

IMPLEMENTACIÓN DE UN CONTROLADOR DE CAPA CRUZADA BAJO EL PARADIGMA SDR/SDN PARA REDES INALÁMBRICAS

Autores:
Julián Andrés Riveros Camargo
Nelson Damián Gómez García

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería, Proyecto curricular de Ingeniería Electrónica
Agosto 2020

IMPLEMENTACIÓN DE UN CONTROLADOR DE CAPA CRUZADA BAJO EL PARADIGMA SDR/SDN PARA REDES INALÁMBRICAS

Julián Andrés Riveros Camargo
Nelson Damián Gómez García

Trabajo de grado como requisito parcial para optar al título de:
Ingeniero Electrónico en modalidad de Investigación/Innovación

Director:
Ph.D Gustavo Adolfo Puerto Leguizamón

Linea de investigación:
Virtualización de redes y telecomunicaciones

Grupo de investigación:
Grupo de radiación electromagnética y comunicaciones ópticas (GRECO)

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería, Proyecto curricular de Ingeniería Electrónica
Agosto 2020

Dedicado a Luis Vicente Gómez Pinzón, una persona ejemplar con un legado imborrable. A pesar de que su vida no fue suficiente para lograr ver al primer ingeniero dentro de su familia, espero que desde donde se encuentre pueda estar feliz y se sienta orgulloso de la persona que ayudó a formar, intentaré llevar a cabo todas sus enseñanzas y hacer hueco donde quiera que me pare. Gracias por todo, Descanse en paz abuelito.

Nelson Damián Gómez García

Dedicado a nuestras familias por apoyarnos a lo largo de todo el proceso de formación que nos ha llevado a ser las personas que somos y los profesionales que pronto seremos.

Julián Andrés Riveros Camargo

Agradecimientos

Gracias a Dios y a la vida por permitirnos estar en este mundo, a nuestra familia por apoyarnos en este largo camino, a toda la comunidad de la Universidad Distrital Francisco José de Caldas que nos permitió hacer de este lugar nuestro segundo hogar, Agradecemos a todos los docentes que aportaron en nuestra formación como profesionales.

ÍNDICE

I. Generalidades **1**

1. Introducción 1
2. Justificación 2
3. Planteamiento de problema 3
4. Objetivos 3

II. Marco referencial **4**

1. Marco teórico 4
   - 1.1. SDN (Software defined network) 4
   - 1.2. SDWN (Software defined wireless network) 5
   - 1.3. SDR (Software defined radio) 5
   - 1.4. Redes Centralizadas 7
   - 1.5. Redes Distribuidas 8
   - 1.6. Controlador Opendaylight 8
   - 1.7. Protocolo OpenFlow 10
   - 1.8. Mininet 13
   - 1.9. Miniedit 16
2. Estado del arte 17
   - 2.1. Integración de SDR y SDN para 5G 17
   - 2.2. Asignación y programación de recursos de red para múltiples aplicaciones de control en SDN 19
   - 2.3. Un sistema de monitoreo de radiodifusión de FM de múltiples estaciones basado en SDR 22
   - 2.4. Un marco basado en SDN / NFV para la gestión y el despliegue de una red central 5G basada en servicios 25
   - 2.5. CrossFlow: una arquitectura de capas cruzadas para SDR utilizando los principios de SDN 28
   - 2.6. Integración de la virtualización de la función de red con SDR y SDN para redes 4G / 5G 29

III. Implementación del controlador y configuración del entorno de prueba **32**

1. Instalación de complementos necesarios 32
   - 1.1. Instalación Opendaylight(ODL) 32
   - 1.2. Instalación Mininet 36
   - 1.3. Instalación de Mininet-WiFi 38
   - 1.4. Interfaz Gráfica Miniedit 39

1.5. Conexión entre Opendaylight y Mininet-WiFi 43
1.6. Uso de Mininet-WiFi en conjunto con scripts de Python 46

IV. Evaluación y Resultados **48**
1. Potencia 48
1.1. Rango de la señal 48
1.2. Ganancia de la antena 52
1.3. Potencia de la señal 55
2. Velocidad o tasa de transmisión 58
3. Frecuencia de trabajo 62
4. Scripts de Python para modificar los parámetros 64
5. Escenario 1: Múltiples puntos de acceso 64
6. Escenario 2: Handover 71
6.1. Pruebas con tasa de pérdida de paquetes y retardos 72
6.2. Pruebas con la distancia y nivel de señal 73
7. Escenario 3: Ad-Hoc 77
8. Escenario 4: Red mixta SDN y Ad-Hoc 81

