 ttl=64 time=10.4 ms

```

Figura 17 Pruebas de conectividad entre estaciones por el controlador.

3.4. Implementación del Testbed

3.4.1. Topología de la red

Teniendo claros los componentes que conformarán el Testbed, tanto hardware como software, se puede detallar la topología de la red, la cual consistirá en el servidor que actuará como controlador OpenFlow y servidor Radius, los Access Points con OpenFlow y el switch capa 2 que permitirá la interconexión de los demás componentes.

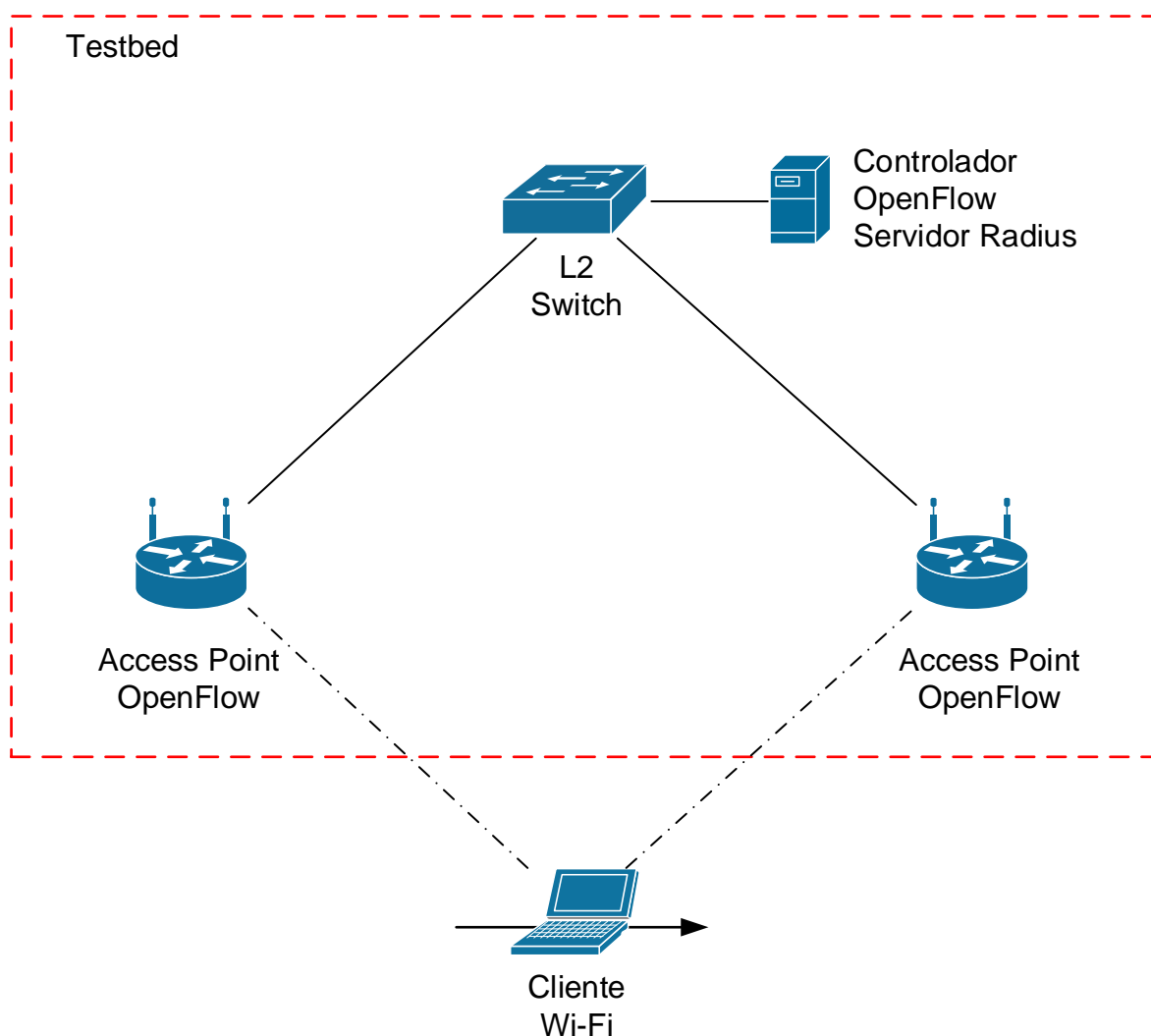


Figura 18 Topología del Testbed

El cliente Wi-Fi será el equipo que realiza el proceso de roaming, conectándose a uno de los Access Points y moviéndose físicamente hacia el segundo Access Point para que este lo autentique y le permita acceder a los servicios de la red nuevamente, completando el procedimiento de roaming.

3.4.2. Instalación y configuración de ryu y SimpleSwitch 2.0

La instalación de los componentes que permitirán controlar los Access Points mediante el protocolo OpenFlow involucra a ryu y SimpleSwitch 2.0. Para la instalación de ryu se utilizaron los siguientes comandos:

```
$ git clone git://github.com/osrg/ryu.git
```

```
$ cd ryu
```

```
$ sudo python ./setup.py install
```

De igual manera, la instalación de SimpleSwitch 2.0 requiere del siguiente comando:

```
$ git clone git://github.com/inside-openflow/simpleswitch2.git
```

Debido a que los Access Points únicamente permiten el uso de las tablas 0-99 por el controlador, se debe modificar a SimpleSwitch 2.0 para que instale flujos en tablas diferentes a las usadas por defecto: las tablas 101-103, por lo que se debe modificar el archivo “~/simpleswitch2/ss2/defaults.cfg” de la siguiente manera:

```
# the learning switch tables
```

```
table_acl:          0
```

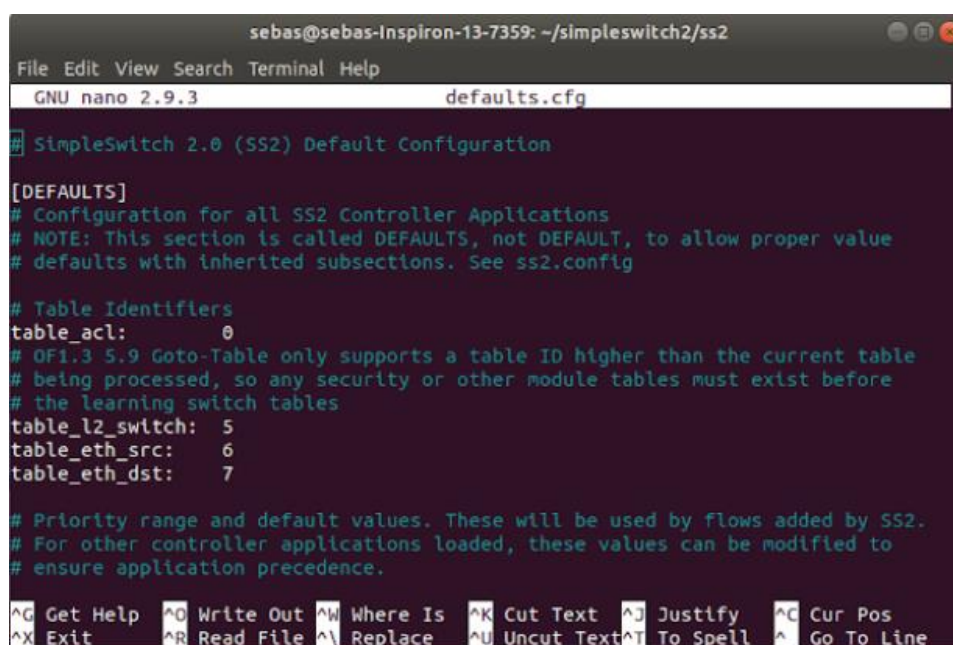


```

table_l2_switch: 5
table_eth_src: 6
table_eth_dst: 7

```

De esta manera, el controlador instalará los flujos para el reenvío de paquetes en las tablas con ID 5, 6 y 7.



```

sebas@sebas-Inspiron-13-7359: ~/simpleswitch2/ss2
File Edit View Search Terminal Help
GNU nano 2.9.3 defaults.cfg

# SimpleSwitch 2.0 (SS2) Default Configuration

[DEFAULTS]
# Configuration for all SS2 Controller Applications
# NOTE: This section is called DEFAULTS, not DEFAULT, to allow proper value
# defaults with inherited subsections. See ss2.config

# Table Identifiers
table_acl: 0
# OF1.3 5.9 Goto-Table only supports a table ID higher than the current table
# being processed, so any security or other module tables must exist before
# the learning switch tables
table_l2_switch: 5
table_eth_src: 6
table_eth_dst: 7

# Priority range and default values. These will be used by flows added by SS2.
# For other controller applications loaded, these values can be modified to
# ensure application precedence.

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^I To Spell ^_ Go To Line

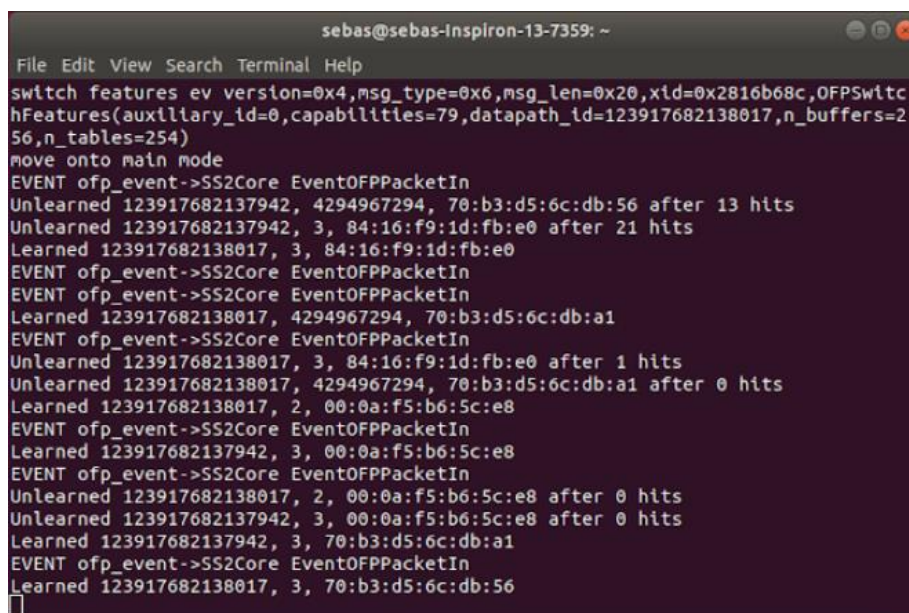
```

Figura 19 Archivo defaults.cfg modificado

Una vez configurado SimpleSwitch 2.0 se puede iniciar ryu junto con dicho módulo, el cual debería inmediatamente reconocer a los dos Access Points y empezar a instalar flujos relevantes. Se puede inicializar el controlador mediante el comando:

```
$ ryu-manager -verbose ~/simpleswitch2/ss2/core.py
```

El cual además de inicializar el módulo, imprimirá en pantalla mensajes importantes como los cambios que existan en las tablas de flujos en tiempo real.



```

sebas@sebas-Inspiron-13-7359: ~
File Edit View Search Terminal Help
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x2816b68c,OFPSwitchFeatures(auxiliary_id=0,capabilities=79,datapath_id=123917682138017,n_buffers=256,n_tables=254)
move onto main mode
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 123917682137942, 4294967294, 70:b3:d5:6c:db:56 after 13 hits
Unlearned 123917682137942, 3, 84:16:f9:1d:fb:e0 after 21 hits
Learned 123917682138017, 3, 84:16:f9:1d:fb:e0
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
Learned 123917682138017, 4294967294, 70:b3:d5:6c:db:a1
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 123917682138017, 3, 84:16:f9:1d:fb:e0 after 1 hits
Unlearned 123917682138017, 4294967294, 70:b3:d5:6c:db:a1 after 0 hits
Learned 123917682138017, 2, 00:0a:f5:b6:5c:e8
EVENT ofp_event->SS2Core EventOFPPacketIn
Learned 123917682137942, 3, 00:0a:f5:b6:5c:e8
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 123917682138017, 2, 00:0a:f5:b6:5c:e8 after 0 hits
Unlearned 123917682137942, 3, 00:0a:f5:b6:5c:e8 after 0 hits
Learned 123917682137942, 3, 70:b3:d5:6c:db:a1
EVENT ofp_event->SS2Core EventOFPPacketIn
Learned 123917682138017, 3, 70:b3:d5:6c:db:56

```

Figura 20 Salida del CLI de ryu, en la cual se observa la instalación y desinstalación de flujos en los respectivos dispositivos de red.

En la figura se puede observar como el controlador instala flujos de acuerdo a los dispositivos conectados utilizando sus direcciones MAC para filtrar los paquetes; así como los cambios en las tablas de flujos cuando existe un cambio en la topología (por ejemplo, el roaming de un cliente inalámbrico entre los dos Access Points).

3.4.3. Instalación y configuración de FreeRADIUS

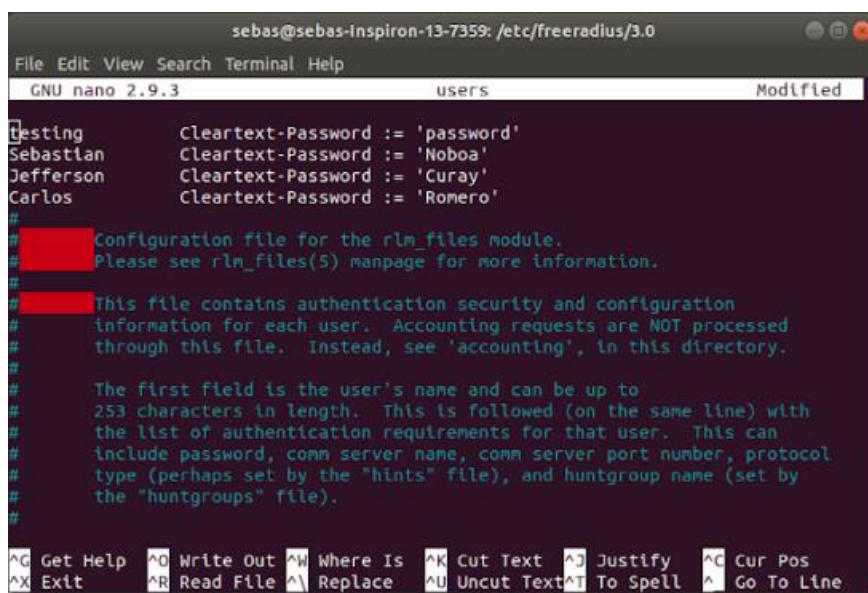
El segundo servicio esencial que se necesita en el Testbed es el servidor Radius, el cual permitirá tener un sistema AAA centralizado para autenticar y autorizar a los

clientes inalámbricos mediante los Access Points utilizando varios protocolos de seguridad.

Para la instalación de FreeRADIUS se utilizó la versión disponible en los repositorios de Ubuntu, por lo que el único comando utilizado fue:

```
$ sudo apt-get install freeradius
```

Una vez instalado, se deben crear los usuarios que tendrán acceso a la red en el archivo `/etc/freeradius/3.0/users` de la siguiente manera:

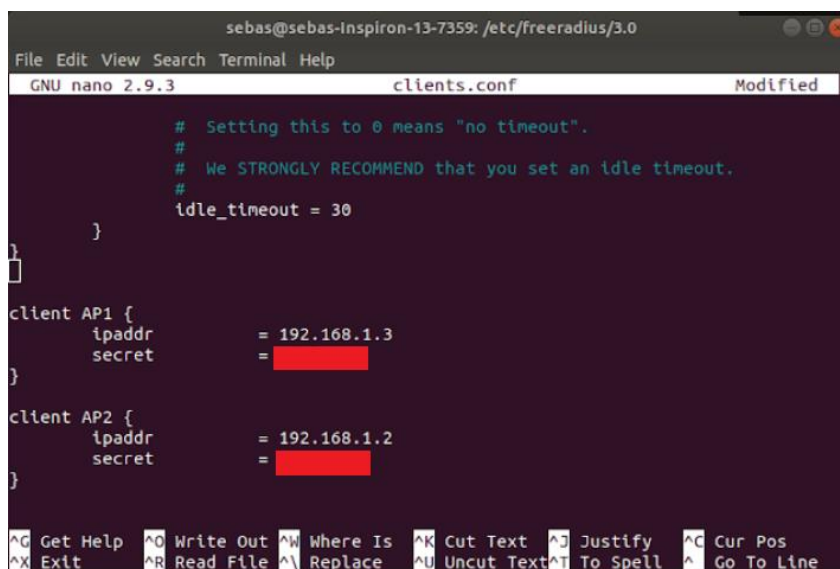


```
sebas@sebas-Inspiron-13-7359: /etc/freeradius/3.0
File Edit View Search Terminal Help
GNU nano 2.9.3 users Modified
testing      Cleartext-Password := 'password'
Sebastlan   Cleartext-Password := 'Noboa'
Jefferson   Cleartext-Password := 'Curay'
Carlos      Cleartext-Password := 'Romero'
#
# Configuration file for the rlm_files module.
# Please see rlm_files(5) manpage for more information.
#
# This file contains authentication security and configuration
# information for each user.  Accounting requests are NOT processed
# through this file.  Instead, see 'accounting', in this directory.
#
# The first field is the user's name and can be up to
# 253 characters in length.  This is followed (on the same line) with
# the list of authentication requirements for that user.  This can
# include password, comm server name, comm server port number, protocol
# type (perhaps set by the "hints" file), and huntgroup name (set by
# the "huntgroups" file).
#
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit     ^R Read File ^\ Replace  ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Figura 21 Usuarios creados para el acceso a la red inalámbrica.

El siguiente paso consiste en la creación de los clientes. En el contexto de FreeRADIUS, el cliente es el dispositivo suplicante que solicita al servidor la autenticación de los usuarios, es decir, el Access Point. Para ello se debe crear una entrada en el archivo `/etc/freeradius/3.0/clients.conf` por cada cliente, en la cual esté incluida su

dirección IP y un secreto que será enviado por el cliente al servidor para verificar su identidad.



```
sebas@sebas-Inspiron-13-7359: /etc/freeradius/3.0
File Edit View Search Terminal Help
GNU nano 2.9.3 clients.conf Modified

# Setting this to 0 means "no timeout".
#
# We STRONGLY RECOMMEND that you set an idle timeout.
#
idle_timeout = 30
}
}

client AP1 {
    ipaddr      = 192.168.1.3
    secret      = ██████████
}

client AP2 {
    ipaddr      = 192.168.1.2
    secret      = ██████████
}

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File ^V Replace  ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Figura 22 Clientes validados como suplicantes.

Una vez creados los clientes y usuarios, se debe cambiar la configuración de EAP para que utilice los mismos protocolos de autenticación que la red de la Universidad. La Red Eduroam utiliza EAP-TTLS como autenticación de primera fase y PAP como autenticación de segunda fase, información obtenida del archivo de Autoridad de Certificación (CA):

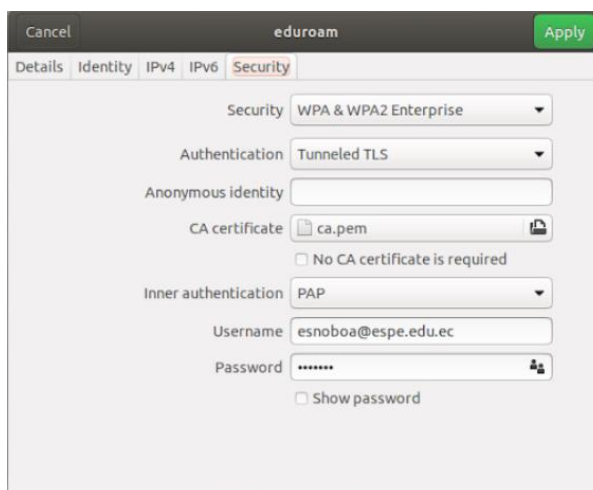


Figura 23 Información de autenticación para Eduroam proporcionada por el CA.

Teniendo la información sobre el método de autenticación en cuenta, se modifica el archivo `/etc/freeradius/3.0/mods-available/eap` para que el servidor utilice únicamente este tipo de autenticación:

```

sebas@sebas-Inspiron-13-7359: /etc/freeradius/3.0/mods-available
File Edit View Search Terminal Help
GNU nano 2.9.3 eap Modified

## EAP-TTLS
#
# The TTLS module implements the EAP-TTLS protocol,
# which can be described as EAP inside of Diameter,
# inside of TLS, inside of EAP, inside of RADIUS...
#
# Surprisingly, it works quite well.
#
ttls {

    tls = tls-common
    default_eap_type = pap

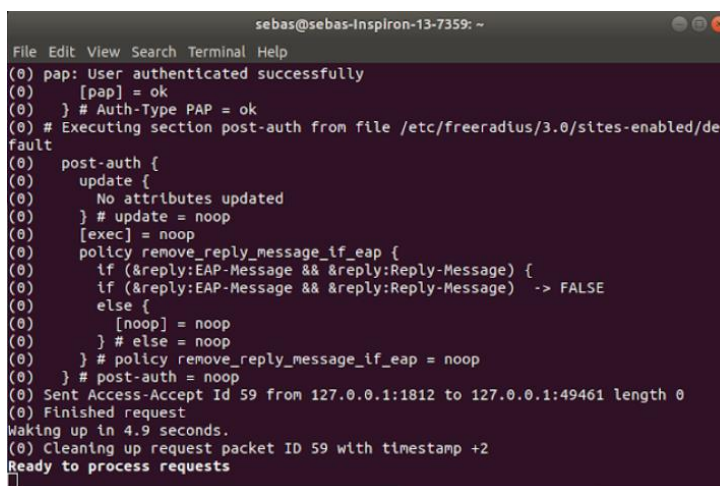
    # The tunneled authentication request does not usually
    # contain useful attributes like 'Calling-Station-Id',
    # etc. These attributes are outside of the tunnel,
    # and normally unavailable to the tunneled
    # authentication request.
  
```

Figura 24 Configuración de TTLS en el archivo eap.

Con el método de autenticación implementado, y habiendo configurado los usuarios, incluido el usuario de pruebas, se puede realizar una prueba antes de continuar con el paso final. Para esta prueba se utiliza el siguiente comando:

```
$ radtest testing password 127.0.0.1 0 testing123
```

En el cual “testing” es el nombre del usuario de prueba, “password” es la contraseña de prueba y “testing123” es el secreto para el cliente local, por lo que la dirección IP utilizada es una dirección loopback. Con el servicio ejecutándose y el comando emitido, el resultado indica que la autenticación fue exitosa:



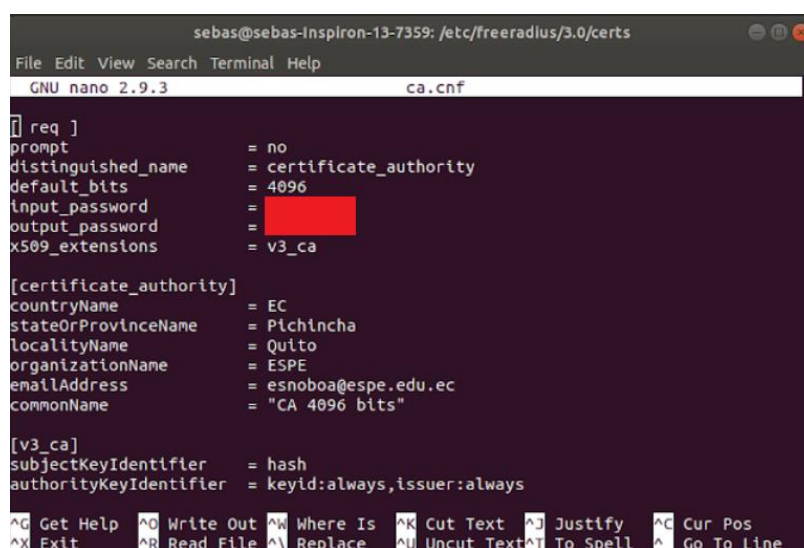
```
sebas@sebas-Inspiron-13-7359: ~  
File Edit View Search Terminal Help  
(0) pap: User authenticated successfully  
(0) [pap] = ok  
(0) ) # Auth-Type PAP = ok  
(0) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/de  
fault  
(0) post-auth {  
(0) update {  
(0) No attributes updated  
(0) } # update = noop  
(0) [exec] = noop  
(0) policy remove_reply_message_if_eap {  
(0) if (&reply:EAP-Message && &reply:Reply-Message) {  
(0) if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE  
(0) else {  
(0) [noop] = noop  
(0) } # else = noop  
(0) } # policy remove_reply_message_if_eap = noop  
(0) } # post-auth = noop  
(0) Sent Access-Accept Id 59 from 127.0.0.1:1812 to 127.0.0.1:49461 length 0  
(0) Finished request  
Waking up in 4.9 seconds.  
(0) Cleaning up request packet ID 59 with timestamp +2  
Ready to process requests
```

Figura 25 Mensaje de Access-Accept por parte del servidor Radius.

Una vez confirmado que el servidor funciona y permite la autenticación de los usuarios antes creados, se deberán crear los certificados de producción. La red Eduroam utiliza certificados de 4096 bits, por lo que se deberá generar un CA y un certificado de servidor con el mismo nivel de cifrado. Debido a que el método de

autenticación de primera fase es EAP-TTLS, no se necesita generar certificados para los usuarios (Funk & Blake-Wilson, 2008).

FreeRADIUS incluye una utilidad que permite generar los certificados mediante OpenSSL, configurando parámetros necesarios como la información de la entidad emisora, la versión de X.509, entre otros.



```
sebas@sebas-Inspiron-13-7359: /etc/freeradius/3.0/certs
File Edit View Search Terminal Help
GNU nano 2.9.3 ca.cnf
[ req ]
prompt = no
distinguished_name = certificate_authority
default_bits = 4096
input_password = 
output_password = 
x509_extensions = v3_ca

[certificate_authority]
countryName = EC
stateOrProvinceName = Pichincha
localityName = Quito
organizationName = ESPE
emailAddress = esnoboa@espe.edu.ec
commonName = "CA 4096 bits"

[v3_ca]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figura 26 Configuración para la generación de certificados y CAs.

Con los parámetros configurados, se puede generar el CA y el certificado de servidor:

```

sebas@sebas-Inspiron-13-7359: /etc/freeradius/3.0/certs
GNU nano 2.9.3 ca.pem
-----BEGIN CERTIFICATE-----
MIIGLzCCBH+gAwIBAgIJAJqd8pE/xzjCMA0GCSqGSIb3DQEBCwUAMHsx CzA3BgNV
BAYTAkVDRlIwEAYDVQQIDAlQawNoaW5jaGExDjAMBGNVBAcMBVFlaxRVmQ0wCwYD
VQQKQARFU1BFMSIwIAYJKoZIhvcNAQkBFNlc25vYm9hZGVzZGUuZmRlLnVjMRUw
EwYDVQDDAxQ0S0MDk2IGp0dHMwHhcnMTg0MTE2MTE2MDQyYWhcnMTk0MTE2MTE2
MDQyYjB7MQswCQYDVQQGEwJFQzESMBAGA1UECAwJUGUjagLUy2hhMQ4wDAYDVQQH
DAVRdWl0bzENMAsGA1UECgWERVNQRTElMCAgCSqGSIb3DQEJARYTZXNub2JvYVUB
Lc3B1LnVkdS51YzEVbHBGA1UEAwW0QEGNDASNB1BtaXRzMIICIjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICCGkCAGEarGSLwTds41poUXj0BjbcFNLH0Qb1jt+Lxfm9suJ
NDJL0X5w0C204U5cxEDJHLsLg5LYLEUFAQFshahba1f0b9ppfx3cog7kmv14h
jVqS5p1bAYHwEVE1FSYHStteay3zsrRgagEdofCcrF392HFf0BRZEL0e1wkoX
3IqAUcrs1dod5x+oepXnpjT0B7koyLXrdJucsh/XZLze0bgpA87kPrvpd8P76
vbm3Z0j+7PpohCq1aUrYANS7z7P7jvvtHCnBVtqvh9sFXdrUskLH07G/P89XQ/
uwj5Q+VcufNVVzn2vdHRj35zXkz2HPQe2duByJ0fT9BTwr8H7f5n4SK1no2i+9r
F0sgG2tG4zrc1dM60V70D0nCMCN5pUXp4Fkr7wPglU2roaF50Nk1eGNy00d0PD
KCG/6on1+Lxw20X2WjeJlBr/Bz3b4P35q00Pze/W7rDn8es51k4Chfa2s8tz/d6
r4s1zcvJj3DR1XCqa+nhvaSUCwB6M0xepTzHb5+J0LpDvu5a32buMK2nIN1Bd
S8kaB2M9LRh4a600Dw+R19g3G3vr7QhYBkVn1GpCHStAFmCbvhqXT50/+1JxtyVr
eNxs5XnCGrmsL2xJZcdn10JX+z5mPP/+mptL293apm1SeUewoh+cwV0xa+hHNGU
E3UCAwEAaAOCARwggEYMB0GA1UdDgQBBQXUeoVVD/pPZ/B1as1EdxxJvu012CB
rQYDVROjB1G1MIGlgBQXUeoVVD/pPZ/B1as1EdxxJvu016F/pH0wezELMAKGA1UE
BHMCRUMxEJAQBgNVBAGMVCBpY2hpbmNoYTEOMAwGA1UEBwwFUXVpdG8xDTALBgNV
BAoMBEVTUEUxIjAgBgkqhkiG9w0BQCQEWZvzbn9lb2FAZXNwZS51ZHUuZmRlLnVj
BgNVBAMDENBID0wOTYgYn10c4IJAJqd8pE/xzjCMA8GA1UdEwEB/wQFMAMBAF8w
NgYDVROjB1G1MIGlgBQXUeoVVD/pPZ/B1as1EdxxJvu016F/pH0wezELMAKGA1UE
BHMCRUMxEJAQBgNVBAGMVCBpY2hpbmNoYTEOMAwGA1UEBwwFUXVpdG8xDTALBgNV
BAoMBEVTUEUxIjAgBgkqhkiG9w0BQCQEWZvzbn9lb2FAZXNwZS51ZHUuZmRlLnVj
X2NhLmlybDANBgkqhkiG9w0BAQsFAAOCAgEAQ0X0J6vr8TSzntyjF5Dhusc+ss
70HZokdqW1f1IEBt/A2Nha04mxkY4RMVYKYagihFjzcVcxS1L6WAYLans54qBusL
CWZC7LEPHUCDLKczY4tBuW7C4zC3tFCqs11XC6K1S50Q5bHnQ90zqsGRWEncLzT
skcT9ly/zpJ0ao4/qnzna17b/c2h09yT17LD2b7c7wGIW9ec81tTzeRgtfJ45m
42gM0v0wscZ2gYWRlQ3V0/0A0UHV1CX00y4MNIIDtH2Mv/11hd0HfNkgp/iIk
yh2fGt+YbdFAxJ0snt1a0U5xJKFqV9IsxSOLjNY+51EK/wZ5Bh4nc/Ee7fzI83z8
/23s3JbF90Hvcq093700ItIL32yFBK0e01kLwEEngh39nLQPC1xrzZHBThIkCBy
roU2Fhs303Pp8n0Tf6wvPskxA9czyx9WRL2maM814W/rvsf1M154tGM1Bv0NjH
g1RgXyMh6cJgaHvesDnsYn81WgM0H0wQ52PbSZX5VdSvC359HAsVhwmP0a9nGR
+ybn3u1Q0WVFrBubQ4R0fjoLdZlgRV23oLZmHlgByF9YFzFT04v+16NdFe3vH7UP
ItF+RrzAYvVKAhV/VqNoXpMY49Un+8Fxn2Nq3WtU7MqHR90M8hFHZB0CC8Naba
SQnRtdUxh2Cvvcw=
-----END CERTIFICATE-----
AG Get Help  ^O Write Out  ^K Where Is    ^X Cut Text    ^J Justify
^X Exit      ^R Read File   ^\ Replace    ^U Uncut Text ^I To Spell

```

Figura 27 CA de producción.

Tanto el certificado de servidor como el CA deberán permanecer en el servidor para la correcta operación de este. Además, el CA deberá ser distribuido e instalado en los dispositivos que se vayan a conectar a la red mediante Wi-Fi.

3.4.4. Configuración de los Access Points

El paso final para poner en funcionamiento al Testbed es la configuración de los Access Points. Estos utilizan una versión modificada de OpenWRT, por lo que cuentan con una interfaz gráfica para la configuración de sus parámetros, así como un CLI al cual se puede acceder mediante SSH.

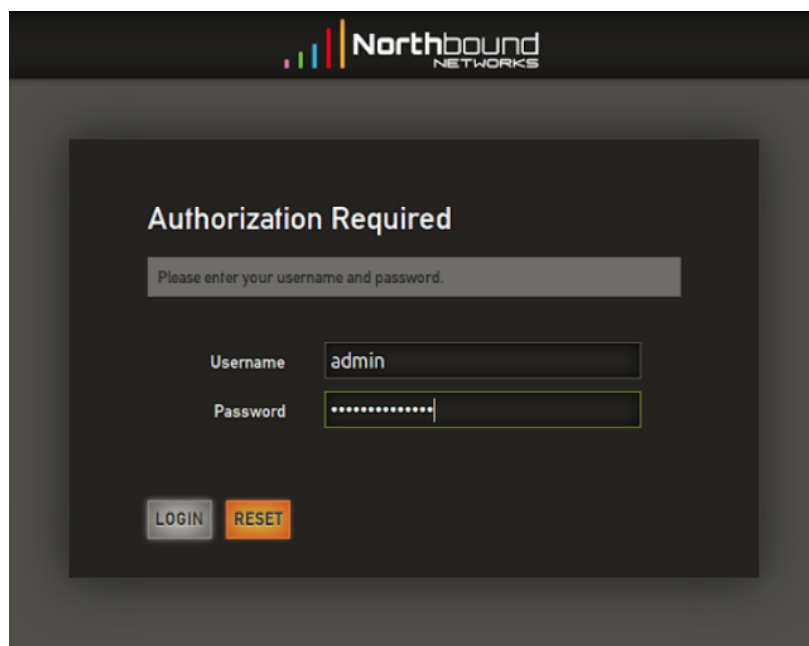


Figura 28 Ventana de autorización para la configuración de un Zodiac WX.

En primera instancia se deberá configurar las interfaces Ethernet de los Access Points para que se encuentren en la misma subred que el servidor:

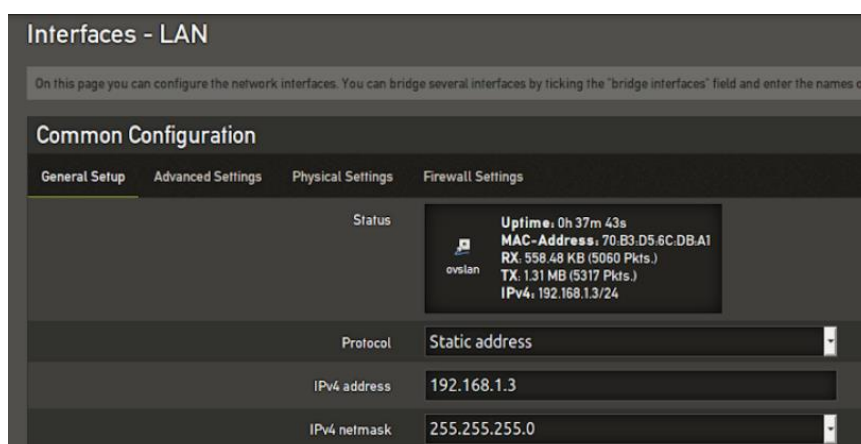
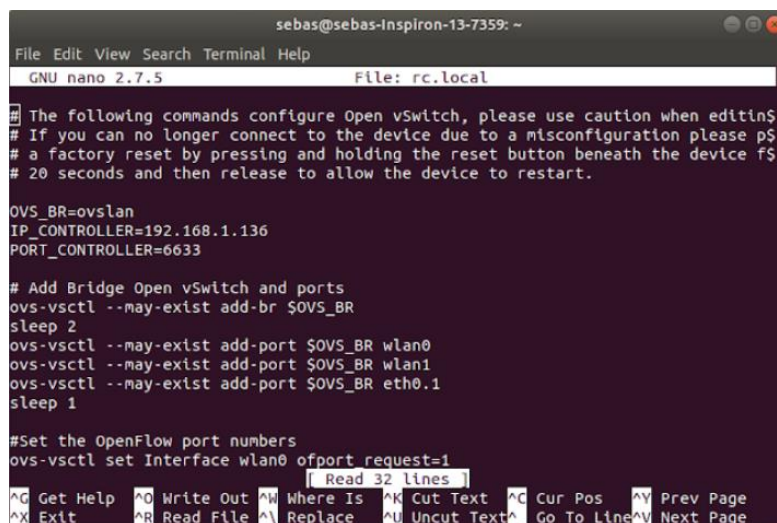


Figura 29 Configuración de IP para las interfaces cableadas.

Después, se debe configurar el archivo de inicialización de Open vSwitch `/etc/rc.local` para que los Access Points puedan reconocer al controlador, para lo cual se debe proporcionar la IP estática del controlador y el puerto TCP por el que se comunicarán:



```
sebas@sebas-inspiron-13-7359: ~
File Edit View Search Terminal Help
GNU nano 2.7.5 File: rc.local
# The following commands configure Open vSwitch, please use caution when editing
# If you can no longer connect to the device due to a misconfiguration please p
# a factory reset by pressing and holding the reset button beneath the device f
# 20 seconds and then release to allow the device to restart.

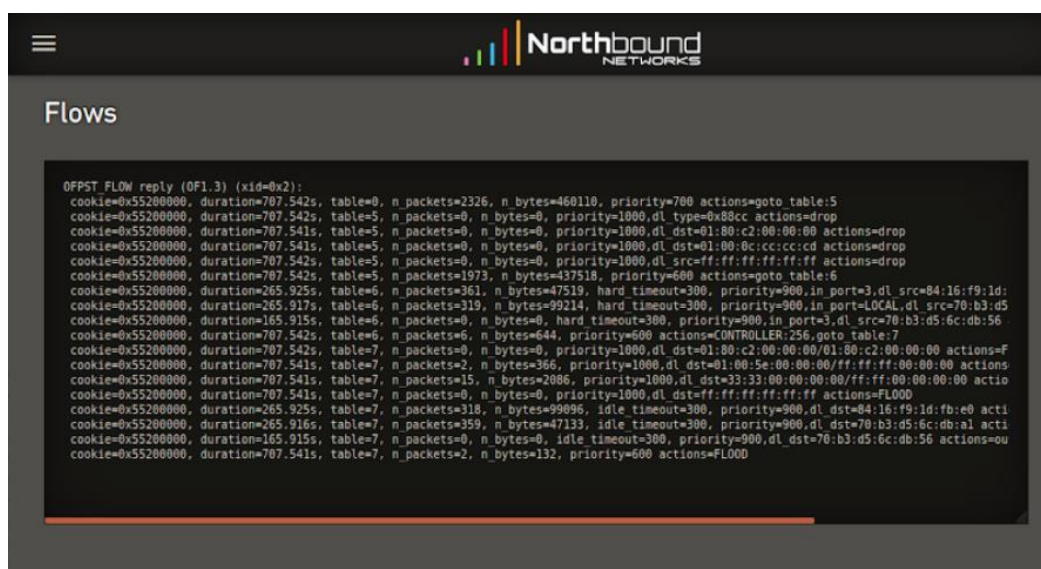
OVS_BR=ovs1an
IP_CONTROLLER=192.168.1.136
PORT_CONTROLLER=6633

# Add Bridge Open vSwitch and ports
ovs-vsctl --may-exist add-br $OVS_BR
sleep 2
ovs-vsctl --may-exist add-port $OVS_BR wlan0
ovs-vsctl --may-exist add-port $OVS_BR wlan1
ovs-vsctl --may-exist add-port $OVS_BR eth0.1
sleep 1

#Set the OpenFlow port numbers
ovs-vsctl set Interface wlan0 ofport request=1
Read 32 lines
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^C Cur Pos ^Y Prev Page
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^_ Go To Line ^V Next Page
```

Figura 30 Script de inicialización de Open vSwitch.

Los demás parámetros están configurados para que funcionen con las interfaces inalámbricas y puertos Ethernet del Access Point, por lo que se dejan en sus valores por defecto. Se puede comprobar que existe conexión con el controlador verificando los flujos instalados en los Access Points. Normalmente, si el Access Point no está trabajando como un switch OpenFlow, no existen flujos instalados. Por lo tanto, se puede verificar en la ventana de flujos que, en efecto, ryu ha instalado flujos que permiten la comunicación de los dispositivos conectados a la red:



```

OFFST_FLOW reply (OF1.3) (xid=0x2):
cookie=0x55200000, duration=787.542s, table=0, n_packets=2326, n_bytes=460118, priority=700 actions=goto table:5
cookie=0x55200000, duration=787.542s, table=5, n_packets=0, n_bytes=0, priority=1000, dl_type=0x88cc actions=drop
cookie=0x55200000, duration=787.541s, table=5, n_packets=0, n_bytes=0, priority=1000, dl_dst=01:80:c2:00:00:00 actions=drop
cookie=0x55200000, duration=787.541s, table=5, n_packets=0, n_bytes=0, priority=1000, dl_dst=01:00:0c:cc:cc:cd actions=drop
cookie=0x55200000, duration=787.542s, table=5, n_packets=0, n_bytes=0, priority=1000, dl_src=ff:ff:ff:ff:ff:ff actions=drop
cookie=0x55200000, duration=787.542s, table=5, n_packets=1973, n_bytes=437518, priority=600 actions=goto table:6
cookie=0x55200000, duration=265.925s, table=6, n_packets=361, n_bytes=47519, hard_timeout=300, priority=900, in_port=3, dl_src=84:16:f9:1d:
cookie=0x55200000, duration=265.917s, table=6, n_packets=319, n_bytes=99214, hard_timeout=300, priority=900, in_port=LOCAL, dl_src=70:b3:d5
cookie=0x55200000, duration=165.915s, table=6, n_packets=0, n_bytes=0, hard_timeout=300, priority=900, in_port=3, dl_src=70:b3:d5:6c:db:56
cookie=0x55200000, duration=787.542s, table=6, n_packets=6, n_bytes=644, priority=600 actions=CONTROLLER:256,goto table:7
cookie=0x55200000, duration=787.542s, table=7, n_packets=0, n_bytes=0, priority=1000, dl_dst=01:80:c2:00:00/01:80:c2:00:00:00 actions=F
cookie=0x55200000, duration=787.541s, table=7, n_packets=2, n_bytes=366, priority=1000, dl_dst=01:00:5e:00:00/ff:ff:ff:00:00:00 actions
cookie=0x55200000, duration=787.541s, table=7, n_packets=15, n_bytes=2086, priority=1000, dl_dst=33:33:00:00:00/ff:ff:00:00:00:00 actio
cookie=0x55200000, duration=787.541s, table=7, n_packets=0, n_bytes=0, priority=1000, dl_dst=ff:ff:ff:ff:ff:ff actions=FL000
cookie=0x55200000, duration=265.925s, table=7, n_packets=318, n_bytes=99098, idle_timeout=300, priority=900, dl_dst=84:16:f9:1d:fb:e0 acti
cookie=0x55200000, duration=265.916s, table=7, n_packets=359, n_bytes=47133, idle_timeout=300, priority=900, dl_dst=70:b3:d5:6c:db:d1 acti
cookie=0x55200000, duration=165.915s, table=7, n_packets=0, n_bytes=0, idle_timeout=300, priority=900, dl_dst=70:b3:d5:6c:db:56 actions=ou
cookie=0x55200000, duration=787.541s, table=7, n_packets=2, n_bytes=132, priority=600 actions=FL000

```

Figura 31 Flujos OpenFlow instalados en el Zodiac WX.

Después, se debe configurar la seguridad de las interfaces inalámbricas para que los usuarios se puedan autenticar únicamente mediante el servidor FreeRADIUS. Para esto se debe seleccionar WPA2-EAP como método de autenticación e ingresar la información del servidor FreeRADIUS:

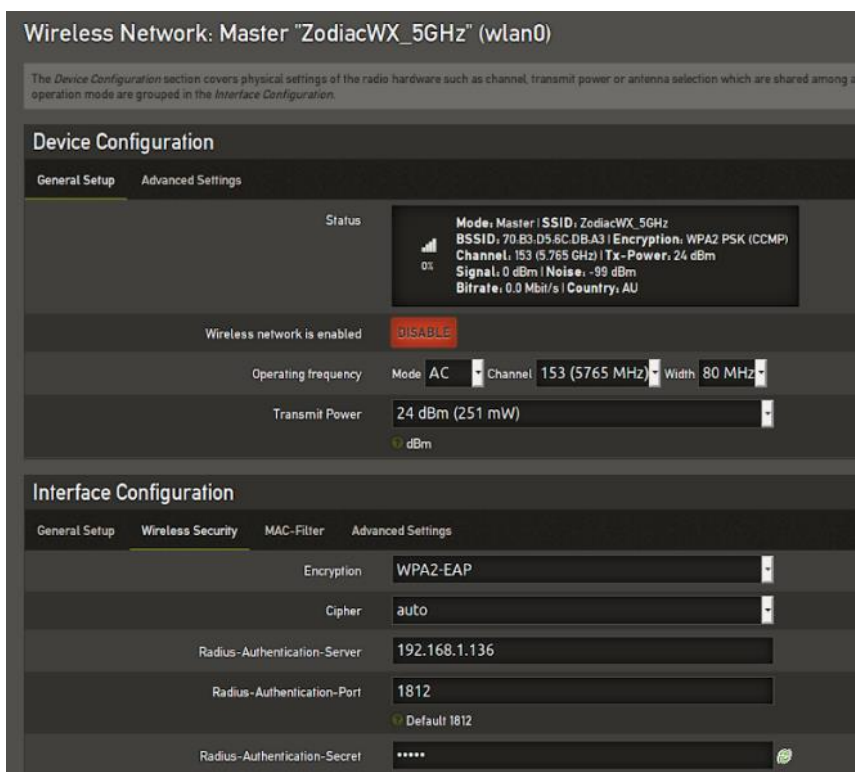


Figura 32 Configuración de seguridad inalámbrica para la interfaz de 5 GHz.