V. Conclusiones **87**

Anexos **89**

Índice de figuras

II.1. Red convencional vs una red definida por software [1] 4
II.2. Interfaz SDWN [2] 5
II.3. Estructura básica de un sistema SDR [3] 6
II.4. Ejemplo red centralizada [5] 7
II.5. Red P2P [6] 7
II.6. Ejemplo red de conmutación distribuida [8] 8
II.7. Campos de cabecera de paquete en la versión 1.0.0 del protocolo OpenFlow [12] 11
II.8. Campos de cabecera de paquete en la versión 1.0.0 del protocolo OpenFlow [12] 12
II.9. Campos de cabecera de paquete en la versión 1.5.0 del protocolo OpenFlow [12] 13
II.10. Comandos Mininet. Fuente:Autor 14
II.11. Componentes de Mininet WiFi [14] 15
II.12. Interfaz MiniEdit para Mininet Wi-fi. Fuente:Autor 16
II.13. Arquitectura por capas de redes 4G y redes 5G [15] 17
II.14. Bosquejo de la arquitectura SDN/ SDR implementada [15] 18
II.15. Uso de ancho de banda vs espectro [15] 18
II.16. Latencia vs tiempo en minutos [15] 19
II.17. Tiempo de inserción vs numero de dispositivos [15] 19
II.18. Implementación y Diseño VFS [16] 20
II.19. Rendimiento de VFS en comparación a otro modelo existente (MPLSTE) [16] 21
II.20. Tasa de pérdida de caché en función de número de flujos populares en caché para el modelo propio, el modelo óptimo y otros modelos existentes [16] 21
II.21. Diagrama de bloques de SDR [17] 22
II.22. Interfaz OpenWebRX [17] 23
II.23. Diseño sistema con implementación SDR y OpenWebRX [17] 24
II.24. Capacidad de cómputo [17] 24
II.25. Velocidad de internet [17] 24
II.26. Comparación entre el modelo SDN/NFV y otros modelos existentes [18] . 25
II.27. Arquitectura SDN/NFV [18] 26
II.28. Arquitectura de corte de red [18] 26
II.29. Costos en función del precio de la energía de servidores de borde [18] . . 27
II.30. Costos totales en función del tamaño de la solicitud de la red [18] 27
II.31. Diseño controlador capa cruzada [19] 29

II.32. Arquitectura de integración SDN SDR NFV [20] 30
II.33. Evolución en las redes móviles [20] 31
II.34. Integración de tecnologías [20] 31

III.1. Página oficial de Opendaylight. Fuente: Autor. 33
III.2. Distribuciones disponibles de ODL. Fuente: Autor. 33
III.3. Distribuciones disponibles para la versión 0.4.0-Beryllium de ODL. Fuente: Autor. 34
III.4. Versión Java. Fuente: Autor. 34
III.5. Inicialización de l controlador Opendaylight. Fuente: Autor. 35
III.6. Instalación de complementos para ODL. Fuente: Autor. 35
III.7. Página oficial de Mininet.Fuente: Autor. 36
III.8. Opciones de descarga de Mininet. Fuente: Autor. 36
III.9. Comandos para la especificación de la versión de mininet a instalar. Fuente: Autor. 37
III.10Prueba final de la instalación de Mininet. Fuente: Autor. 37
III.11Inicialización de la red por defecto de Mininet-WiFi. Fuente: Autor. . . . 38
III.12Prueba de la red por defecto de Mininet-WiFi. Fuente: Autor. 39
III.13Ejemplo de Topología utilizando Miniedit. Fuente: Autor. 40
III.14Preferencias de la red en Miniedit. Fuente: Autor. 40
III.15Configuración del controlador SDN en Miniedit. Fuente: Autor. 41
III.16Configuración del Access Point en Miniedit. Fuente : Autor. 42
III.17Configuración de la Estación Móvil en Miniedit. Fuente: Autor. 42
III.18Ventana para exportar la topología como un script de Python. Fuente: Autor. 43
III.19Inicialización por defecto Mininet-wifi utilizando controlador ODL. Fuente: Autor. 44
III.20Prueba de conexión de la red con el controlador ODL. Fuente: Autor. . . . 44
III.21Página de inicio interfaz gráfica Dlux. Fuente: Autor. 45
III.22Red creada y visualizada mediante la interfaz gráfica de ODL. Fuente: Autor. 46
III.23Red generada por el script de Python utilizado. Fuente: Autor. 47

IV.1. Red generada por el script de Python utilizado. Fuente: Autor. 49
IV.2. Información del enlace entre el punto de acceso y la estación 1. Fuente: Autor. 49
IV.3. Prueba 1 de tasa de pérdida de paquetes en la conexión entre estaciones. Fuente: Autor. 50
IV.4. Variación de Rango de ap1. Fuente: Autor. 50
IV.5. Estado del enlace entre estaciones y el punto de acceso luego del cambio de rango de ap1. Fuente: Autor. 51
IV.6. Prueba 2 de tasa de pérdida de paquetes en la conexión entre estaciones. Fuente: Autor. 52
IV.7. Escenario para pruebas con ganancia de antena. Fuente: Autor. 53
IV.8. Ganancias por defecto de las estaciones y el punto de acceso. Fuente: Autor 53
IV.9. Estado del enlace entre sta1 y ap1 luego de aumentar la ganancia en ap1. Fuente: Autor 54