Con estos cambios el Testbed se encuentra listo para las pruebas. Como paso final se puede verificar que todos los componentes funcionen juntos, para lo cual se conecta un dispositivo a la red inalámbrica y se observan las salidas de los terminales:

```

sebas@sebas-Inspiron-13-7359: ~
File Edit View Search Terminal Help
(8) } # post-auth = noop
(8) Sent Access-Accept Id 8 from 192.168.1.136:1812 to 192.168.1.3:58420 length
0
(8) MS-MPPE-Recv-Key = 0xa879802962bc4870d54d645e21bf2b32fa12e56698b03faff42d6
e724f61a0f9
(8) MS-MPPE-Send-Key = 0x972bba49004644932f71a93743d8e8ea5045d340dbf0a64a75775
1ddb616a392
(8) EAP-Message = 0x03090004
(8) Message-Authenticator = 0x00000000000000000000000000000000
(8) User-Name = "Sebastian"
(8) Finished request
Waking up in 4.7 seconds.
(0) Cleaning up request packet ID 0 with timestamp +11
(1) Cleaning up request packet ID 1 with timestamp +11
Waking up in 0.1 seconds.
(2) Cleaning up request packet ID 2 with timestamp +11
(3) Cleaning up request packet ID 3 with timestamp +11
(4) Cleaning up request packet ID 4 with timestamp +11
(5) Cleaning up request packet ID 5 with timestamp +11
(6) Cleaning up request packet ID 6 with timestamp +11
(7) Cleaning up request packet ID 7 with timestamp +11
(8) Cleaning up request packet ID 8 with timestamp +11
Ready to process requests

```

Figura 33 Autenticación exitosa del usuario por parte de FreeRadius.

```

sebas@sebas-Inspiron-13-7359: ~
File Edit View Search Terminal Help
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0x2816b68c_0FPSwltc
hFeatures(auxiliary_id=0,capabilities=79,datapath_id=123917682138017,n_buffers=2
56,n_tables=254)
move onto main node
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 123917682137942, 4294967294, 70:b3:d5:6c:db:56 after 13 hits
Unlearned 123917682137942, 3, 84:16:f9:1d:fb:e0 after 21 hits
Learned 123917682138017, 3, 84:16:f9:1d:fb:e0
EVENT ofp_event->SS2Core EventOFPPacketIn
EVENT ofp_event->SS2Core EventOFPPacketIn
Learned 123917682138017, 4294967294, 70:b3:d5:6c:db:a1
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 123917682138017, 3, 84:16:f9:1d:fb:e0 after 1 hits
Unlearned 123917682138017, 4294967294, 70:b3:d5:6c:db:a1 after 0 hits
Learned 123917682138017, 2, 00:0a:f5:b6:5c:e8
EVENT ofp_event->SS2Core EventOFPPacketIn
Learned 123917682137942, 3, 00:0a:f5:b6:5c:e8
EVENT ofp_event->SS2Core EventOFPPacketIn
Unlearned 123917682138017, 2, 00:0a:f5:b6:5c:e8 after 0 hits
Unlearned 123917682137942, 3, 00:0a:f5:b6:5c:e8 after 0 hits
Learned 123917682137942, 3, 70:b3:d5:6c:db:a1
EVENT ofp_event->SS2Core EventOFPPacketIn
Learned 123917682138017, 3, 70:b3:d5:6c:db:56

```

Figura 34 Instalación de nuevos flujos por ryu dado el ingreso de un dispositivo nuevo a la red.

CAPÍTULO IV: METODOLOGÍA DE PRUEBAS

4.1. Descripción de los datos a ser obtenidos

Una vez concluida la implementación del Testbed, este se encuentra listo para realizar pruebas para la obtención de datos, los cuales permitirán analizar el desempeño de la red en cuanto a funcionamiento, calidad de servicio y tiempo de roaming.

Puntualmente, los parámetros de interés que serán obtenidos mediante las pruebas son:

- **Bitrate:** La velocidad de transmisión entre los extremos de las pruebas que se realizarán, medida en Kbps o Mbps.
- **Paquetes perdidos:** El porcentaje de paquetes que no fueron recibidos, respecto al total de paquetes enviados.
- **Delay:** La latencia que existe entre el envío de un paquete y la recepción del mismo, medido en milisegundos.
- **Jitter:** La variación de la latencia, también medida en milisegundos.
- **Tiempo de autenticación:** El tiempo entre el primer mensaje de autenticación enviado por el cliente inalámbrico y la respuesta satisfactoria por parte del servidor RADIUS, medido en milisegundos.

- Tiempo de roaming: El tiempo que toma el cliente inalámbrico en migrar de un Access Point a otro, medido en milisegundos. Este tiempo puede variar según el criterio bajo el cual se defina el tiempo de roaming, el cual será explorado a mayor profundidad una vez obtenidos los datos.

Tomando en cuenta estos parámetros. La metodología de pruebas debe ser diseñada de manera que se puedan medir todos estos confiablemente en una misma prueba, por lo que se decidió obtener los datos utilizando dos de las herramientas detalladas en el capítulo 3: D-ITG y las tarjetas de captura AirPcap NX combinadas con el software Wireshark.

D-ITG permite la inyección de tráfico desde un computador hacia otro, luego de la cual se puede analizar los logs de inyección para determinar el bitrate, delay, jitter y paquetes perdidos entre los dos extremos de la transmisión, lo cual indica el desempeño de la red en cuanto a estas métricas de QoS.

Por otro lado, las tarjetas de captura AirPcap NX permiten el monitoreo de todo el tráfico 802.11 en un canal, ya sean tramas de datos, gestión o control. Mediante el uso de dos tarjetas en conjunto con Wireshark se puede capturar simultáneamente el tráfico de dos canales, tanto en 2.4 como en 5 GHz. De esta manera se puede observar todos los paquetes intercambiados entre el cliente inalámbrico y los dos Access Points, los cuales por estar en espacios adyacentes deberán trabajar en dos canales diferentes para evitar causar interferencia entre ellos.

Source	Destination	Protocol	Length	Info
IeeeRegi_6c:db:58 (71:b3:d5:6c:db:58) -	HonHaiPr_6a:ca:0f (f8:da:0c:6a:ca:0f) -	802.11	46	Request-to-send, Flags=.....C
IeeeRegi_6c:db:58 (71:b3:d5:6c:db:58) -	HonHaiPr_6a:ca:0f (f8:da:0c:6a:ca:0f) -	802.11	46	Request-to-send, Flags=.....C
Northbou_0b:a3	HonHaiPr_6a:ca:0f	TLSv1..	129	Change Cipher Spec, Encrypted Handshake Message
HonHaiPr_6a:ca:0f	Northbou_0b:a3	TLSv1..	147	Application Data
	HonHaiPr_6a:ca:0f (f8:da:0c:6a:ca:0f) -	802.11	48	Acknowledgement, Flags=.....C
IeeeRegi_6c:db:58 (71:b3:d5:6c:db:58) -	HonHaiPr_6a:ca:0f (f8:da:0c:6a:ca:0f) -	802.11	46	Request-to-send, Flags=.....C
IeeeRegi_6c:db:58 (71:b3:d5:6c:db:58) -	HonHaiPr_6a:ca:0f (f8:da:0c:6a:ca:0f) -	802.11	46	Request-to-send, Flags=.....C
Northbou_0b:a3	HonHaiPr_6a:ca:0f	EAP	72	Success
Northbou_0b:a3	HonHaiPr_6a:ca:0f	EAPOL	185	Key (Message 1 of 4)
HonHaiPr_6a:ca:0f	Northbou_0b:a3	EAPOL	203	Key (Message 2 of 4)
	HonHaiPr_6a:ca:0f (f8:da:0c:6a:ca:0f) -	802.11	48	Acknowledgement, Flags=.....C
Northbou_0b:a3	HonHaiPr_6a:ca:0f	EAPOL	219	Key (Message 3 of 4)
HonHaiPr_6a:ca:0f	Northbou_0b:a3	EAPOL	163	Key (Message 4 of 4)

Figura 35 Paquetes 802.11 capturados con las tarjetas AirPcap NX.

Mediante la captura de estos paquetes se puede determinar los tiempos exactos en los cuales el cliente inalámbrico transmite paquetes tales como los de datos, autenticación, asociación, desasociación, entre otros; e identificar a cuál de los dos Access Points va dirigido cada paquete. De esta manera se puede calcular el tiempo de interrupción de la inyección de tráfico, tiempo de asociación, tiempo de autenticación y de acuerdo con el criterio que se tenga de roaming obtener un valor del tiempo de roaming, el cual es el enfoque principal de este estudio.

4.2. Ubicación y configuración de potencia de los equipos

Al ser este un estudio comparativo del Testbed con una porción de la red actualmente implementada en la Universidad, se debe replicar las condiciones físicas de la red actual con la mayor fidelidad posible, de manera que la comparación sea justa. Los Access Points que forman parte del Testbed fueron elegidos teniendo este criterio en cuenta, pero adicionalmente se debe considerar los niveles de potencia con los que estos serán configurados.

Los algoritmos que utilizan los clientes Wi-Fi para determinar cuándo hacer roaming se basan en el indicador de potencia de la señal recibida (RSSI) para la toma de

decisiones. Es por este motivo que al hacer un estudio comparativo que involucre roaming se debe tener el mayor cuidado posible configurando los niveles de potencia en los Access Points.

Debido a los múltiples factores físicos (Directividad y ganancia de antenas de transmisión y recepción, interferencia, desvanecimiento, etc.) que pueden afectar a la potencia de la señal en el espacio donde se desarrollará el estudio, la mejor estrategia para poder determinar y replicar la potencia en la red de la Universidad es realizar un estudio de mapas de calor.

Mediante los mapas de calor se puede obtener un buen aproximado de los niveles de potencia de uno o más SSIDs en un área geográfica de manera simultánea para todos los canales tanto en 2.4 como 5 GHz, observando el nivel de recepción en puntos de interés. De esta manera, se puede usar los mapas de calor para determinar la cobertura de un Access Point en particular y recrear la misma al momento de configurar los Access Points del Testbed.

Teniendo las consideraciones antes mencionadas en cuenta, se procedió a realizar un survey o inspección técnica del sitio donde se implementará el Testbed para obtener los mapas de calor de los SSIDs correspondientes a la red de la Universidad. Este survey se realizó utilizando el software Acrylic Wi-Fi HeatMaps con una licencia estudiantil. Este software permite la creación de mapas de calor mediante la medición de la potencia recibida de un Access Points en varios puntos del espacio.

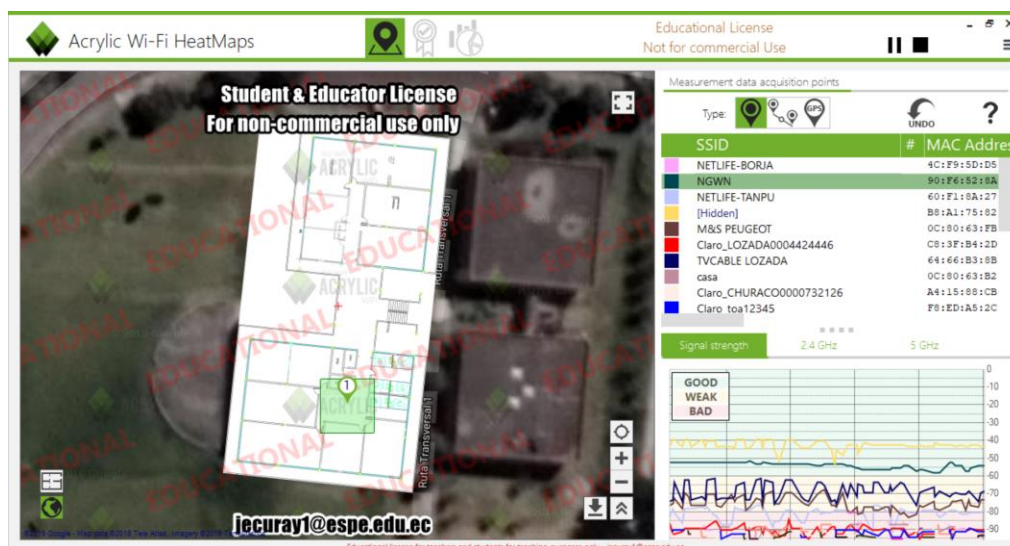


Figura 36 Acrylic Wi-Fi Heatmaps realizando un survey.

Teniendo el software listo, junto con los planos del edificio donde se realizarán las pruebas, obtenidos gracias a la colaboración del Departamento de Desarrollo Físico de la Universidad, se procedió a realizar el survey del área. Una vez tomadas las mediciones en distintos puntos a lo largo del trayecto en cual se realizarán las pruebas de roaming, se filtran los SSIDs de manera que solo se grafique la cobertura de los Access Points relevantes al estudio, tanto en las frecuencias de 2.4 como 5 GHz.

Así entonces, los mapas de calor obtenidos para 2.4 GHz son:

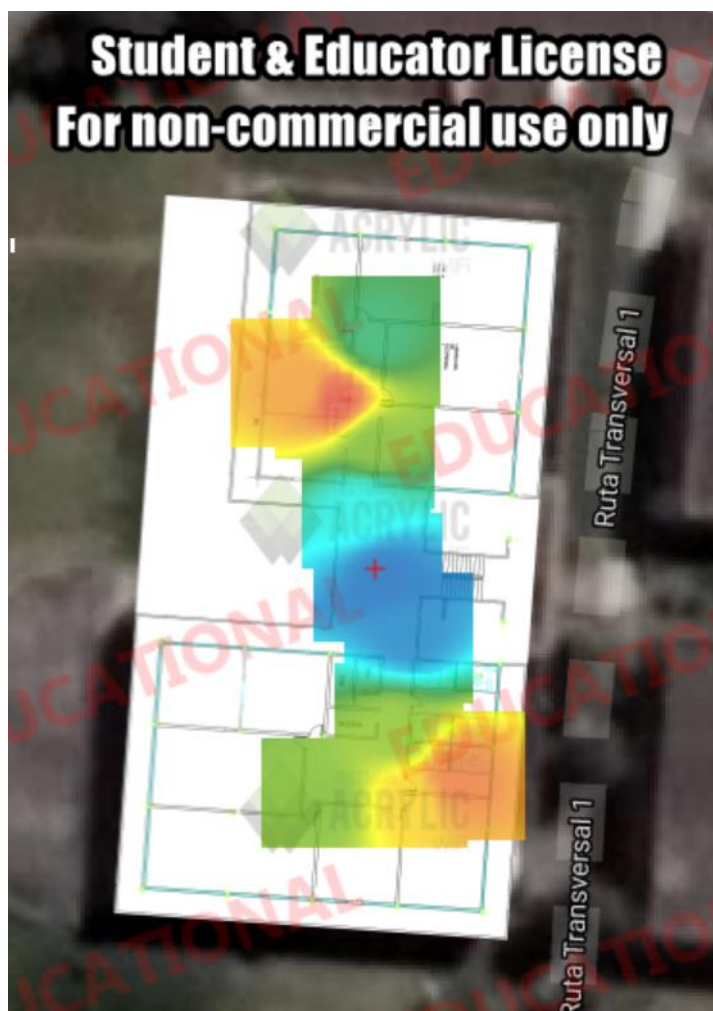


Figura 37 Mapa de calor para el SSID Eduroam en 2.4 GHz.

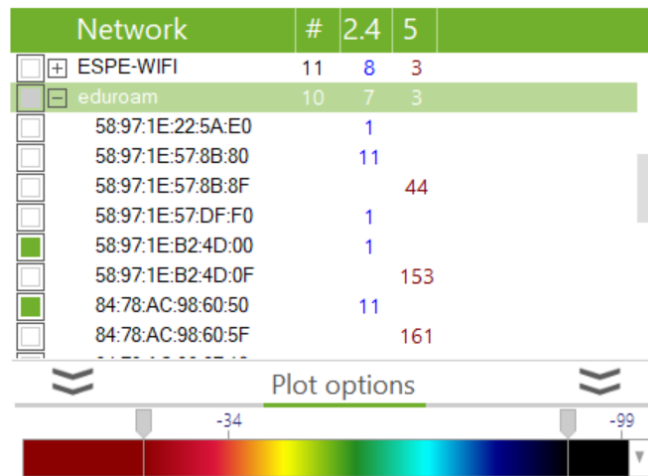


Figura 38 Filtrado de direcciones MAC en 2.4 GHz de los APs relevantes y escala de potencia recibida en dBm.

Y para 5 GHz:



Figura 39 Mapa de calor para el SSID Eduroam en 5 GHz.

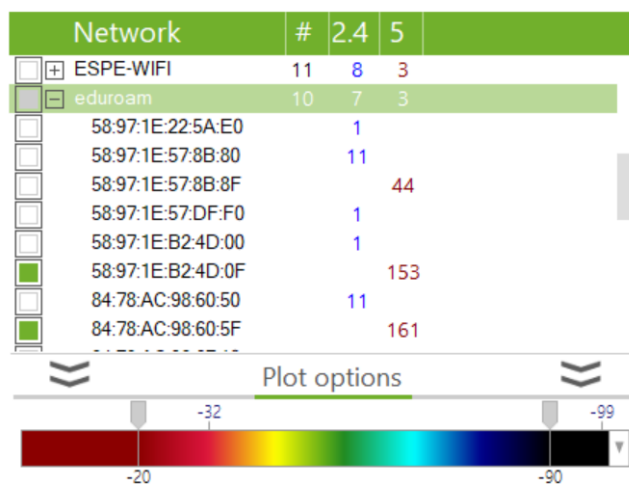


Figura 40 Filtrado de direcciones MAC en 5 GHz de los APs relevantes y escala de potencia recibida en dBm.

Como se puede observar, los dos APs que cubren el espacio geográfico donde se realizarán las pruebas operan en los canales 1 y 11 (2412 y 2462 MHz) para la frecuencia de 2.4 GHz; y en los canales 153 y 161 (5765 y 5805 MHz) para 5 GHz.

Los mapas de calor dan una buena aproximación de cómo se distribuye la cobertura del SSID Eduroam en el edificio. Con esta información se puede proceder a configurar los Access Points que conforman el Testbed.

Para esta configuración, los Access Points de la red de la Universidad deberán ser temporalmente desconectados, ya que podrían causar interferencia en la red Wi-Fi del Testbed debido a que esta operará en los mismos canales. Se colocan los nuevos Access Points en el mismo lugar y orientación que los de la Universidad y se procede a realizar surveys con el software Acrylic probando diferentes valores de potencia hasta obtener el resultado más parecido al estudio anterior.

De esta manera, se obtuvieron los siguientes mapas de calor para la red Wi-Fi del Testbed:

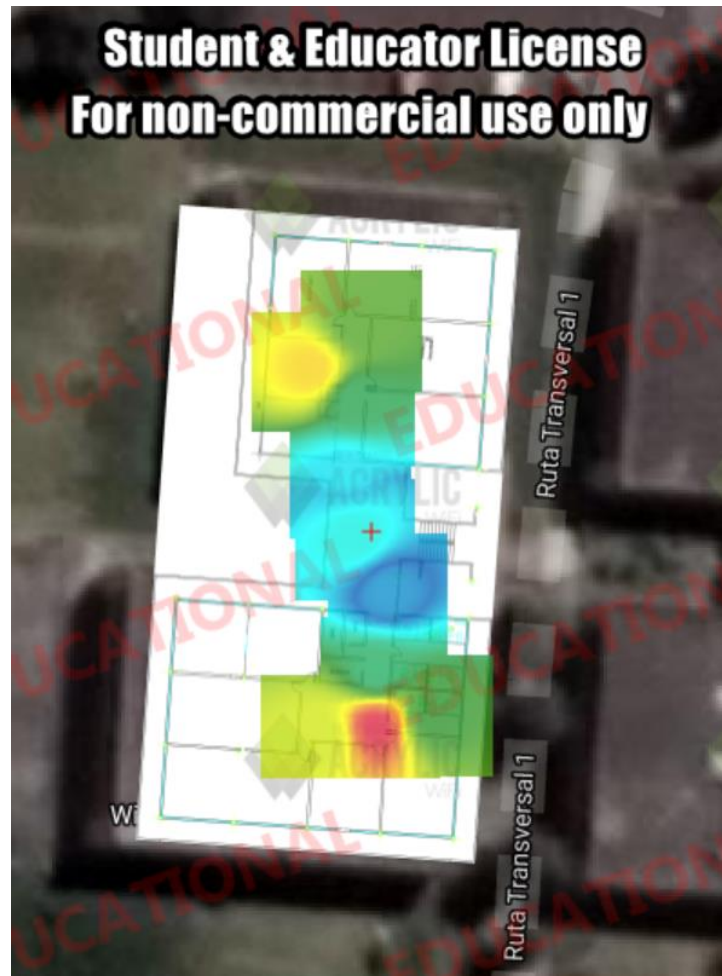


Figura 41 Mapa de calor para la red inalámbrica del Testbed en 2.4 GHz.

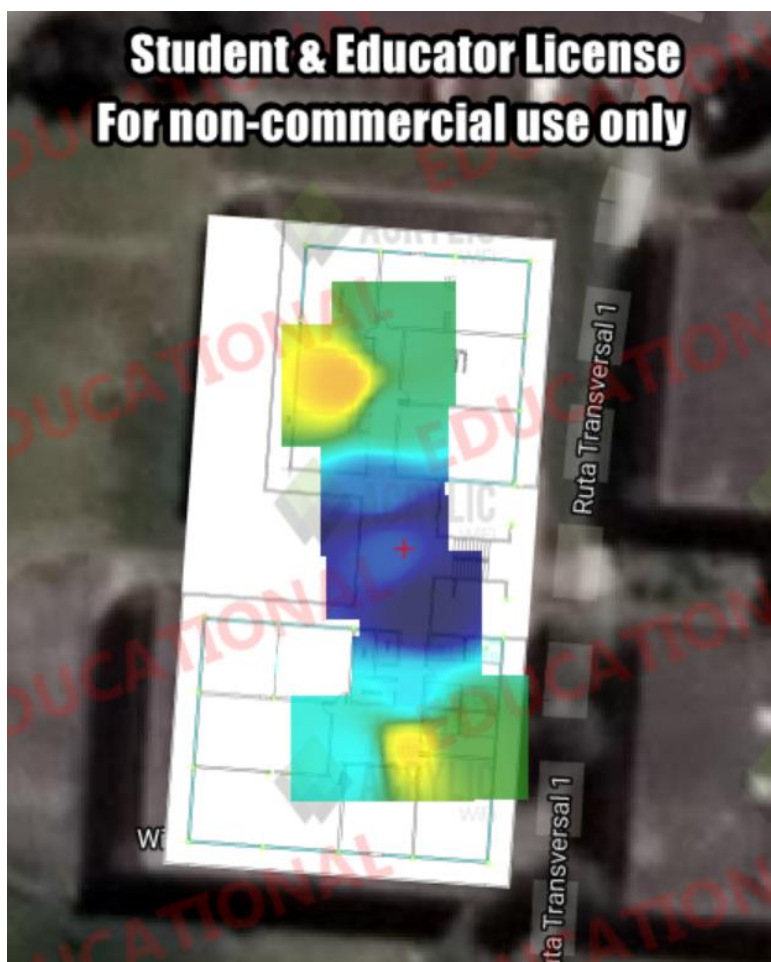


Figura 42 Mapa de calor para la red inalámbrica del Testbed en 2.4 GHz.

En ambos casos, se puede apreciar que la potencia en la cercanía de los Access Points no excede los -35 dBm, caso que sí se da en la red de la Universidad. Esto no es de gran importancia debido a que, como se explicará posteriormente, se limitará la tasa de transmisión usada en las pruebas de inyección de tráfico y los puntos de interés son las zonas azules donde la recepción es mínima y ocurrirá el roaming. En estos puntos de interés las potencias son casi idénticas entre las dos redes, con una variación aproximada de 2 dB.

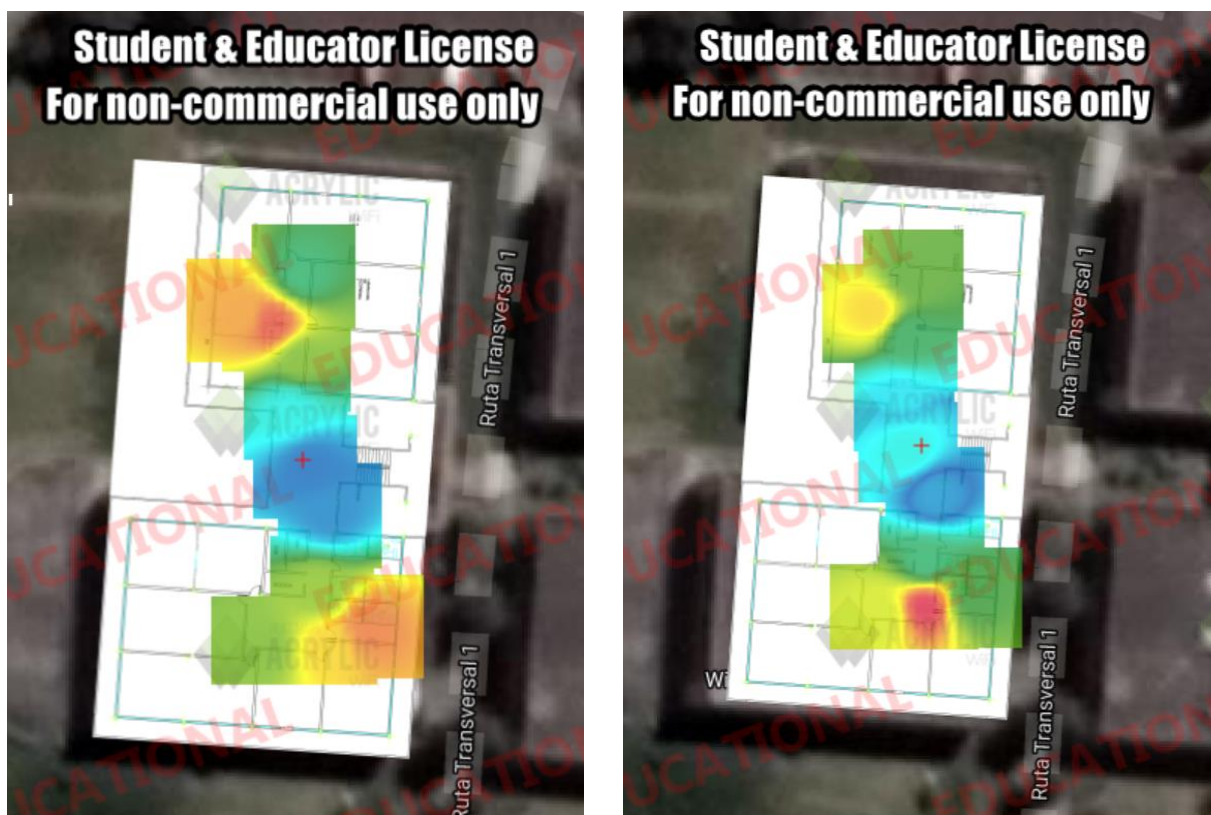


Figura 43 Izquierda: Mapa de calor para la red Eduroam en 2.4 GHz. Derecha: Mapa de calor para el Testbed en 2.4 GHz.

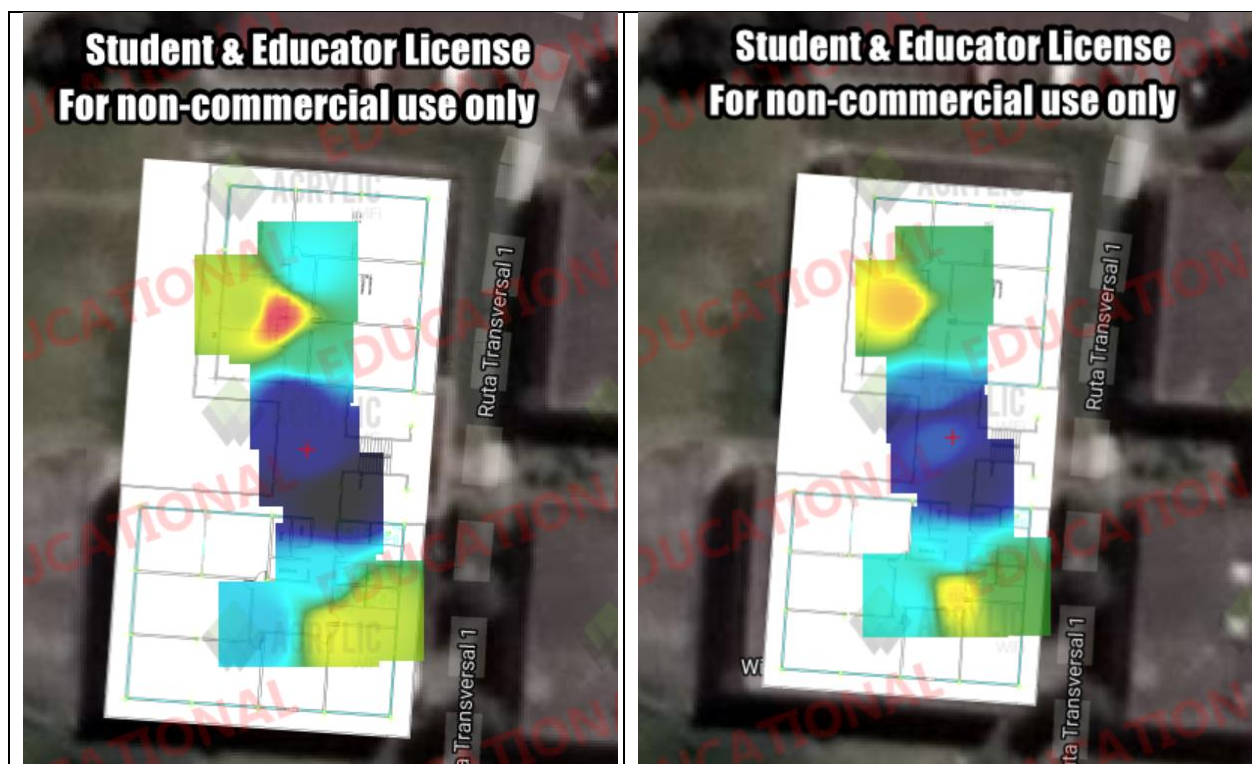


Figura 44 Izquierda: Mapa de calor para la red Eduroam en 5 GHz. Derecha: Mapa de calor para el Testbed en 5 GHz.

De esta manera, los resultados de configuración de potencia y ubicación física de los Access Point son satisfactorios y se puede proceder a definir la metodología que se utilizará en las pruebas.

4.3. Descripción de las pruebas

Las pruebas a realizar están basadas en obtener las métricas de calidad de servicio y los tiempos de roaming antes descritos. Para ello, se debe diseñar una metodología que permita obtener todos estos valores confiable y simultáneamente, de

manera que se pueda analizar el desempeño de la red durante el periodo en el que un cliente inalámbrico hace roaming.

Como ya se mencionó, se utilizará el software D-ITG para inyectar tráfico y obtener valores de las cuatro métricas de QoS propuestas. Adicionalmente a esto, se capturará todo el tráfico Wi-Fi utilizando Wireshark en conjunto con las tarjetas de captura AirPcap NX; esto permitirá la captura y ubicación en el tiempo de cada paquete Wi-Fi enviado durante la inyección de tráfico. En base a las etiquetas de tiempo en los paquetes relevantes al proceso de roaming (paquetes de autenticación, asociación, des-asociación, datos, validación con el servidor Radius), se puede determinar el tiempo que tarda el cliente inalámbrico en hacer roaming y comparar dichos tiempos entre los escenarios descritos a continuación.

4.3.1. Escenarios de pruebas

Los Access Points utilizados para el Testbed pueden trabajar con un amplio rango de tecnologías, no solo limitados a funcionar como switches inalámbricos Openflow. Específicamente, estos tienen la capacidad de funcionar tanto con un controlador Openflow como independientemente, es decir, como Access Points 802.11 comunes y corrientes. Además, soportan varios protocolos de autenticación para los clientes inalámbricos tales como las versiones 1 y 2 de WPA-PSK y WPA Enterprise. El objeto principal de estudio de este proyecto es un Testbed basado en la combinación de Openflow con autenticación mediante un servidor Radius (WPA2 Enterprise), sin embargo, es posible implementar cualquiera de estas tecnologías, así como

combinaciones de dichas tecnologías, para poder obtener más datos que ayudarán a tener un mejor entendimiento de las Redes Definidas por Software.

Es debido a esta flexibilidad que se decidió implementar 4 escenarios, los cuales difieren en la combinación de tecnologías utilizadas en los Access Points en cuanto a autenticación y el uso de Openflow. Estos permitirán obtener una mayor cantidad de datos para observar cómo afecta Openflow al comportamiento y desempeño de la red.

Los 4 escenarios son los siguientes:

Tabla 2

Escenarios de pruebas del Testbed

	Con Controlador Openflow	Sin Controlador Openflow
WPA2 Enterprise	Escenario 1	Escenario 2
WPA2-PSK	Escenario 3	Escenario 4

Adicionalmente se debe tener en cuenta que estos 4 escenarios aplican tanto para 2.4 como para 5 GHz, por lo que en total se obtendrán datos de 8 escenarios distintos.

4.3.2. Parámetros de la inyección de tráfico

Para cada uno de los 8 escenarios los parámetros configurados para la inyección de tráfico serán idénticos, y fueron elegidos en base a los siguientes criterios:

- Tiempo: Se tiene que definir un periodo adecuado para la inyección de manera que permita que el cliente Wi-Fi tenga el suficiente tiempo para

hacer roaming mientras se mueve a una velocidad constante desde un punto cercano al primer Access Point hacia el segundo.

- **Bitrate de la inyección:** Como se observó en los mapas de calor, se debe considerar que en el espacio entre los dos Access Points el RSSI es bastante bajo (esto es particularmente notorio cuando se trabaja en 5 GHz debido a que la mayor frecuencia implica un menor alcance de las ondas electromagnéticas), lo que resulta en el ajuste dinámico de la tasa de datos por parte del protocolo. Debido a esto, se decidió utilizar un bitrate menor a la tasa base del protocolo más lento usado en el Testbed, el cual es 802.11n y su tasa base es de 1 Mbps (Intel Corporation, s.f.).
- **Protocolo de transporte:** D-ITG permite inyectar flujos utilizando TCP o UDP como protocolo de la capa de transporte. Dos métricas de interés de este estudio son el delay y la pérdida de paquetes, sobre las cuales influye el tipo de protocolo de transporte utilizado. Se quiere observar cómo afectan los diferentes escenarios a la pérdida de paquetes durante el roaming, y este valor sería disminuido si se utiliza TCP, por lo que se utilizará UDP para las inyecciones, evitando que la corrección de errores y retransmisión que podrían tener un impacto sobre el delay y la pérdida de paquetes.

Con estas consideraciones en mente, los parámetros con los que se configurará el computador que actuará como inyector en las pruebas es:

Tabla 3

Parámetros más relevantes configurados en la inyección de tráfico.

Métrica de delay	Unidireccional
Duración	60 segundos
Protocolo de transporte	UDP
Puerto de destino	8999
Paquetes/s	800
Tamaño de paquete	128 bytes
Bitrate	998.4 kb/s

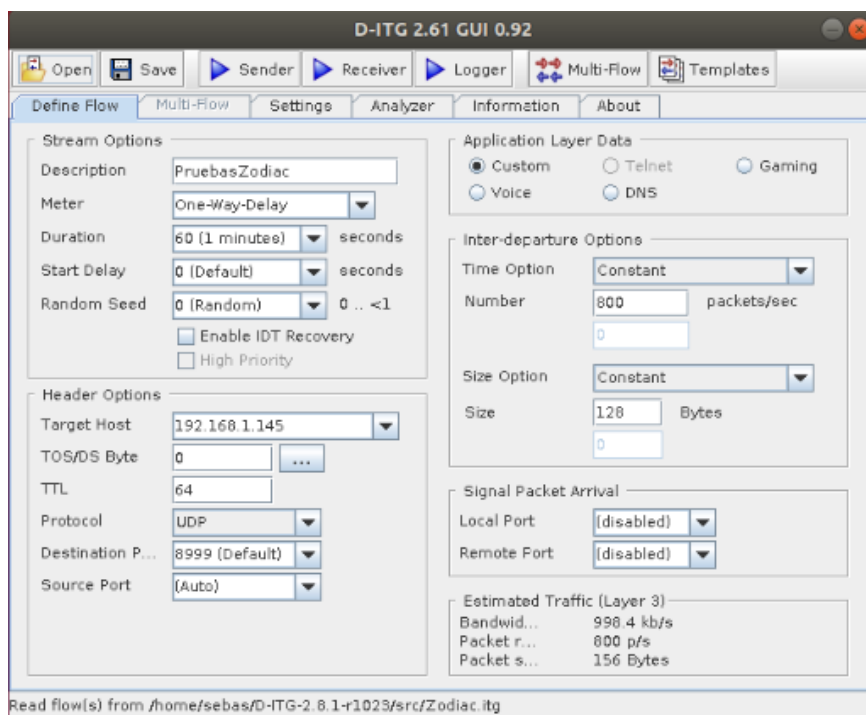


Figura 45 Parámetros configurados en D-ITG.

La razón del tiempo elegido será explicada en la siguiente sección.

4.3.3. Configuración de la captura de paquetes

El segundo software que se utilizará para recolectar datos es Wireshark. Este no requiere ningún tipo de configuración especial, sin embargo, se debe utilizar las tarjetas AirPcap NX como las interfaces de las cuales se capturará el tráfico. Usualmente al diseñar redes inalámbricas, dos Access Points adyacentes operarán en dos canales diferentes, de manera que no causen interferencia co-canal. Este es el caso en la red de la Universidad, y es por este motivo que se debe usar dos tarjetas de captura, una en cada canal, y se debe desplegar los paquetes capturados por ambas simultáneamente utilizando una interfaz agregada; esto permitirá la captura de paquetes en dos canales diferentes con una misma referencia de tiempo, lo cual será esencial para calcular el tiempo de roaming entre dos Access Points que operan en diferentes canales.

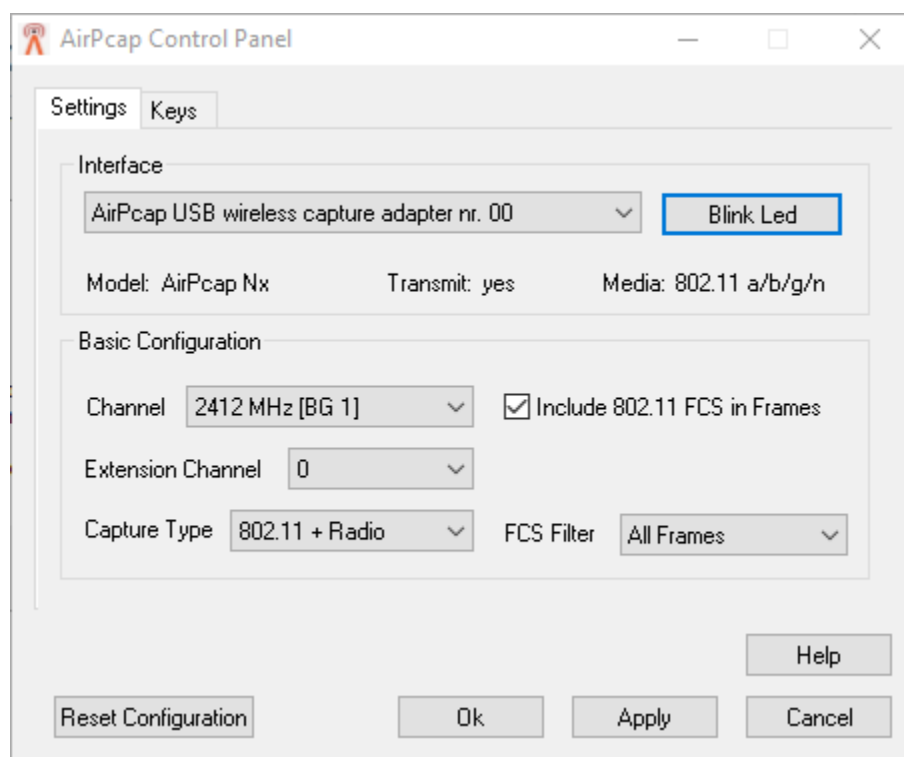


Figura 46 Panel de control de AirPcap, donde se puede seleccionar el canal en el cual se capturará el tráfico.

Entonces, para poder capturar y visualizar correctamente todo el tráfico involucrado en el roaming se utiliza la interfaz agregada para la captura en dos canales simultáneamente. Además de esto, debido a que las tarjetas capturan todo el tráfico en el canal, incluyendo de otras redes, se utilizó un filtro con la dirección MAC del cliente inalámbrico de manera que se pueda ignorar el tráfico que no corresponda al Testbed.

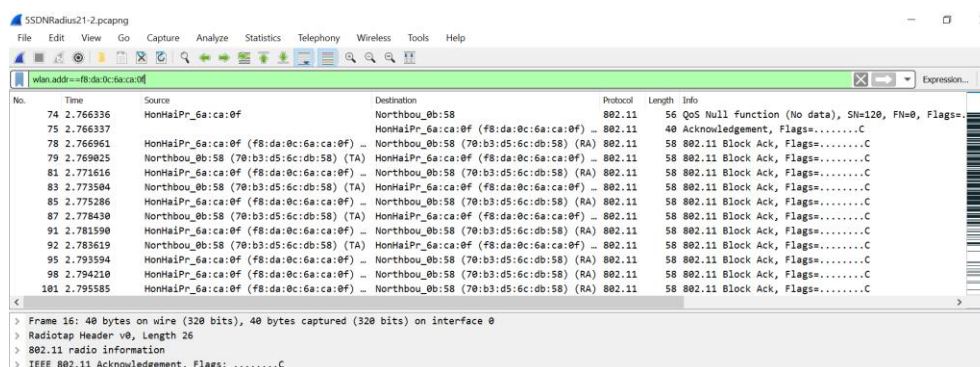


Figura 47 Wireshark mostrando parte del tráfico capturado en una de las pruebas, utilizando un filtro de dirección MAC.

4.4. Desarrollo de las pruebas

Teniendo clara la configuración del software a ser utilizado en las pruebas, el paso final es definir como se desarrollarán estas.

Se midió la distancia que comprende el recorrido de una persona entre los dos Access Points en 38 metros. Además de esto, se determinó que una velocidad adecuada para realizar las pruebas es aproximadamente 0.8 m/s, lo cual implica que se completaría un recorrido en aproximadamente 48 segundos. Es por este motivo que se optó por un tiempo de inyección de 60 segundos, el cual permite un margen de seguridad para asegurarse de que se complete exitosamente el proceso de roaming durante el recorrido.

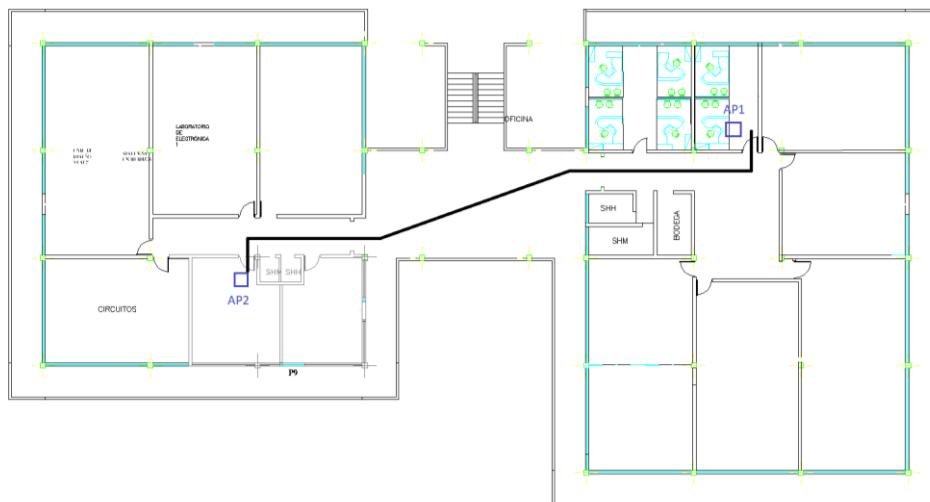


Figura 48 Recorrido para las pruebas de roaming entre los Access Points.

Debido a lo impredecible que puede ser el medio en el cual se realizarán las pruebas, es necesario realizar múltiples pruebas en cada escenario para poder promediar los datos y obtener valores más confiables. Por ello, se decidió realizar 5 recorridos por cada escenario, es decir, 5 pruebas de 60 segundos en las cuales el cliente Wi-Fi se desplazará a una velocidad constante de aproximadamente 0.8 m/s desde un Access Point hacia el otro en el trayecto establecido, todo mientras actúa como receptor de la inyección de tráfico.

Adicional a esto se realizó una prueba de control por cada escenario en la cual se inyectó tráfico de la misma manera, pero estando el cliente estático en la cercanía de uno de los dos Access Points. Teniendo los 4 escenarios antes descritos para dos bandas de frecuencia y 5 pruebas de roaming por cada uno más una de control da como resultado 40 pruebas de roaming, y 8 de control.

Finalmente, para observar el comportamiento del cliente inalámbrico mientras este se mueve y los niveles de potencia recibida de cada Access Point varían, se realizaron pruebas en 2.4 y 5 GHz en las cuales se siguió al cliente inalámbrico con las tarjetas de captura, de manera que se pueda graficar la caída y subida de las potencias del primer y segundo Access Point, respectivamente, y cómo influye esto en el tiempo en el que el cliente inalámbrico decide iniciar el proceso de roaming.

En 2.4 GHz, el gráfico que indica la variación de potencia y el instante en el que empieza el proceso de autenticación en el segundo Access Point es:

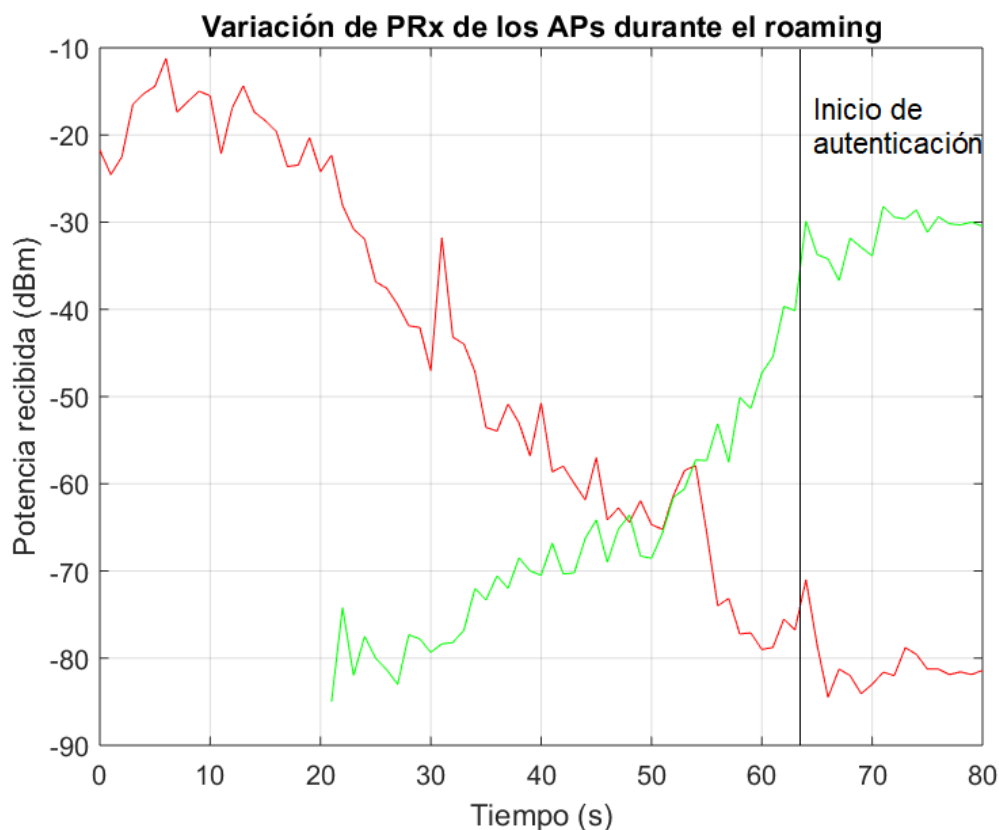


Figura 49 Variación de potencia recibida durante movimiento del cliente inalámbrico en 2.4 GHz.