IV.10.Estado del enlace entre sta1 y ap1 luego de aumentar la ganancia en sta1. Fuente: Autor. 54
IV.11.Estado del enlace entre sta1 y ap1 luego de disminuir la ganancia en sta1. Fuente: Autor. 55
IV.12.Potencias de transmisión por defecto para puntos de acceso y estaciones. Fuente: Autor. 55
IV.13.Estado inicial del enlace entre sta3 y ap1. Fuente: Autor. 56
IV.14.Estado inicial del enlace entre sta3 y ap1. Fuente: Autor. 56
IV.15.Estado del enlace entre sta3 y ap1 al aumentar la potencia en sta3. Fuente. Autor. 57
IV.16.Estado del enlace entre sta3 y ap1 al disminuir la potencia de ap1. Fuente: Autor. 57
IV.17.Condiciones iniciales experimento con estándar IEEE 802.11g canal 1. Fuente: Autor. 59
IV.18.Prueba de pérdida de paquetes para condiciones iniciales con estándar IEEE 802.11g canal 1. Fuente: Autor. 59
IV.19.Test de ancho de banda y estado del enlace para condiciones iniciales. Fuente: Autor. 60
IV.20.Cambio de velocidad de transmisión en el punto de acceso. Fuente: Autor. 61
IV.21.Cambio de velocidad de transmisión en la estación 1. Fuente: Autor. . . . 61
IV.22.Prueba de pérdida de paquetes luego de experimentación con velocidad de transmisión. Fuente: Autor. 62
IV.23.Evolución del estándar IEEE 802.11 [21]. 63
IV.24.Configuración modo de operación y canal. Fuente: Autor. 64
IV.25.Estado del enlace estación Sta1 y Punto de acceso AP1. Fuente: Autor. . . 64
IV.26.Escenario 1: Múltiples puntos de acceso. Fuente: Autor. 65
IV.27.Condiciones iniciales escenario 1. Fuente: Autor. 65
IV.28.Prueba de conexión escenario 1. Fuente: Autor. 66
IV.29.Prueba de retardos escenario 1. Fuente: Autor. 66
IV.30.Estado del enlace estación 1 escenario 1. Fuente: Autor. 67
IV.31.Estado del enlace estación 2 escenario 1. Fuente: Autor. 67
IV.32.Estado del enlace estación 3 escenario 1. Fuente: Autor. 67
IV.33.Tasa de pérdida de paquetes entre la estación 4 y la estación 3. Fuente: Autor. 69
IV.34.Tasa de pérdida de paquetes entre la estación 2 y la estación 1. Fuente: Autor. 69
IV.35.Tasa de pérdida de paquetes entre la estación 6 y la estación 5. Fuente: Autor. 69
IV.36.Retardo vs paquetes con cambios en términos de potencia y tasa de transmisión. Fuente: Autor. 70
IV.37.Escenario 2 ubicaciones iniciales. Fuente: Autor. 71
IV.38.Escenario 2 ubicaciones finales. Fuente: Autor. 72
IV.39.Sta5 asociada con ap4. Fuente: Autor. 74
IV.40.Sta5 asociada con ap3. Fuente: Autor. 74
IV.41.Sta4 asociada con ap4. Fuente: Autor. 76
IV.42.Sta4 asociada con ap5. Fuente: Autor. 76

IV.43.Sta4 asociada con ap2. Fuente: Autor. 76
IV.44.Topología de la red Ad-Hoc. Fuente:Autor 78
IV.45.Inicialización protocolo OLSRD para las estaciones base. Fuente: Autor. . 78
IV.46.Red Ad-Hoc con movilidad y handover. Fuente: Autor. 79
IV.47.Prueba comunicación sta7-sta4. Fuente: Autor. 80
IV.48.Envío de paquete de estación móvil a estación móvil. Fuente: Autor . . . 81
IV.49.Topología red mixta. Fuente: Autor. 81
IV.50.Configuración tarjetas de red. Fuente: Autor. 82
IV.51.Estado de conexión de red sta5 y sta7. Fuente: Autor. 83
IV.52.Prueba de conexión entre estaciones. Fuente: Autor. 83
IV.53.Topología de red mixta para prueba de conexión. Fuente: Autor. 84
IV.54.Estado de tarjetas de red estaciones sta1 y sta6. Fuente: Autor. 84
IV.55.Prueba de mensajes ICMP a estación base dentro de la red SDN. Fuente: Autor. 85
IV.56.Solicitudes ICMP (ping) con estaciones asociadas a diferentes puntos de acceso. Fuente: Autor. 86
IV.57.Puntos de acceso asociados a las estaciones sta5 y sta2. Fuente: Autor. . . 86