Se puede observar que la autenticación comienza aproximadamente 8 segundos después de que el nivel de potencia recibida del segundo Access Point supera al primero. Un factor influyente en esto es el hecho de que los valores de la gráfica no fueron capturados directamente por el dispositivo que realiza el handoff entre los dos Access Points, sino por tarjetas de captura adyacentes a este que difieren en posición, ganancia de antenas, entre otros. No se puede realizar la captura en el mismo dispositivo que se conecta a los Access Points debido a que, para capturar paquetes 802.11 en una interfaz Wi-Fi, esta debe operar en modo monitor, lo cual no le permitirá conectarse a un Access Point.

Otro punto a considerar es el hecho de que el algoritmo que utiliza una tarjeta de interfaz Wi-Fi para decidir cómo realizar el proceso de roaming varía entre fabricantes, por lo que no se puede saber exactamente cuál es el criterio que usa para decidir iniciar el handoff.

Por otro lado, en 5 GHz se observa el siguiente comportamiento:

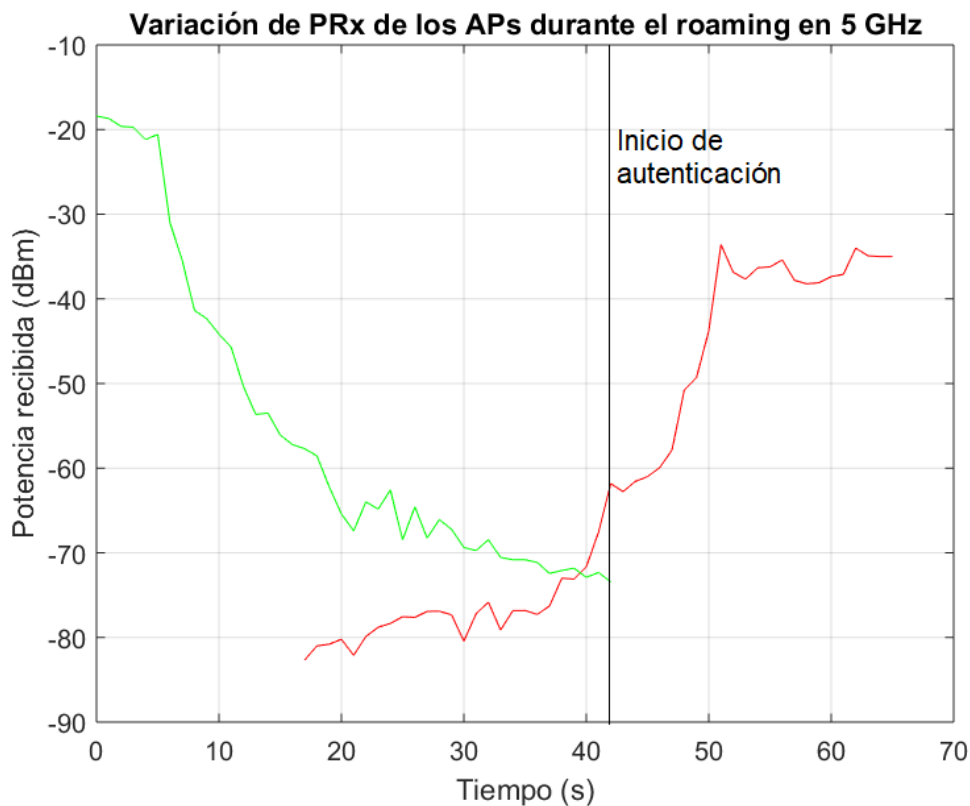


Figura 50 Variación de potencia recibida durante movimiento del cliente inalámbrico en 5 GHz.

En el caso de 5GHz se realizan las mismas observaciones que en la gráfica anterior, notando que esta vez el proceso de autenticación inicia más rápido con respecto al cruce de los niveles de potencia, teniendo en cuenta que una vez más, esta inconsistencia se puede explicar con la inhabilidad de utilizar la tarjeta de interfaz para capturar y conectarse simultáneamente.

Con esto queda descrita la manera como se realizarán las pruebas, por lo que se procede a realizarlas y analizar los resultados.

CAPÍTULO V: RESULTADOS

Las pruebas se realizan en dos bandas de redes inalámbricas, en las frecuencias de 2.4 GHz y 5 GHz. Para cada banda se realizaron dos análisis de comparación, el primero entre las arquitecturas de red SDN con seguridad WPA, SDN con servidor Radius, red tradicional con servidor Radius y red tradicional con seguridad WPA implementadas.

El segundo análisis de comparación es entre la mejor red SDN implementada (SDNRadius) y la red inalámbrica establecida en la infraestructura de los laboratorios de electrónica y telecomunicaciones de la Universidad de las Fuerzas Armadas - ESPE.

Las pruebas se enfocan en obtener indicadores de rendimiento de las redes inalámbricas a través de los valores promedio de los parámetros de calidad de servicio (delay, jitter, bitrate y paquetes perdidos), adicionalmente se compara el tiempo de autenticación y tiempo de roaming.

5.1. Pruebas de red Wi-Fi en 2,4 GHz

5.1.1. Comparación de arquitecturas de red implementadas en 2.4 GHz

En las pruebas de comparación de delay se obtiene que la red con el promedio más bajo es la red SDN con servidor Radius con 53,84 [ms], seguido de la red tradicional con servidor Radius con 75,03 [ms], la red SDN con seguridad WPA2-PSK con 124,15 [ms] y la red tradicional con seguridad WPA2-PSK con 185,66 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 4

Comparación de redes implementadas en parámetro: Delay en 2.4 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- SDN + Radius	28,104	39,747	127,325	31,736	43,016	53,109	53,840
2.- Radius	60,55	98,803	116,31	52,357	57,473	64,695	75,031
3.- SDN + WPA2	121,227	134,749	134,711	116,929	116,754	120,55	124,153
4.- WPA2	75,043	82,008	100,374	393,923	369,328	93,297	185,662

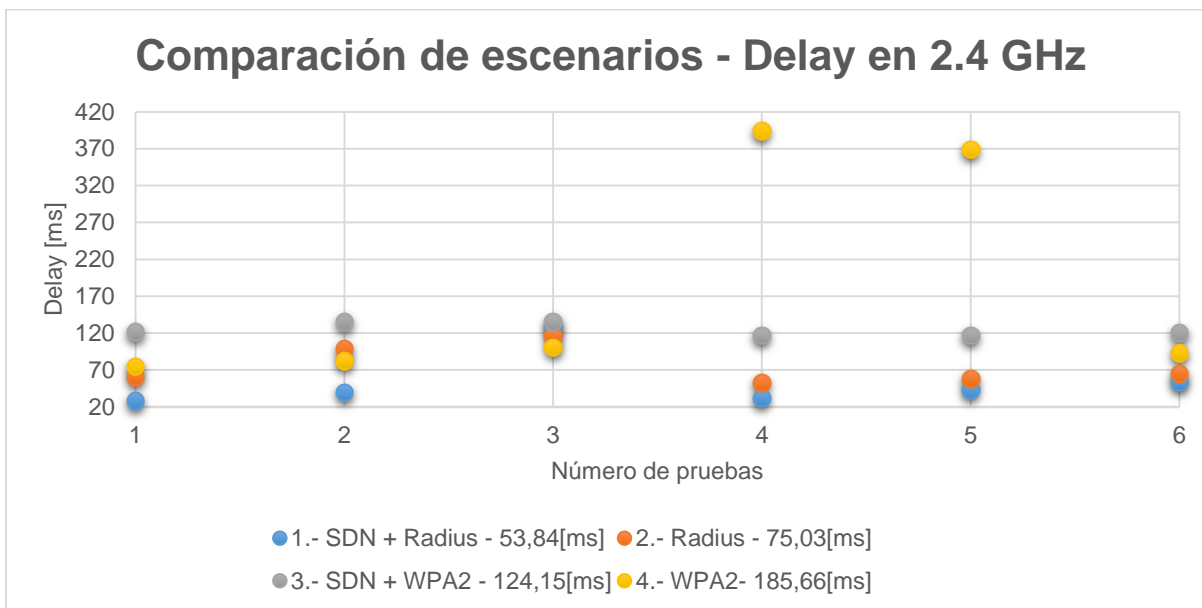


Figura 51 Resultados de comparación en delay de redes implementadas en 2.4 GHz.

En las pruebas de comparación de jitter se obtiene que la red con el promedio más bajo es la red SDN con seguridad WPA2-PSK con 1,51 [ms], seguido de la red tradicional con seguridad WPA2-PSK con 1,55 [ms], la red tradicional con servidor Radius con 1,56 [ms] y la red SDN con servidor Radius con 1,82 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 5

Comparación de redes implementadas en parámetro: Jitter en 2.4 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- SDN + WPA2	1,74	1,749	1,731	1,231	1,36	1,246	1,510
2.- WPA2	1,786	1,764	1,726	1,461	1,195	1,385	1,553
3.- Radius	1,562	1,675	1,636	1,604	1,458	1,406	1,557
4.- SDN + Radius	2,047	1,696	2,636	1,394	1,402	1,772	1,825

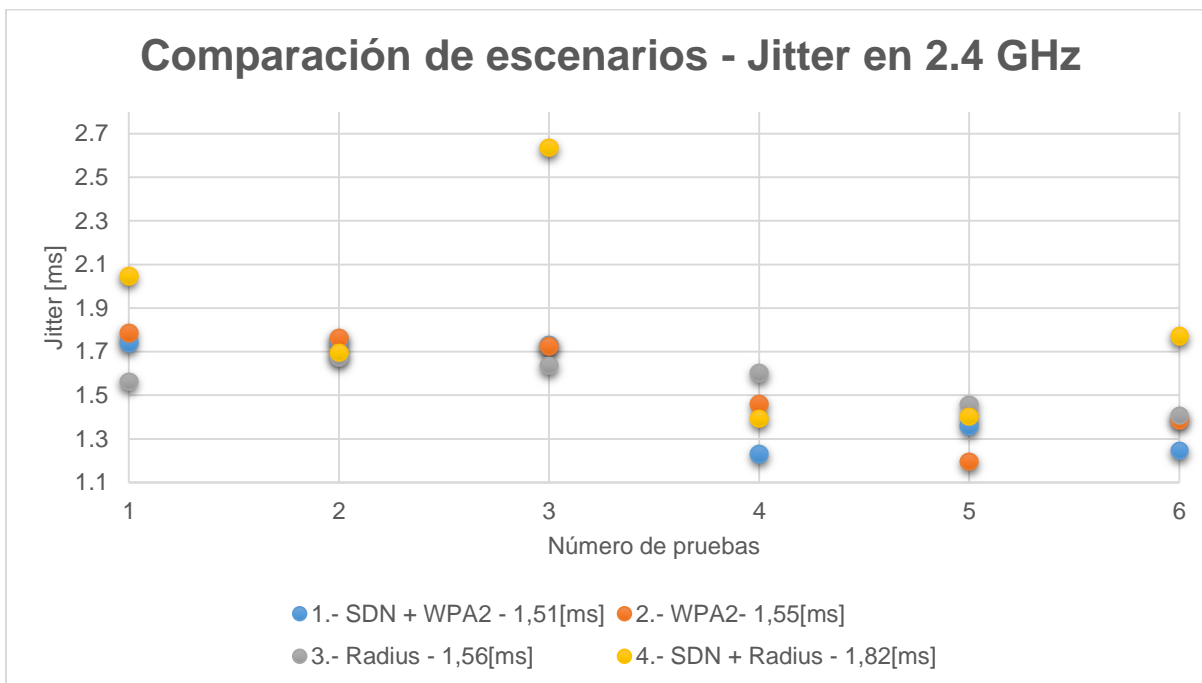


Figura 52 Resultados de comparación en jitter de redes implementadas en 2.4 GHz.

En las pruebas de comparación de bitrate se obtiene que la red con el promedio más alto es la red SDN con seguridad WPA2-PSK con 650,53 [kbps], seguido de la red tradicional con servidor Radius con 626,43 [kbps], la red tradicional con seguridad WPA2-PSK con 621,49 [kbps] y la red SDN con servidor Radius con 543,40 [kbps]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 6

Comparación de redes implementadas en parámetro: Bitrate en 2.4 GHz

Arquitectura de red	1 [kbps]	2 [kbps]	3 [kbps]	4 [kbps]	5 [kbps]	6 [kbps]	Promedio total [kbps]
1.- SDN + WPA2	653,133	676,020	676,517	648,740	627,069	621,691	650,529
2.- Radius	663,862	665,605	662,028	544,920	661,043	561,104	626,427

3.- WPA2	623,037	663,895	677,535	572,017	651,899	540,563	621,491
4.- SDN + Radius	542,098	610,304	427,023	645,863	573,257	461,830	543,396

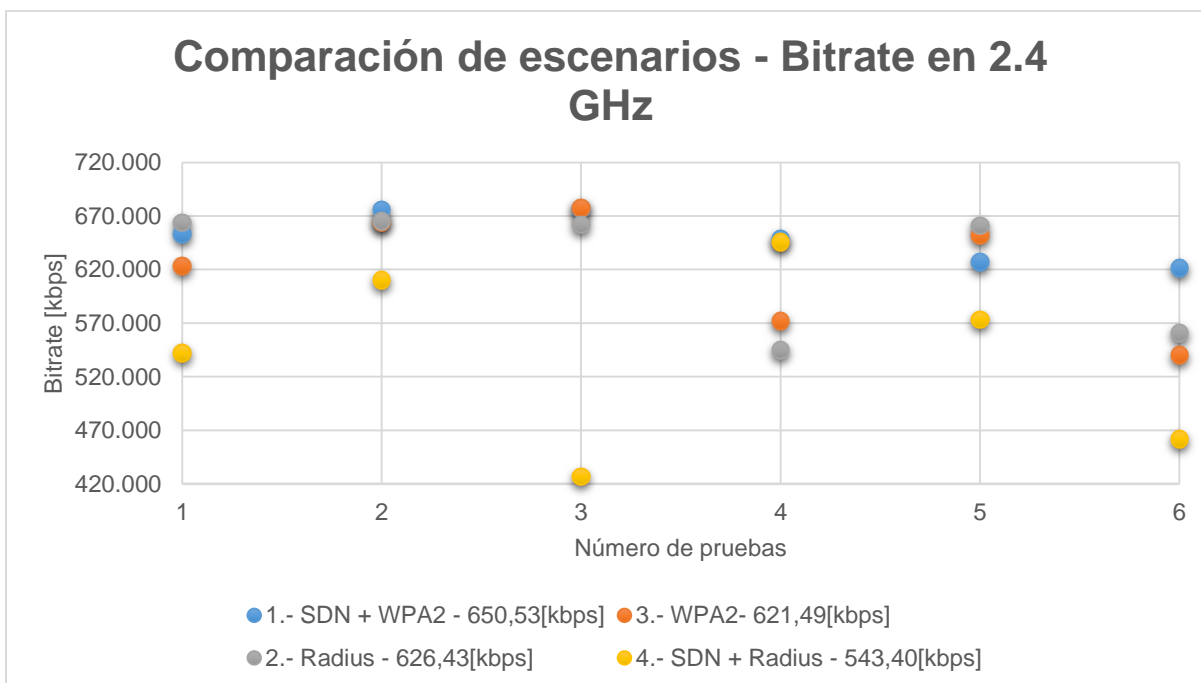


Figura 53 Resultados de comparación en bitrate de redes implementadas en 2.4 GHz.

En las pruebas de comparación de paquetes perdidos (pp) se obtiene que la red con el promedio más bajo es la red SDN con seguridad WPA2-PSK con 6,57%, seguido de la red tradicional con servidor Radius con 12,49%, la red tradicional con seguridad WPA2-PSK con 12,78% y la red SDN con servidor Radius con 24,08%. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 7

Comparación de redes implementadas en parámetro: pp en 2.4 GHz

Arquitectura de red	1	2	3	4	5	6	Promedio total
---------------------	---	---	---	---	---	---	----------------

1.- SDN + WPA2	8,35%	5,29%	5,19%	8,30%	4,89%	7,42%	6,57%
2.- Radius	7,49%	7,13%	7,28%	23,94%	7,54%	21,57%	12,49%
3.- WPA2	12,71%	7,22%	5,32%	20,00%	8,67%	22,75%	12,78%
4.- SDN + Radius	24,34%	14,85%	40,48%	9,87%	19,89%	35,06%	24,08%

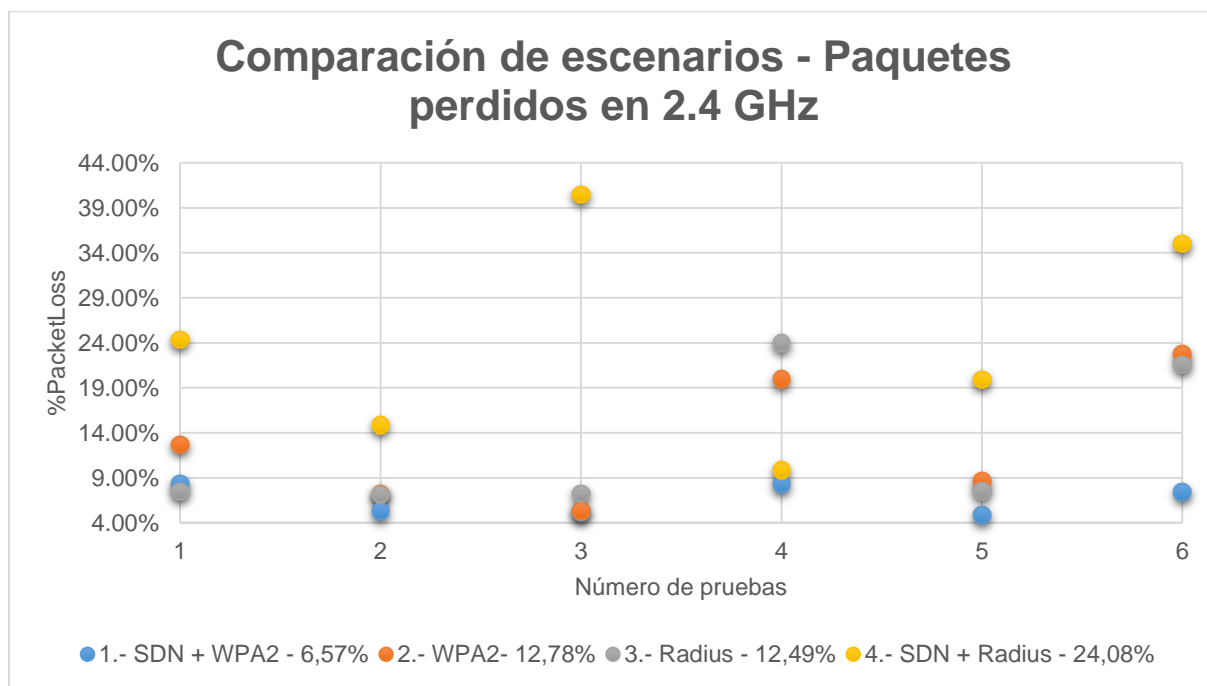


Figura 54 Resultados de comparación en pp de redes implementadas en 2.4 GHz.

En las pruebas de comparación de tiempo de autenticación (ta) se obtiene que la red con el promedio más bajo es la red tradicional con seguridad WPA2-PSK con 18,48 [ms], seguido de la red SDN con seguridad WPA2-PSK con 22,91 [ms], la red tradicional con servidor Radius con 92,43 [ms] y la red SDN con seguridad Radius con 166,68 [ms]. El detalle de las cinco pruebas efectivas se muestra a continuación:

Tabla 8

Comparación de redes implementadas en parámetro: t_a en 2.4 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	Promedio total [ms]
1.- WPA2	14,628	21,989	16,978	20,96	17,84	18,48
2.- SDN + WPA2	29,74	22,90	32,92	11,53	17,45	22,91
3.- Radius	42,13	67,43	92,43	167,75	92,43	92,43
4.- SDN + Radius	154,75	174,00	185,37	165,87	153,42	166,68

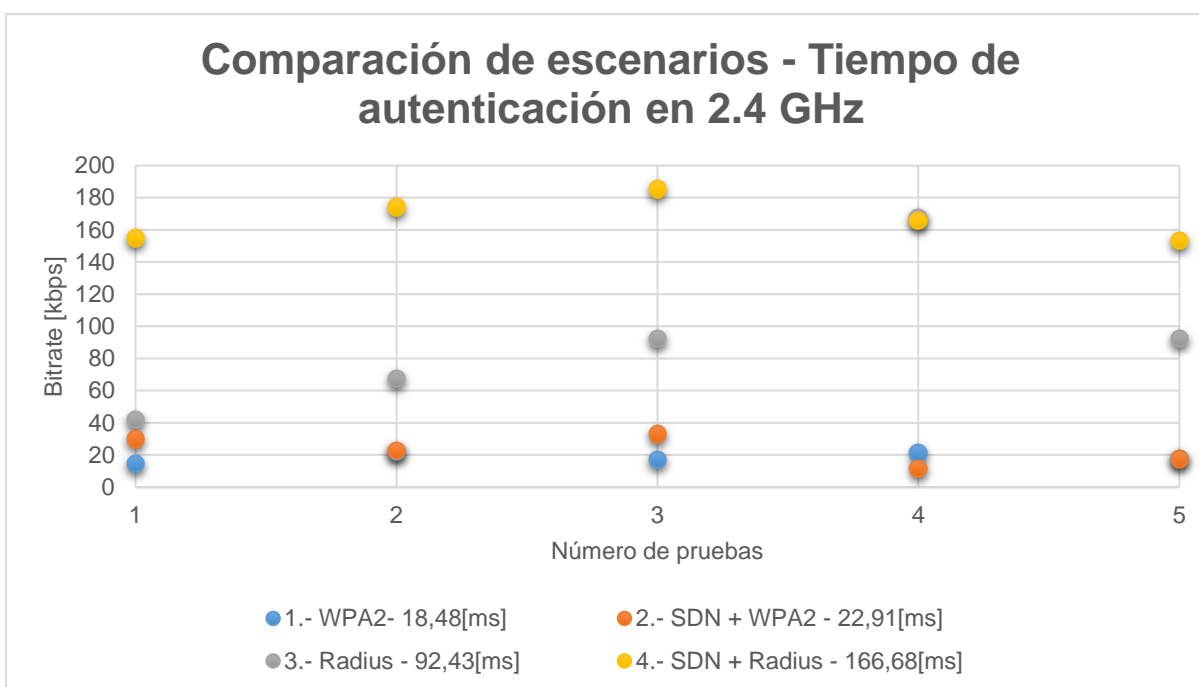


Figura 55 Resultados de comparación en t_a de redes implementadas en 2.4 GHz.

5.1.2. Comparación de arquitectura SDN implementada y red universitaria

En las pruebas de comparación de delay se obtiene que la red con el promedio más alto es la red Universitaria Eduroam con 566,64 [ms] y la red con mejor rendimiento

es la SDN con servidor Radius con 53,84 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 9

Comparación con red universitaria en parámetro: Delay en 2.4 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- SDN + Radius	28,104	39,747	127,325	31,736	43,016	53,109	53,840
2.- Eduroam	465,324	444,546	535,575	506,604	767,796	679,981	566,638

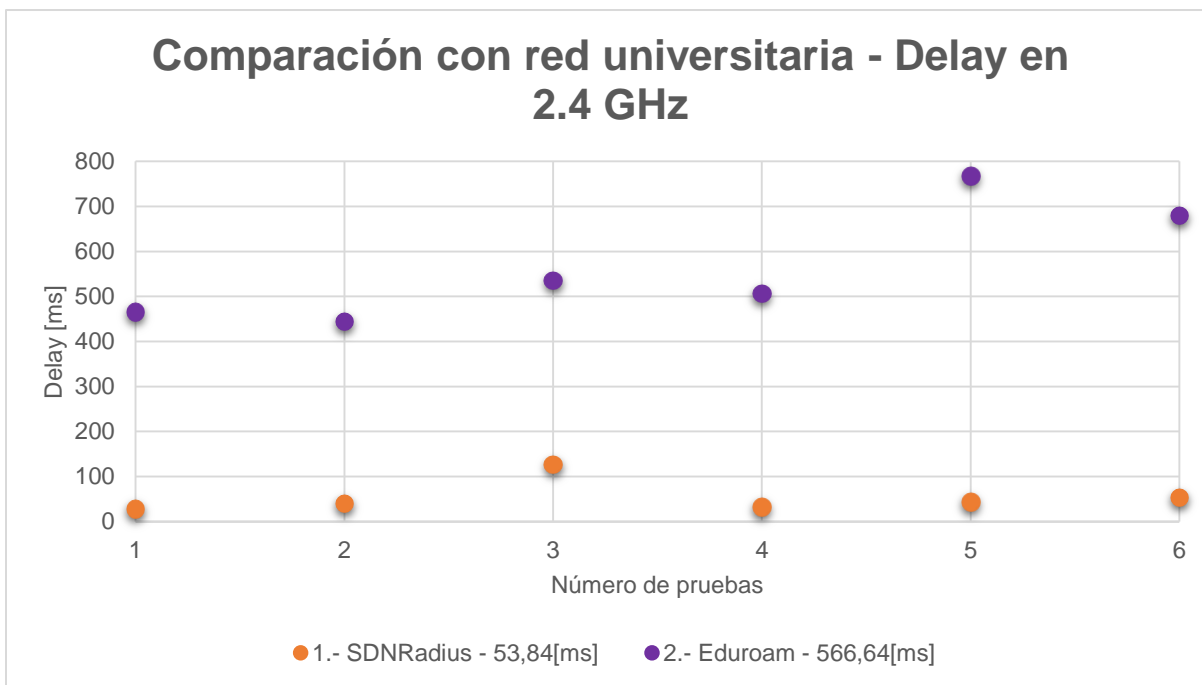


Figura 56 Resultados de comparación en delay con red universitaria en 2.4GHz.

En las pruebas de comparación de jitter se obtiene que la red con el promedio más alto es la red Universitaria Eduroam con 5,12 [ms] y la red con mejor rendimiento es la SDN con servidor Radius con 1,83 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 10

Comparación con red universitaria en parámetro: Jitter en 2.4 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- SDN + Radius	2,047	1,696	2,636	1,394	1,402	1,772	1,825
2.- Eduroam	4,936	3,941	5,12	4,735	6,206	5,775	5,119

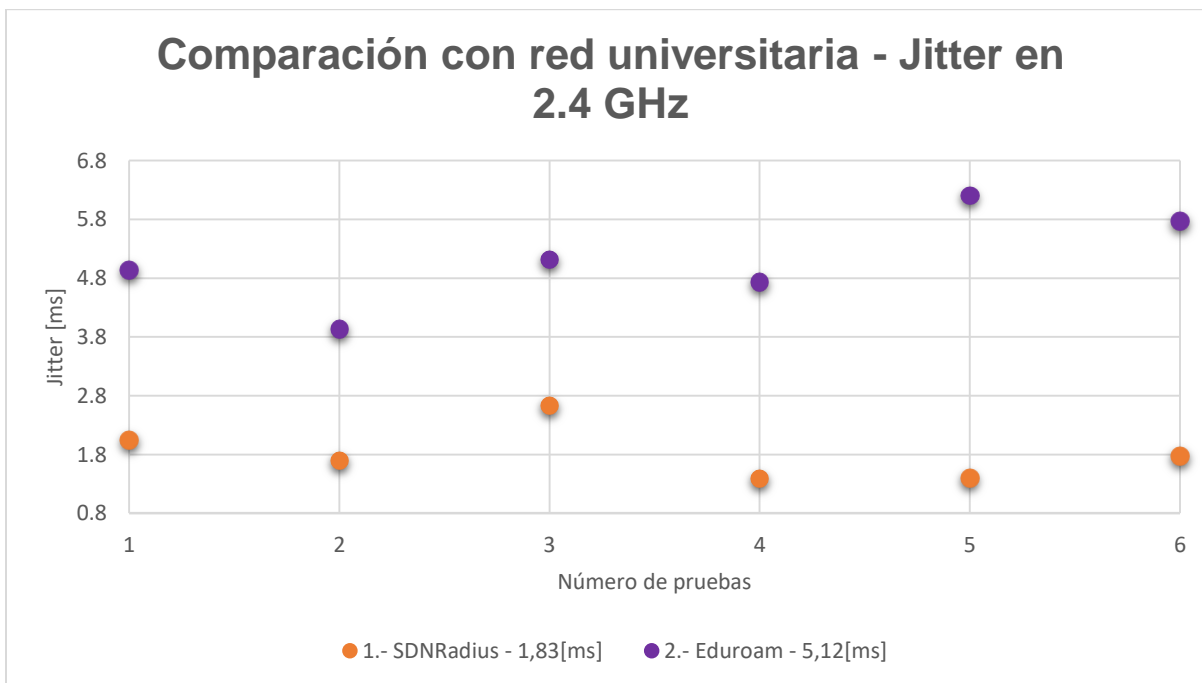


Figura 57 Resultados de comparación en jitter con red universitaria en 2.4GHz.

En las pruebas de comparación de bitrate se obtiene que la red con el promedio más bajo es la red Universitaria Eduroam con 346,10 [kbps] y la red con mejor rendimiento es la SDN con servidor Radius con 543,40 [kbps]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 11

Comparación con red universitaria en parámetro: Bitrate en 2.4 GHz

Arquitectura de red	1 [kbps]	2 [kbps]	3 [kbps]	4 [kbps]	5 [kbps]	6 [kbps]	Promedio total [kbps]
1.- SDN + Radius	542,098	610,304	427,023	645,863	573,257	461,830	543,396
2.- Eduroam	359,5149	436,8629	328,86	344,135	287,85203	319,3833	346,101

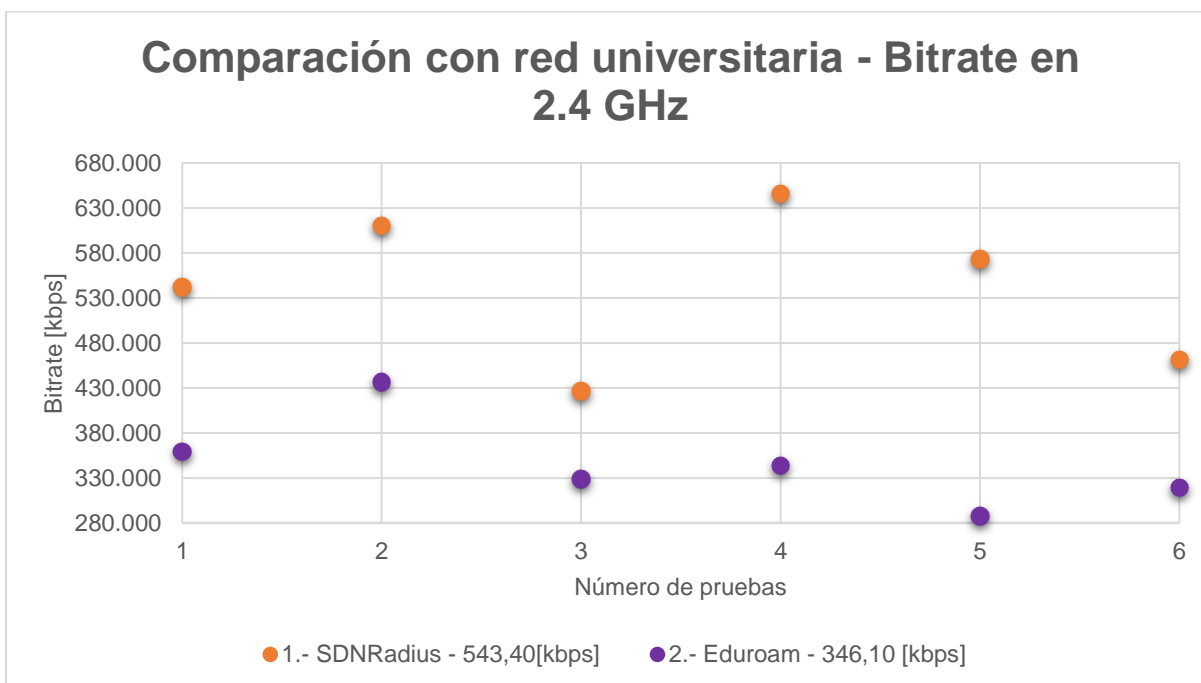


Figura 58 Resultados de comparación en bitrate con red universitaria en 2.4GHz.

En las pruebas de comparación de paquetes perdidos (pp) se obtiene que la red con el promedio más alto es la red Universitaria Eduroam con 34,38% y la red con mejor rendimiento es la SDN con servidor Radius con 24,08%. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 12

Comparación con red universitaria en parámetro: pp en 2.4 GHz

Arquitectura de red	1	2	3	4	5	6	Promedio total
1.- SDN + Radius	24,34%	14,85%	40,48%	9,87%	19,89%	35,06%	24,08%
2.- Eduroam	25,42%	36,73%	38,56%	25,34%	42,78%	37,45%	34,38%

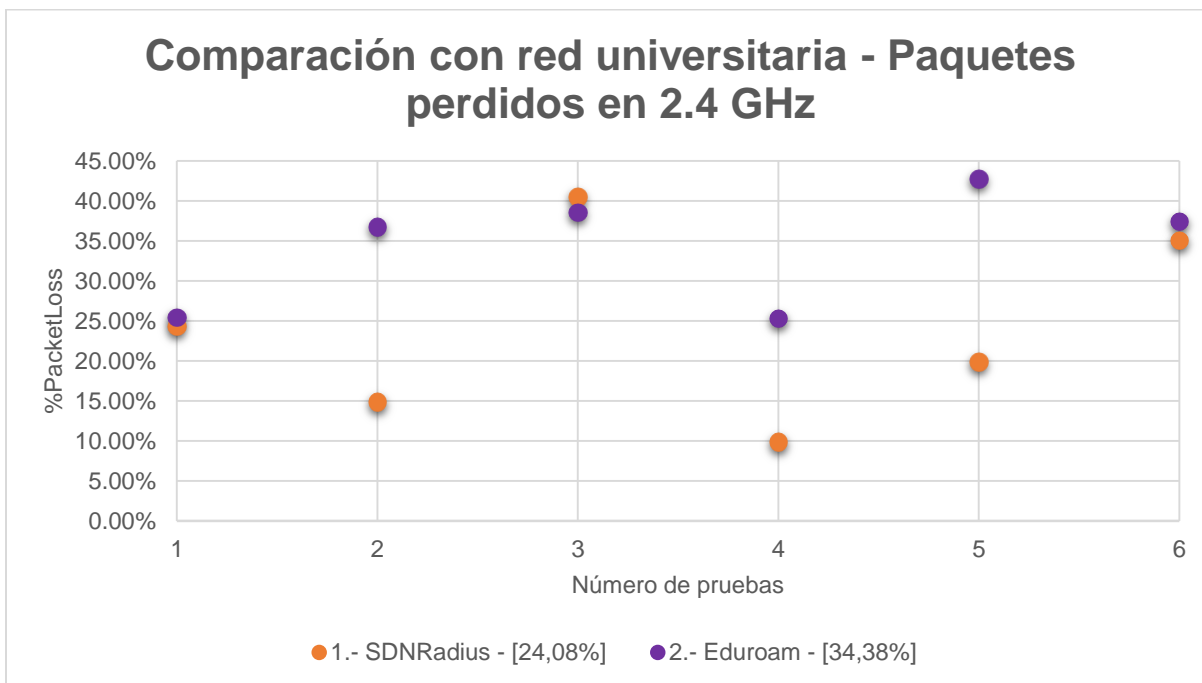


Figura 59 Resultados de comparación en pp con red universitaria en 2.4GHz.

En las pruebas de comparación de tiempo de autenticación (ta) se obtiene que la red con el promedio más alto es la red SDN con servidor Radius con 167,46 [ms] y la red con mejor rendimiento es la red universitaria Eduroam con 23,04 [ms]. El detalle de las cinco pruebas efectivas se muestra a continuación:

Tabla 13

Comparación con red universitaria en parámetro: ta en 2.4 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	Promedio total [ms]
1.- Eduroam	16,05	22,37	14,40	50,35	12,03	23,04

1.- SDNRadius	0,909	4,139	4,382	0,328	0,444	2,040
2.- Eduroam	2,970	0,810	7,464	12,434	5,164	5,768

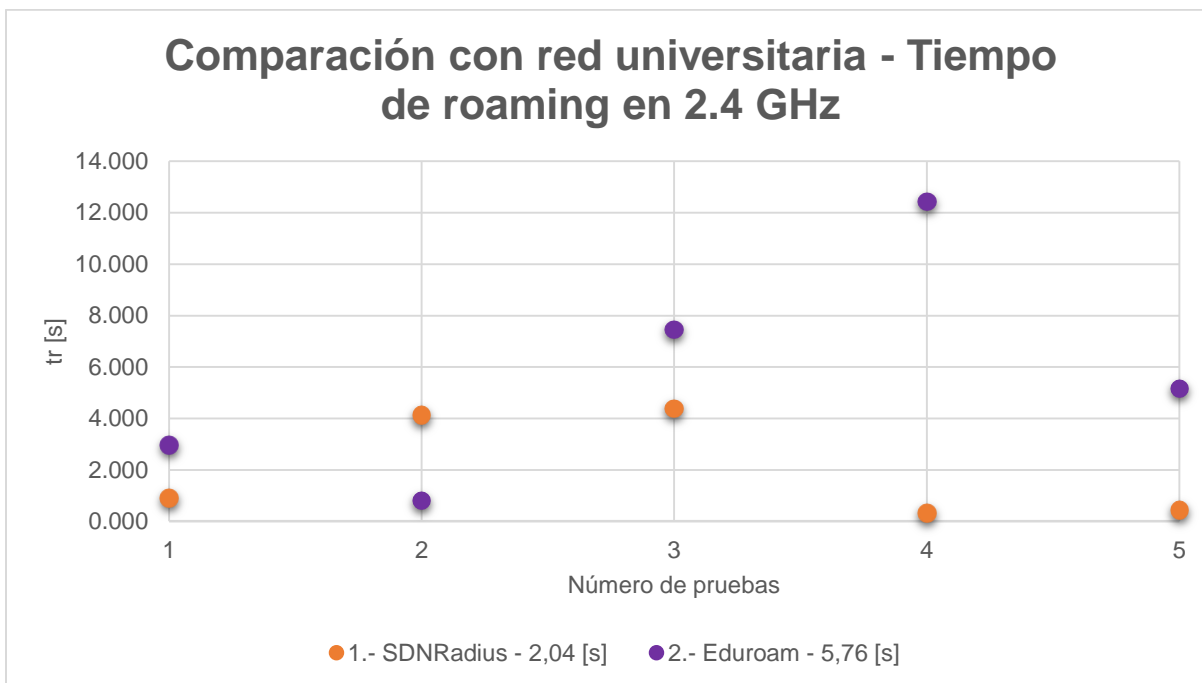


Figura 61 Resultados de comparación en tr con red universitaria en 2.4GHz.

5.2. Pruebas de red Wi-Fi 5 GHz

5.2.1. Comparación de arquitecturas de red implementadas en 5 GHz

En las pruebas de comparación de delay se obtiene que la red con el promedio más bajo es la red tradicional con servidor Radius con 56,18 [ms], seguido de la red SDN con servidor Radius con 64,55 [ms], red tradicional con seguridad WPA2-PSK con 90,48

[ms] y la red SDN con seguridad WPA2-PSK con 93,30 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 15

Comparación de redes implementadas en parámetro: Delay en 5 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- Radius	39,659	65,08	15,007	25,317	27,162	164,825	56,175
2.- SDN + Radius	26,519	22,826	30,362	61,987	43,704	201,914	64,552
3.- WPA2	44,4	36,794	41,881	172,397	101,369	146,031	90,479
4.- SDN + WPA2	52,142	50,084	52,07	162,255	119,534	123,723	93,301

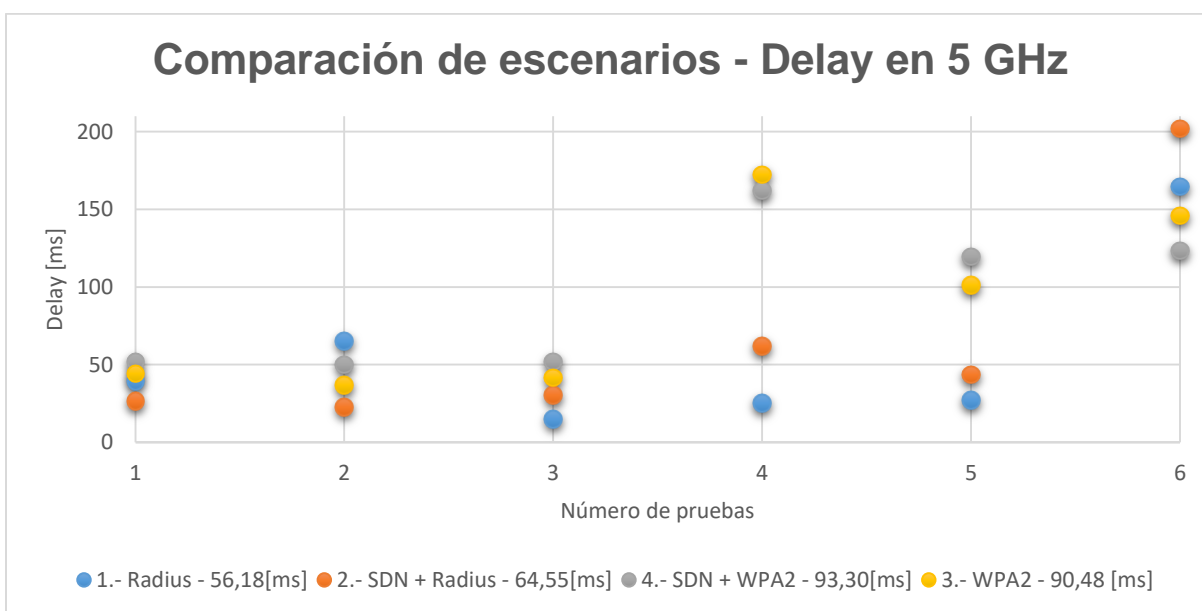


Figura 62 Resultados de comparación en delay de redes implementadas en 5 GHz.

En las pruebas de comparación de jitter se obtiene que la red con el promedio más bajo es la red tradicional con servidor Radius con 1,036 [ms], seguido de la red SDN con servidor Radius con 1,242 [ms], la red SDN con seguridad WPA2-PSK con 1,303 [ms] y

la red tradicional con seguridad WPA2-PSK con 1,425 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 16

Comparación de redes implementadas en parámetro: Jitter en 5 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- Radius	0,996	1,249	0,886	0,883	0,806	1,395	1,036
2.- SDN + Radius	1,122	0,931	1,099	1,39	1,159	1,751	1,242
3.- SDN + WPA2	1,13	1,053	1,078	1,543	1,766	1,246	1,303
4.- WPA2	1,245	1,074	1,107	1,976	1,58	1,565	1,425

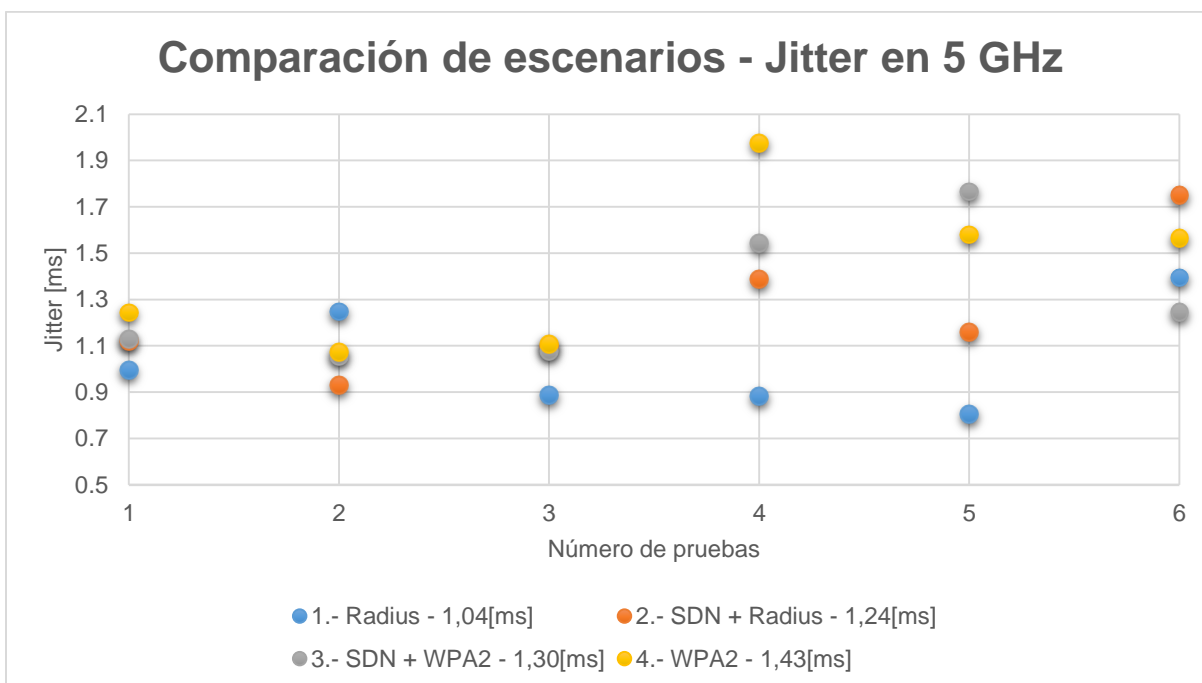


Figura 63 Resultados de comparación en jitter de redes implementadas en 5 GHz.

En las pruebas de comparación de bitrate se obtiene que la red con el promedio más alto es la SDN con servidor Radius con 673,92 [kbps], seguido de la red SDN con seguridad WPA2-PSK con 656,52 [kbps], la red tradicional con servidor Radius con

651,53 [kbps] y la red tradicional con seguridad WPA2-PSK con 630,95 [kbps]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 17

Comparación de redes implementadas en parámetro: Bitrate en 5 GHz

Arquitectura de red	1 [kbps]	2 [kbps]	3 [kbps]	4 [kbps]	5 [kbps]	6 [kbps]	Promedio total [kbps]
1.- SDN + Radius	711,64	709,56	706,37	623,92	708,19	583,85	673,92
2.- SDN + WPA2	709,37	711,03	708,50	591,04	623,67	595,48	656,52
3.- Radius	704,61	580,96	564,51	692,90	707,37	658,84	651,53
4.- WPA2	598,63	708,88	710,30	587,22	571,96	608,73	630,95

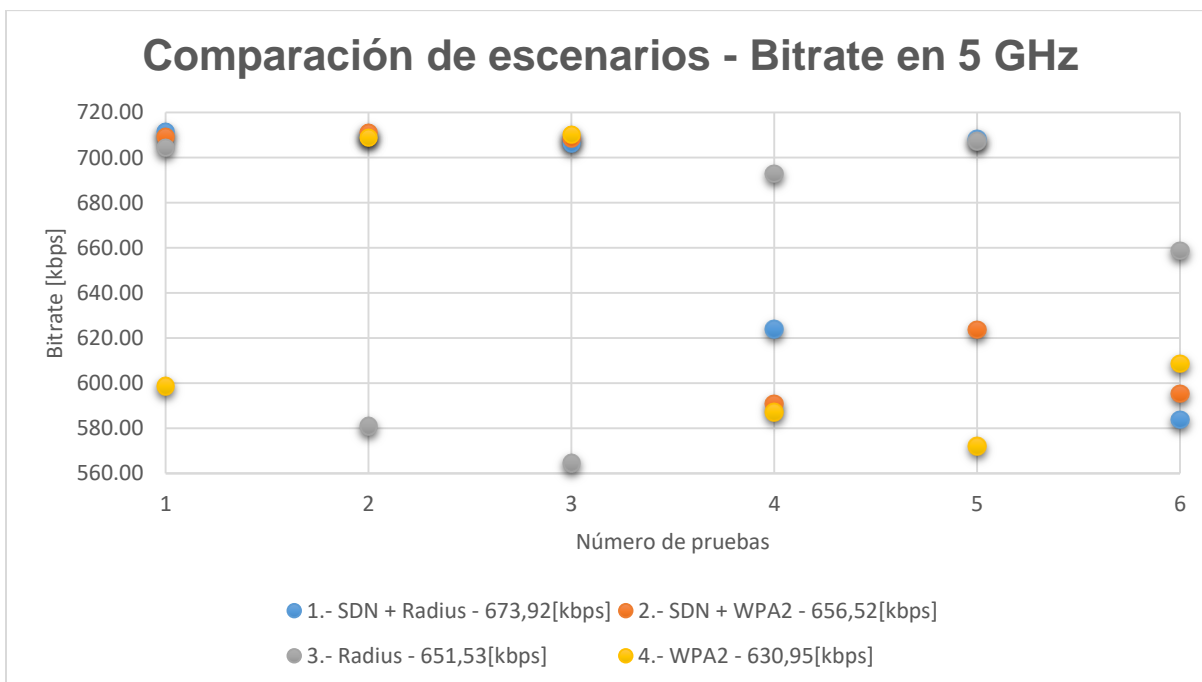


Figura 64 Resultados de comparación en bitrate de redes implementadas en 5 GHz.

En las pruebas de comparación de paquetes perdidos (pp) se obtiene que la red con el promedio más bajo es la SDN con servidor Radius con 5,74%, seguido de la red

SDN con seguridad WPA2-PSK con 7,87%, la red tradicional con servidor Radius con 8,82% y la red tradicional con seguridad WPA2-PSK con 11,50%. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 18

Comparación de redes implementadas en parámetro: pp en 5 GHz

Arquitectura de red	1	2	3	4	5	6	Promedio total
1.- SDN + Radius	1,57%	0,49%	0,89%	12,38%	0,79%	18,30%	5,74%
2.- SDN + WPA2	0,40%	0,29%	0,55%	17,11%	12,41%	16,48%	7,87%
3.- Radius	1,38%	18,88%	21,18%	3,01%	0,81%	7,67%	8,82%
4.- WPA2	16,42%	0,52%	0,37%	17,44%	19,70%	14,52%	11,50%

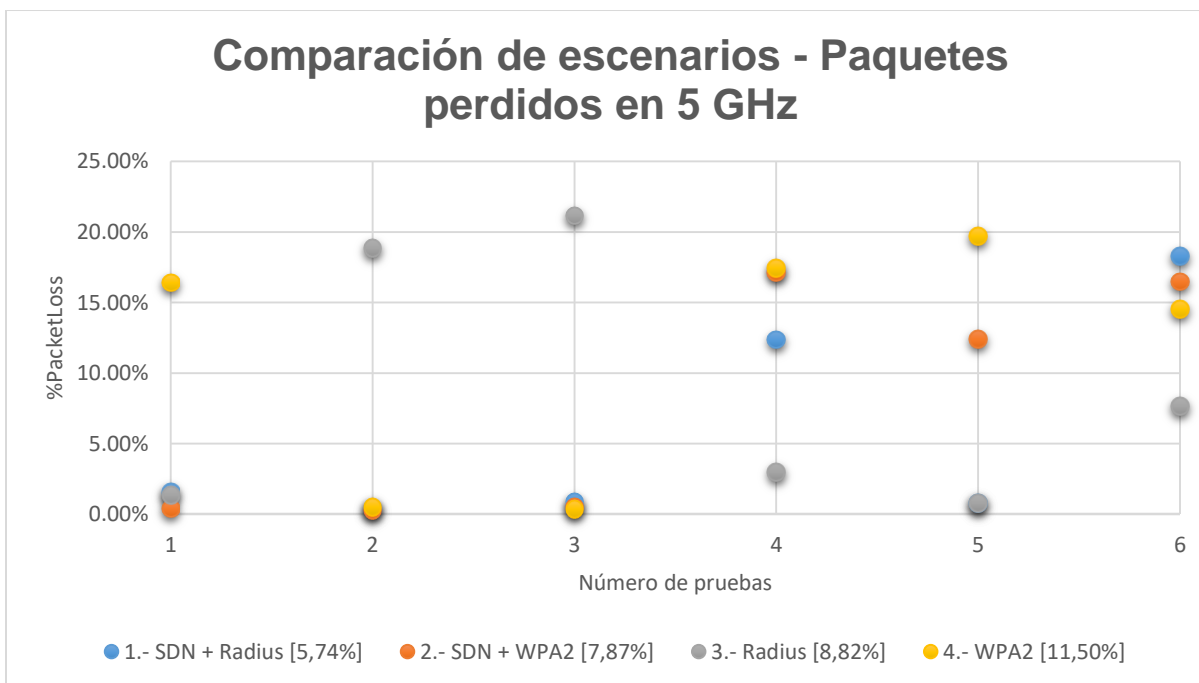


Figura 65 Resultados de comparación en pp de redes implementadas en 5 GHz.

En las pruebas de comparación de tiempo de autenticación (ta) se obtiene que la red con el promedio más bajo es la red tradicional con seguridad WPA2-PSK con 10,21 [ms], seguida de la red SDN con seguridad WPA2-PSK con 48,63 [ms], la red tradicional con servidor Radius con 95,31 [ms] y la red SDN con seguridad Radius con 100,03 [ms]. El detalle de las cinco pruebas efectivas se muestra a continuación:

Tabla 19

Comparación de redes implementadas en parámetro: ta en 5 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	Promedio total [ms]
1.- WPA	10,375	9,374	11,56	10,80	8,95	10,21
2.- SDNWPA	29,37	51,01	74,89	26,24	61,67	48,63
3.- Radius	117,13	85,98	89,83	83,06	100,54	95,31
4.- SDNRadius	82,54	102,36	103,99	114,02	97,23	100,03

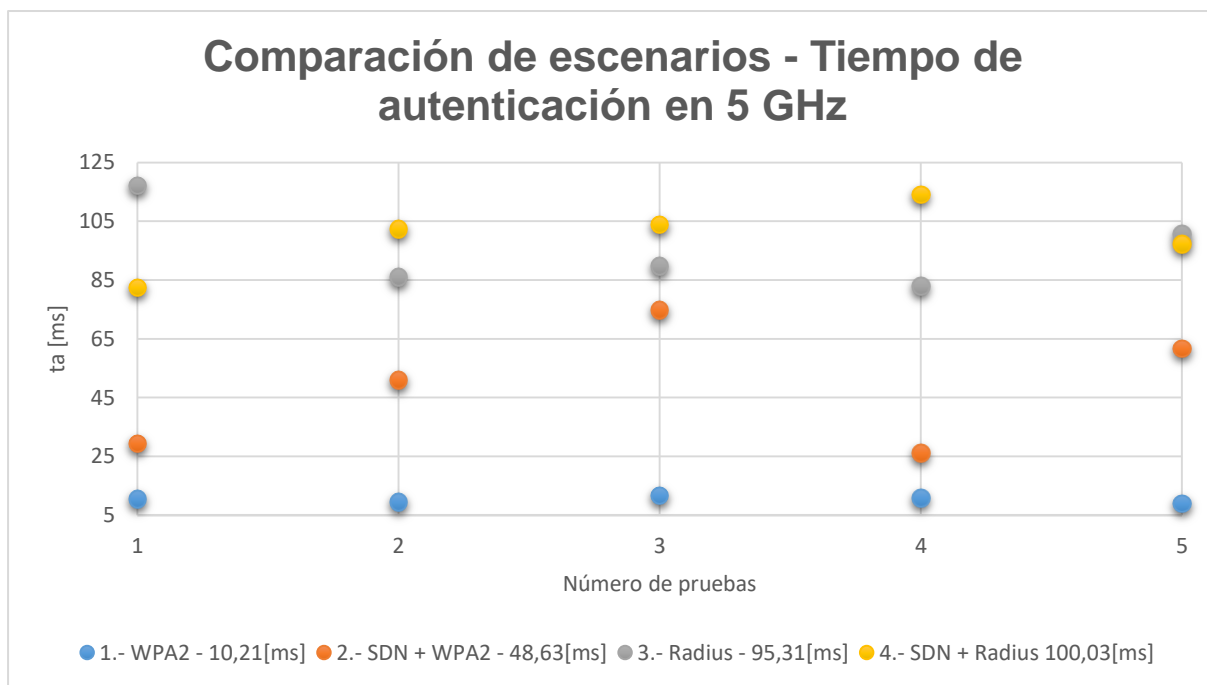


Figura 66 Resultados de comparación en ta de redes implementadas en 5 GHz.

5.2.2. Comparación de arquitectura SDN implementada y red universitaria

En las pruebas de comparación de delay se obtiene que la red con el promedio más alto es la red Universitaria Eduroam con 214,55 [ms] y la red con mejor rendimiento es la SDN con servidor Radius con 64,55 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 20

Comparación con red universitaria en parámetro: Delay en 5 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- SDN + Radius	26,519	22,826	30,362	61,987	43,704	201,914	64,552
2.- Eduroam	208,078	267,167	238,41	271,92	117,458	184,268	214,550

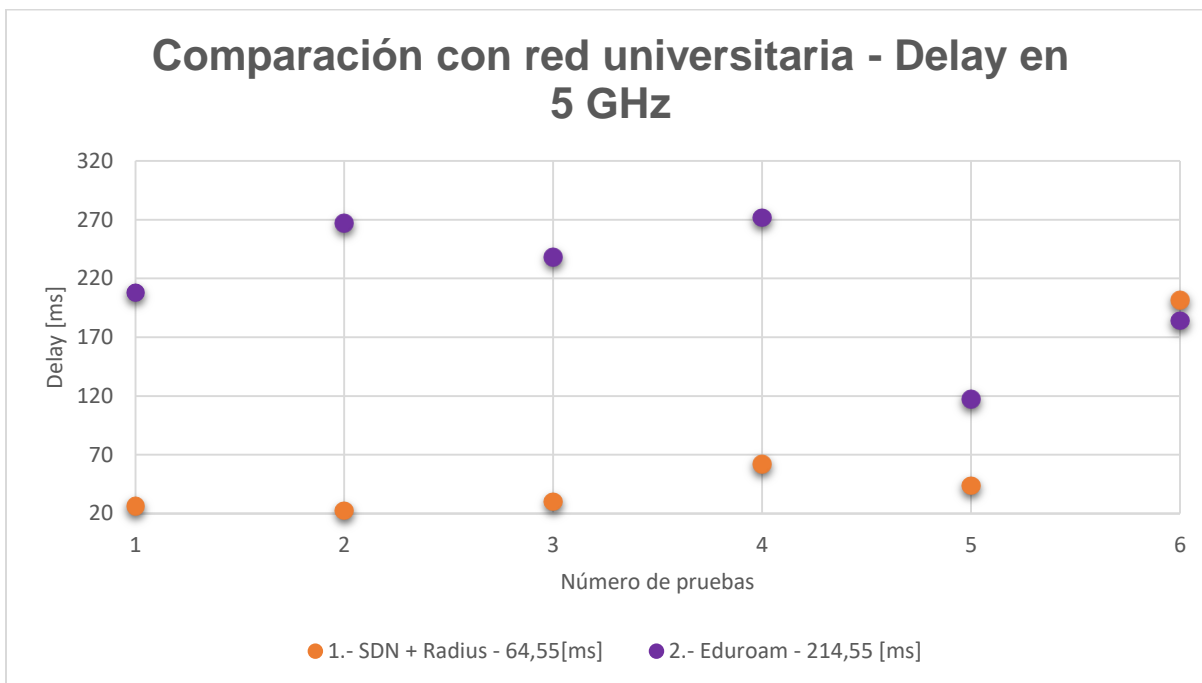


Figura 67 Resultados de comparación en delay con red universitaria en 2.4GHz.

En las pruebas de comparación de jitter se obtiene que la red con el promedio más alto es la red Universitaria Eduroam con 2,1068 [ms] y la red con mejor rendimiento es la SDN con servidor Radius con 1,176 [ms]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 21

Comparación con red universitaria en parámetro: Jitter en 5 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	6 [ms]	Promedio total [ms]
1.- SDN + Radius	1,122	0,931	1,099	1,39	1,159	1,751	1,242

2.- Eduroam 2,479 3,527 2,872 2,789 2,097 2,077 2,640

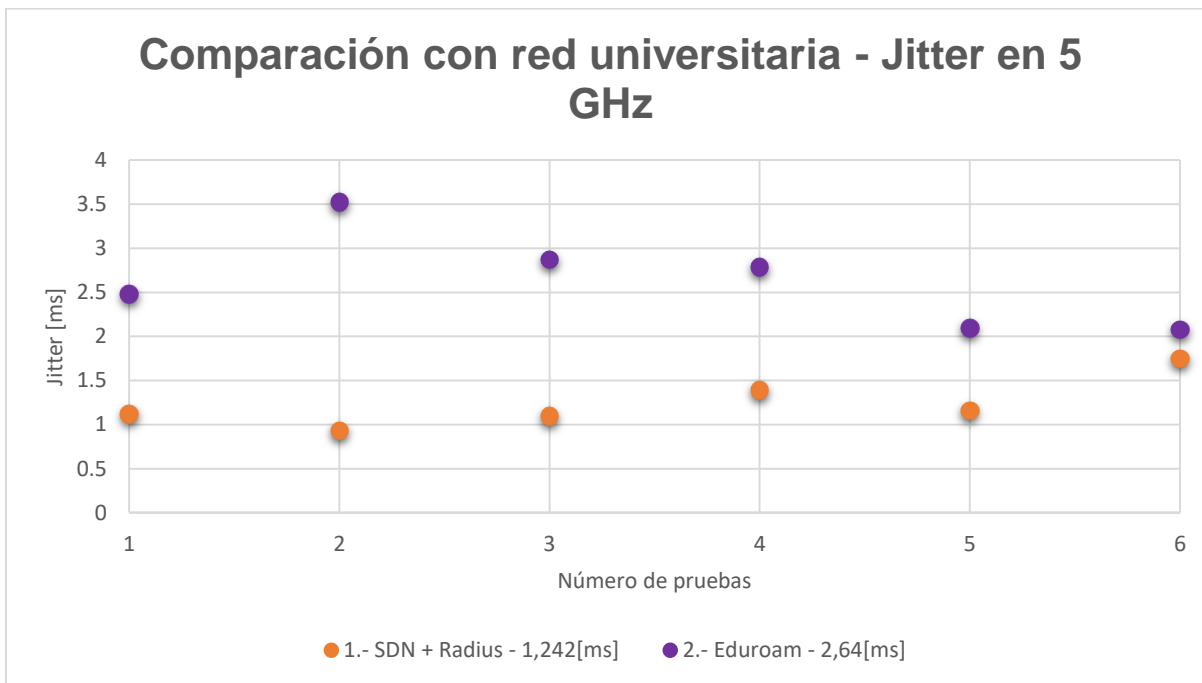


Figura 68 Resultados de comparación en jitter con red universitaria en 5 GHz.

En las pruebas de comparación de bitrate se obtiene que la red con el promedio más bajo es la red Universitaria Eduroam con 563,39 [kbps] y la red con mejor rendimiento es la SDN con servidor Radius con 683,59 [kbps]. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 22

Comparación con red universitaria en parámetro: Bitrate en 5 GHz

Arquitectura de red	1 [kbps]	2 [kbps]	3 [kbps]	4 [kbps]	5 [kbps]	6 [kbps]	Promedio total [kbps]
1.- SDN + Radius	711,64	709,56	706,37	623,92	708,19	583,85	673,921
2.- Eduroam	525,0377	425,6827	486,6304	439,7172	654,1204	583,5858	519,129

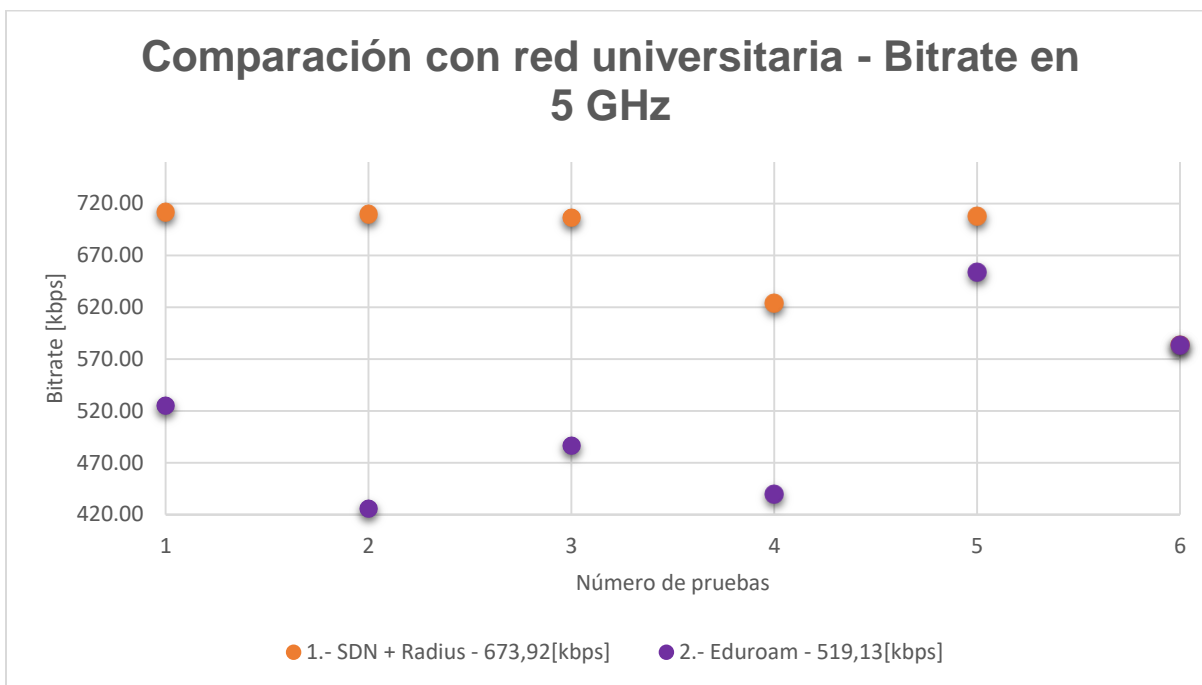


Figura 69 Resultados de comparación en bitrate con red universitaria en 5 GHz.

En las pruebas de comparación de paquetes perdidos (pp) se obtiene que la red con el promedio más alto es la red Universitaria Eduroam con 24,21% y la red con mejor

rendimiento es la SDN con servidor Radius con 5,74%. El detalle de las seis pruebas en los dos escenarios distintos se muestra a continuación:

Tabla 23

Comparación con red universitaria en parámetro: pp en 5 GHz

Arquitectura de red	1	2	3	4	5	6	Promedio total
1.- SDN + Radius	1,57%	0,49%	0,89%	12,38%	0,79%	18,30%	5,74%
2.- Eduroam	23,57%	37,60%	29,33%	36,35%	3,06%	15,35%	24,21%

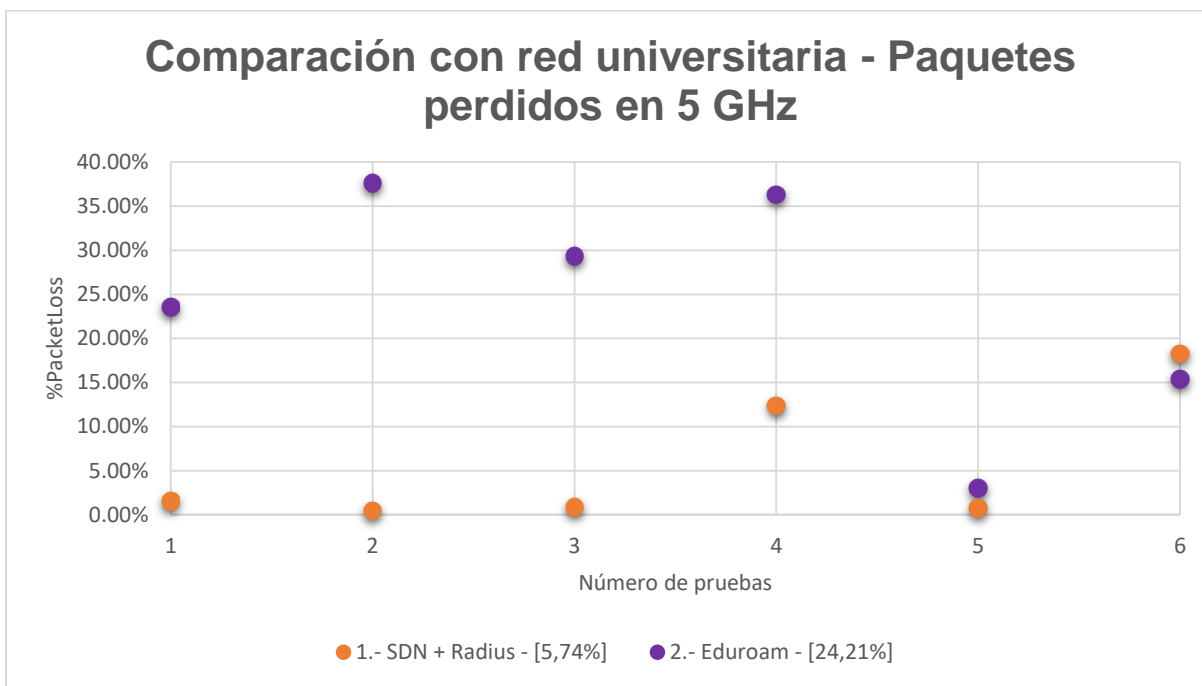


Figura 70 Resultados de comparación en pp con red universitaria en 5 GHz.

En las pruebas de comparación de tiempo de autenticación (t_a) se obtiene que la red con el promedio más alto es la red SDN con servidor Radius con 100,03 [ms] y la red con mejor rendimiento es la red universitaria Eduroam con 10,62 [ms]. El detalle de las cinco pruebas efectivas se muestra a continuación:

Tabla 24

Comparación con red universitaria en parámetro: t_a en 5 GHz

Arquitectura de red	1 [ms]	2 [ms]	3 [ms]	4 [ms]	5 [ms]	Promedio total [ms]
1.- Eduroam	8,77	18,81	8,75	8,30	8,48	10,62
2.- SDNRadius	82,54	102,36	103,99	114,02	97,23	100,03

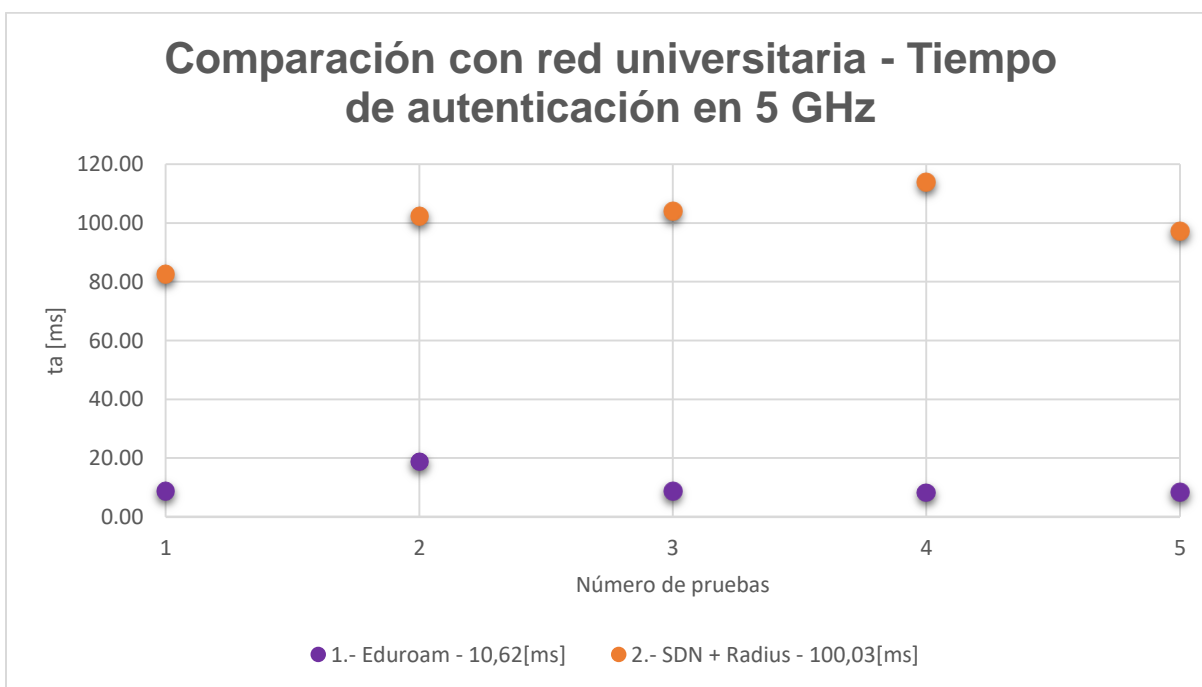


Figura 71 Resultados de comparación en t_a con red universitaria en 5 GHz.

En las pruebas de comparación de tiempo de roaming (t_r) se obtiene que la red con el promedio más alto es la red universitaria Eduroam con 4,12 [s] y la red con mejor rendimiento es la SDN con servidor Radius con 2,07 [s]. El detalle de las cinco pruebas efectivas se muestra a continuación:

Tabla 25

Comparación con red universitaria en parámetro: t_r en 5 GHz

Arquitectura de red	1 [s]	2 [s]	3 [s]	4 [s]	5 [s]	Promedio total [s]
1.- SDNRadius	4,001	0,575	5,305	0,268	0,204	2,071
2.- Eduroam	6,546	4,556	3,283	0,679	5,555	4,124

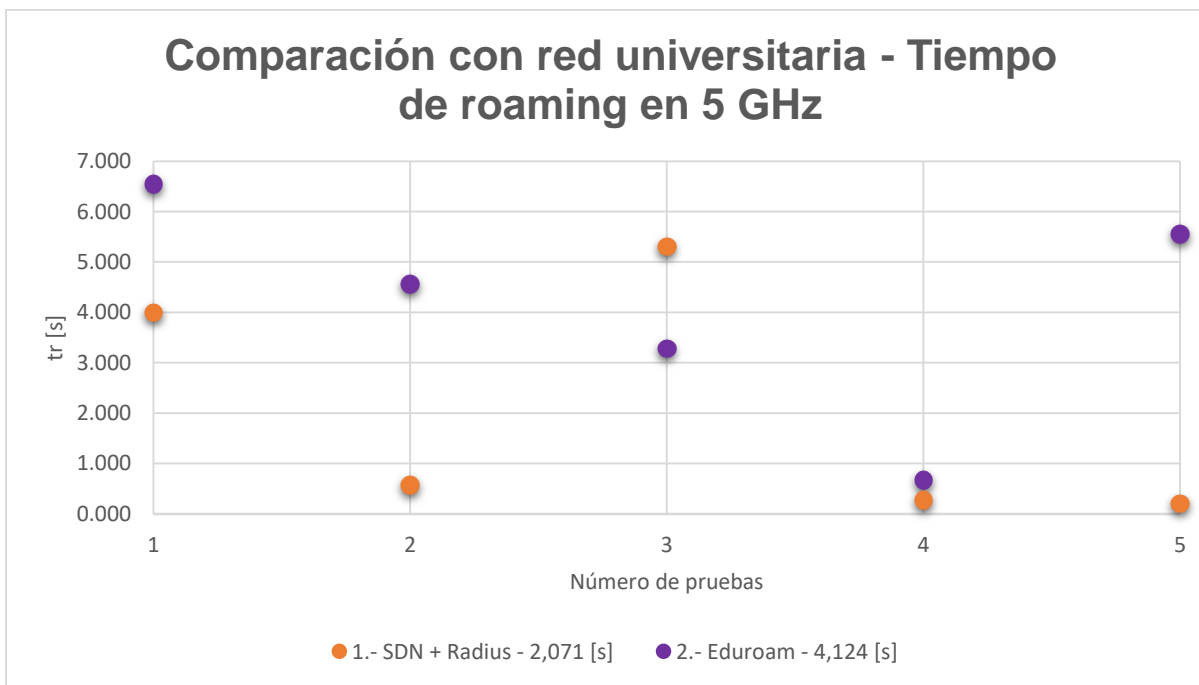


Figura 72 Resultados de comparación en t_r con red universitaria en 5 GHz.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

En la implementación de las redes SDN se pudo obtener una red programable y administrada desde un solo punto y de manera sencilla desde el Controlador SDN, permitiendo equilibrar la carga y distribuir el tráfico de manera eficiente para evitar la generación de cuellos de botella o puntos de bloqueo con ayuda de las tablas de flujo, mejorando el rendimiento en términos de calidad de servicio. Otro de los factores importantes de las SDN es la automatización permitiendo un entorno consistente, flexible y escalable ante cualquier infraestructura de red. Al contar con una gestión unificada a través de protocolos de código abierto como OpenFlow se pudo trabajar con elementos intermedios de red como Zodiac y Cisco sin ser compatibles, logrando en un ambiente organizacional la reducción de costos.

Para la implementación de una infraestructura de red inalámbrica comparable, se tomó en consideración los siguientes parámetros físicos en los puntos de acceso: su ubicación, su potencia y arreglo de antenas, para emular el rango de servicio Wi-fi en los clientes inalámbricos, además se consideró que los dispositivos soporten comunicaciones *dual band* para realizar pruebas en las bandas de 2.4 GHz como en 5 GHz. En los parámetros lógicos se consideró los siguientes aspectos: las tasas teóricas de transmisión, protocolos de encriptación (WEP, WPA, WPA2), estándares 802.11 usados (ac/b/g/n) y la posibilidad de implementar múltiples SSID.

Los resultados obtenidos y presentados en el capítulo 5 son satisfactorios, observándose en promedio un tiempo de roaming para el Testbed con SDN + Radius de tan solo el 50.21% del tiempo promedio logrado por la red Universitaria en 5 GHz, y el 35.36% en el caso de 2.4 GHz, lo cual representa una mejora de 2.053 y 3.728 segundos en el tiempo de recuperación del servicio, respectivamente. Estas mejoras en el tiempo de roaming ocurren a pesar de que el servidor Radius de la Universidad es considerablemente más rápido que el del Testbed, siendo el tiempo de autenticación promedio del Testbed 100.03 ms en 5 GHz y 167.46 ms en 2.4 GHz, bastante más lento que los 10.62 y 23.04 ms, respectivamente, de la red Eduroam. Cabe recalcar que estos tiempos de autenticación no representan un valor significativo frente a los tiempos de roaming, los cuales se encuentran en la magnitud de los segundos.

Las métricas de calidad de servicio también indican resultados satisfactorios. Al realizar varias pruebas de inyección de tráfico en escenarios de roaming, el Testbed mostró un delay promedio del 30.08% con respecto a la red Eduroam en 5 GHz y apenas el 9.5% en 2.4 GHz. El jitter presentó magnitudes promedio relativas del 47.04% en 5 GHz y 35.65% en 2.4 GHz. El bitrate no es un resultado significativo debido a que utilizó una tasa muy baja para la inyección, siendo la intención no causar cuellos de botella a medida que la potencia de recepción, y en consecuencia la tasa de transmisión, bajen. Por otro lado, la tasa de paquetes perdidos si es relevante, y esta muestra un 23.7% de paquetes perdidos en el Testbed respecto a los perdidos en la red Eduroam en 5 GHz y un 70.04% de los paquetes perdidos en 2.4 GHz.

Como se pudo observar, existen considerables principalmente en el tiempo de roaming, delay y jitter, parámetros importantes al momento de evaluar el desempeño de una red inalámbrica para aplicaciones en tiempo real, por lo que se proponen ciertas aplicaciones que se podrían beneficiar de esta nueva arquitectura de red. Puntualmente, y tomando en cuenta el ámbito universitario de la red estudiada, se propone que aplicaciones como la telefonía IP, videoconferencia y streaming de video se verían beneficiadas por esta arquitectura, cortando significativamente el tiempo de pérdida de servicio cuando ocurre el roaming entre Access Points.

6.2. Recomendaciones

Se recomienda realizar pruebas comparativas con los diferentes estándares de encriptación en redes inalámbricas, ya que difieren en la utilización de la suite de cifrado, claves de encriptación, vectores de inicialización y algoritmos de autenticación, lo que provoca un comportamiento diferente en términos de parámetros de calidad de servicio y roaming.

El controlador por el que se decidió para la implementación de este proyecto es ryu. Otro controlador que fue considerado es OpenDaylight, sin embargo, este presenta la desventaja de recibir actualizaciones significativas constantemente, las cuales por lo general son incompatibles con aplicaciones desarrolladas para versiones previas y requieren un trabajo extensivo para poder adaptarlas a nuevas versiones. Por otro lado, ryu actúa como un framework muy ligero para aplicaciones de cualquier nivel de

complejidad con mucha mayor compatibilidad, por lo que finalmente se decidió y se recomienda el uso de este sobre otros controladores.

En cuanto al despliegue de la red, se utilizó la herramienta Acrylic Wi-Fi Heatmaps para la correcta configuración de la cobertura inalámbrica. Herramientas como esta son valiosas para el dimensionamiento de cobertura, y dependiendo del tipo de solución que se utilice esta puede incluir herramientas similares, por lo que el uso de ellas para el despliegue y operación de una red inalámbrica es altamente recomendado.

Se recomienda la utilización del emulador de redes SDN Mininet-wifi previo a las pruebas de rendimiento en escenarios reales, ya que el software permite virtualizar infraestructuras de red con host, switches, routers y enlaces de comunicación a través de kernel de Linux, emulando cualquier parámetro de comunicación inalámbrica como velocidad de transmisión, retardo, modelo de pérdida de potencia y movilidad por parte de los terminales de usuario a través de códigos de programación. Adicionalmente se pueden implementar servicios externos como Radius y configuración de protocolos de encriptación Wi-fi para una emulación completa de la infraestructura de red.

Se recomienda el uso de la plataforma de generación de tráfico D-ITG (Distributed Internet Traffic Generator), ya que permite la construcción de paquetes TDP y UDP en IPv4 e IPv6 en las capas de red, transporte y aplicación. El software cuenta con la ventaja que al iniciar la inyección de tráfico este no se detiene ante la realización de roaming por parte de uno de los clientes, permitiendo obtener un resultado de parámetros de calidad de servicio en los cambios de conectividad entre los puntos de acceso, a diferencia de servicios como streaming, ftp, etc.

6.3. Futuros estudios

Durante el desarrollo de este proyecto se identificaron varias limitantes que restringieron el alcance de este, las cuales de ser solucionadas podrían abrir el paso para nuevas investigaciones en el ámbito de las Redes Definidas por Software aplicadas a redes Wi-Fi.

Específicamente, el protocolo Openflow fue utilizado debido a ser Open Source, sin embargo, desde su inepción varios fabricantes han desarrollado sus propios protocolos de SDN, mucho más poderosos y con diferentes conjuntos de características que pueden beneficiar a las redes inalámbricas.

Una de las mayores limitantes de Openflow en las redes Wi-Fi es que este protocolo no trabaja directamente con paquetes Wi-Fi, es decir, no tiene acceso a la información contenida en las cabeceras 802.11 que podría permitir un mayor control sobre la red basado en dicha información. En un futuro podría surgir la forma de usar este tipo de información para la implementación de nuevas redes que responden dinámicamente a la información proporcionada por este tipo de paquetería.

REFERENCIAS

- Open Networking Foundation. (2016). *SDN Architecture* . Obtenido de https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf
- Arteche, C. (02 de 2014). *Despliegue de una maqueta de red basada en OpenFlow*. Recuperado el 07 de 12 de 2017, de <https://repositorio.unican.es/xmlui/bitstream/handle/10902/4484/Carlos%20Arteche%20Gonzalez.pdf?sequence=1>
- Atlassian Confluence Open Source Project. (04 de 02 de 2013). *What is Beacon?* Recuperado el 06 de 01 de 2018, de <https://openflow.stanford.edu/display/Beacon/Home>
- Banerji, S., & Chowdhury, R. S. (2003). On IEEE 802.11: Wireless Lan Technology. *International Journal of Mobile Network Communications & Telematics (IJMNCT)*.
- Big Switch Networks. (s.f.). *Project Floodlight*. Recuperado el 06 de 01 de 2018, de <http://www.projectfloodlight.org/>
- CEDIA. (s.f.). *EDUROAM*. Recuperado el 13 de 01 de 2019, de <https://www.cedia.edu.ec/es/servicios/tecnologia/conectividad/eduroam>
- CISCO. (s.f.). *¿Cómo el RADIUS trabaja?* Recuperado el 12 de 01 de 2019, de https://www.cisco.com/c/es_mx/support/docs/security-vpn/remote-authentication-dial-user-service-radius/12433-32.pdf
- CISCO. (12 de 07 de 2016). *Cisco Wireless LAN Controller Configuration Guide, Release 7.4*. Obtenido de https://www.cisco.com/c/en/us/td/docs/wireless/controller/7-4/configuration/guides/consolidated/b_cg74_CONSOLIDATED.pdf
- CISCO;. (12 de 07 de 2016). *Cisco Wireless LAN Controller Configuration Guide, Release 7.4*. Recuperado el 07 de 12 de 2017, de https://www.cisco.com/c/en/us/td/docs/wireless/controller/7-4/configuration/guides/consolidated/b_cg74_CONSOLIDATED.pdf
- Cuenca, G., & Flores, M. (3 de 2015). *Redes definidas por software: Solución para servicios portadores del Ecuador*. Recuperado el 3 de 11 de 2017, de <http://revistas.uees.edu.ec/index.php/IRR/article/view/23/21>
- EDUROAM. (s.f.). *How does eduroam work?* Recuperado el 13 de 01 de 2019, de <https://www.eduroam.org/how/>

- Fontes, R., Afzal, S., Brito, S., Santos, M., & Rothenberg, C. (2015). *Mininet-WiFi: Emulating Software-Defined Wireless Networks*. IFIP.
- Funk, P., & Blake-Wilson, S. (Agosto de 2008). *Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)*. Obtenido de IETF Tools: <https://tools.ietf.org/html/rfc5281>
- García, A., Rodríguez, C., Calderón, C., & Casmartiño, F. (2014). Controladores SDN, elementos para su selección y evaluación. *Telem@tica*, 10-20.
- García, A., Rodríguez, C., Calderón, C., & Casmartiño, F. (2014). Controladores SDN, elementos para su selección y evaluación. *Revista Telem@tica*, XIII(3), 10-20.
- Github. (s.f.). *Mininet-Wifi Emulator for Software-Defined Wireless Networks*. Obtenido de <https://github.com/intrig-unicamp/mininet-wifi>
- Gómez, C., & Fernández, C. (23 de 9 de 2013). *Las TIC en la educación*. . Recuperado el 1 de 11 de 2017, de <http://files.cintyha-gomez6.webnode.mx/200000038-093670a316/actividad%209.pdf>
- Guano, J. (05 de 2017). *Prototipo de una SDN utilizando herramientas Open-source*. Obtenido de <http://bibdigital.epn.edu.ec/bitstream/15000/17327/1/CD-7823.pdf>
- Guano, J., & Tipantuña, C. (5 de 2017). *Prototipo de una SDN utilizando herramientas Open-Source*. Recuperado el 6 de 11 de 2017, de <http://bibdigital.epn.edu.ec/bitstream/15000/17327/1/CD-7823.pdf>
- INFORMATION & NETWORKING TECHNOLOGIES RESEARCH & INNOVATION GROUP (INTRIG). (11 de 2018). *The user Manual - Mininet-WiFi*. Recuperado el 09 de 01 de 2019, de <https://github.com/intrig-unicamp/mininet-wifi>
- inside-openflow. (10 de Agosto de 2016). *SimpleSwitch 2.0*. Obtenido de GitHub: <https://github.com/inside-openflow/simpleswitch2>
- INTEL. (08 de 03 de 2018). *Resumen de 802.1 X y tipos de EAP*. Recuperado el 13 de 01 de 2019, de <https://www.intel.la/content/www/xl/es/support/articles/000006999/network-and-i-o/wireless-networking.html>
- Intel Corporation. (s.f.). *Different Wi-Fi Protocols and Data Rates*. Obtenido de Intel: <https://www.intel.com/content/www/us/en/support/articles/000005725/network-and-i-o/wireless-networking.html>
- International Computer Science Institute. (14 de 02 de 2014). Recuperado el 06 de 01 de 2018, de <https://github.com/noxrepo/nox>

- Internet Engineering Task Force IETF. (06 de 2011). *Network Configuration Protocol (NETCONF)*. Recuperado el 06 de 10 de 2018, de <https://tools.ietf.org/html/rfc6241.html>
- Internet Engineering Task Force IETF. (06 de 2012). *An Overview of the IETF Network Management Standards*. Recuperado el 06 de 10 de 2018, de <https://tools.ietf.org/html/rfc6632#page-19>
- Internet Engineering Task Force IETF. (08 de 2016). *the YANG 1.1 Data Modeling Language*. Recuperado el 06 de 10 de 2018, de <https://tools.ietf.org/html/rfc7950>
- Internet Research Task Force - IRTF. (01 de 2015). *Software-Defined Networking (SDN): Layers and Architecture Terminology*. Recuperado el 20 de 09 de 2018, de <https://tools.ietf.org/html/rfc7426>
- Minghui, S., Xuemin, S., & Jon, M. (2004). IEEE802.11 roaming and authentication in wireless LAN / Cellular mobile networks. *IEEE*, 66-75.
- Mirchev, A. (2015). Survey of Concepts for QoS improvements via SDN. *Seminars FI / IITM SS 15*. Munich.
- Nippon Telegraph and Telephone Corporation. (s.f.). *Getting Started - What's Ryu*. Recuperado el 06 de 01 de 2018, de https://ryu.readthedocs.io/en/latest/getting_started.html
- Northbound Networks. (s.f.). *Zodiac WX*. Obtenido de Northbound Networks: <https://northboundnetworks.com/products/zodiac-wx>
- Open Networking Foundation. (26 de 03 de 2015). *OpenFlow Switch Specification*. Recuperado el 17 de 10 de 2018, de <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>
- Open Networking Foundation. (10 de 2015). *TR-518 Relationship of SDN and NFV*. Recuperado el 17 de 09 de 2018, de https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/onf2015.310_Architectural_comparison.08-2.pdf
- Open Networking Foundation. (10 de 2016). *Intent NBI - Definition and Principles*. Recuperado el 17 de 09 de 2018, de https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-523_Intent_Definition_Principles.pdf
- Open Networking Foundation. (2016). *SDN Architecture*. Recuperado el 13 de 09 de 2018, de https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521_SDN_Architecture_issue_1.1.pdf

- Open Networking Foundation. (15 de 03 de 2016). *SDN Architecture for Transport Networks*. Recuperado el 17 de 09 de 2018, de https://www.opennetworking.org/wp-content/uploads/2014/10/SDN_Architecture_for_Transport_Networks_TR522.pdf
- Panda Software. (2005). *Seguridad en redes inalámbricas*. Recuperado el 13 de 01 de 2018, de http://ocw.upm.es/teoria-de-la-senal-y-comunicaciones-1/comunicaciones-moviles-digitales/contenidos/Documentos/WP_wifi_PSE.pdf
- Paredes, M., Urbina, W., & Espinosa, N. (s.f.). *Implementación de un plan piloto de seguridad bajo el protocolo 802.1X para el Departamento de Gestión Tecnológica del Ministerio de Telecomunicaciones y Sociedad de la Información*. Recuperado el 12 de 01 de 2019, de <https://repositorio.espe.edu.ec/bitstream/21000/7251/1/AC-RED-ESPE-047388.pdf>
- Riverbed Technology, Inc. (s.f.). *Riverbed Airpcap*. Obtenido de Riverbed Support: <https://support.riverbed.com/content/support/software/steelcentral-npm/airpcap.html>
- Ryu SDN Framework Community. (2017). *Build SDN Agilely*. Recuperado el 06 de 01 de 2018, de <https://osrg.github.io/ryu/>
- Salazar, J. (s.f.). *Redes Inalámbricas*. Recuperado el 13 de 01 de 2019, de https://upcommons.upc.edu/bitstream/handle/2117/100918/LM01_R_ES.pdf
- Serrano, D. (2015). *Redes Definidas por Software (SDN): OpenFlow*. Recuperado el 12 de 09 de 2018, de [https://riunet.upv.es/bitstream/handle/10251/62801/SERRANO%20-%20Redes%20Definidas%20por%20Software%20\(SDN\):%20OpenFlow.pdf?sequence=3](https://riunet.upv.es/bitstream/handle/10251/62801/SERRANO%20-%20Redes%20Definidas%20por%20Software%20(SDN):%20OpenFlow.pdf?sequence=3)
- The POX network software platform*. (s.f.). Recuperado el 06 de 01 de 2018, de <https://github.com/noxrepo/pox>
- TheLinuxFoundationProjects. (s.f.). *OpenDaylight Downloads*. Recuperado el 06 de 01 de 2018, de <https://docs.opendaylight.org/en/latest/downloads.html>
- Trema. (s.f.). *Trema*. Recuperado el 06 de 01 de 2018, de <http://trema.github.io/trema/>
- University of Kentucky. (27 de 07 de 1999). *Architectural Framework for Active Networks*. Obtenido de <https://pdfs.semanticscholar.org/f6f2/1ca1cb1b32b00bf3004fb1139ec711546809.pdf>
- Wi-Fi Alliance. (s.f.). *How does a client roam?* Obtenido de Wi-Fi Alliance: <https://www.wi-fi.org/knowledge-center/faq/how-does-a-client-roam>

- Yáñez, C., & Gallegos, F. (2015). *Implementación de un prototipo de red definida por software para el hotspot-epoch mediante un controlador basado en openflow*. Recuperado el 13 de 11 de 2017, de <http://dspace.esPOCH.edu.ec/handle/123456789/4388>
- Yáñez, C., & Gallegos, F. (06 de 06 de 2016). *Implementación de un prototipo de Red Definida por Software para el Hotspot-Epoch mediante un controlador basado en OpenFlow*. Obtenido de <http://dspace.esPOCH.edu.ec/handle/123456789/4388>
- Zander, J., & Forchheimer, R. (1988). The SOFTNETProjeCt: A Retrospect . *IEEE*, 343-345.
- Zapata, C. (s.f.). *Topologías Inalámbricas*. Recuperado el 20 de 11 de 2017, de http://200.24.22.78/servicios/CD2/SEMANA_8.pdf