Índice de tablas

4.1. Estándares WiFi, velocidad de transmisión máxima teórica y frecuencias de trabajo [21]. 58
4.2. Métodos de codificación/modulación implementados en estándares WiFi [21]. 63
4.3. Resultados obtenidos con condiciones iniciales escenario 1. Fuente: Autor. 68
4.4. Nuevos valores para pruebas de pérdida de paquetes y retardos entre estaciones con múltiples puntos de acceso. Fuente: Autor. 68
4.5. Valores relevantes en la prueba de retardos entre estaciones. Fuente: Autor. 70
4.6. Velocidades de las estaciones móviles. Fuente: Autor. 72
4.7. Resultados comparativos para 3 diferentes situaciones de handover. Fuente: Autor. 73
4.8. Datos tomados del recorrido de sta5 a lo largo de la emulación. Fuente: Autor. 75
4.9. Toma de datos para el recorrido de sta4 a lo largo de la emulación. Fuente: Autor. 77
4.10. Resultados prueba ping estación móvil sta2. Fuente: Autor. 79
4.11. Resumen prueba comunicación sta7-sta4. Fuente: Autor. 80

Capítulo I

Generalidades

1. Introducción

Hoy en día es un hecho innegable que las redes móviles que asignan recursos de manera dinámica son el futuro de las telecomunicaciones. La tecnología utilizada actualmente para las redes móviles es la 4G LTE (LTE responde a las siglas Long Time Evolution), que hasta el momento satisface las necesidades de los usuarios, pero, ¿Qué pasará cuando la creciente demanda supere esta tecnología?, ¿Cuál es el siguiente paso luego de la 4G?, estas son unas muy buenas preguntas, a lo cual la respuesta es solo una, la implementación de redes 5G. Las llamadas redes 5G son una tecnología innovadora diseñada para las telecomunicaciones del futuro, donde la red global asignará recursos de forma dinámica, además de que todo estará conectado, un ejemplo claro de esto es el internet de las cosas (con las siglas en inglés IoT). La tecnología 5G está pensada para ser completamente diferente a sus antecesoras (1G a 4G LTE), en el sentido de que se enfocará en las redes móviles y que tendrá una arquitectura centralizada, a diferencia de la arquitectura distribuida que se venía manejando, todo con el fin de optimizar los recursos malgastados en la 4G y lograr velocidades y cobertura superior.

Teóricamente hablando, se ha establecido que la forma más efectiva de desarrollar redes 5G que logren cumplir con las predicciones que se tienen a futuro, es la red 5G implementada con una interfaz entre la Radio Definida por Software (SDR) y las Redes Definidas por Software (SDN). Para contextualizar la SDR es un sistema de radiocomunicaciones donde varios de los componentes típicamente implementados en hardware como mezcladores, filtros, moduladores/demoduladores, detectores, etc. son implementados en software. Mientras que SDN es un conjunto de técnicas cuyo objetivo es facilitar la implementación e implantación de servicios de red. Todo esto mediante la separación del plano de control (software) del plano de datos (hardware).

En conclusión, en un primer vistazo, es factible afirmar que realizar el encaminamiento de las redes 5G mediante desarrollo que integre SDR y SDN es un camino muy prometedor y con buena probabilidad de éxito en el cual se puede pasar de la base teórica que se tiene actualmente a una implementación real.

2. Justificación

En la actualidad el uso de dispositivos móviles es indispensable en la vida cotidiana de cualquier persona, lo que conlleva a una gran demanda de servicios, que además de ser masiva, también va en crecimiento exponencial, esto hace difícil el tener tecnología que este a la par con ella, debido a sus requerimientos, como por ejemplo, baja latencia, alta tasa de transmisión, alto ancho de banda, etc. Y todo esto trae a la luz una necesidad de estar avanzando de manera continua (en términos de tecnología). Este trabajo se realiza con la finalidad de implementar un modelo de redes (con la unión SDR y SDN) que solo tiene bases teóricas hasta el día de hoy, y que significaría un avance enorme en las telecomunicaciones móviles de hoy y del futuro.

Los beneficios que se aportarán con redes 5G no solo tienen que ver con el aumento de velocidad con respecto a anteriores tecnologías, aunque los usuarios de estas tendrán el beneficio anteriormente mencionado, también contarán con tiempos de latencia mucho menores, además de tener una calidad de servicio (QoS en inglés) a la par o mejor que en la tecnología 4G LTE. En cuanto a los prestadores de servicios, los beneficios que darán las redes 5G están en la optimización en el uso de los recursos, reducción de costos además de la centralización del control. Los últimos beneficiados con las redes 5G serán las nuevas tecnologías, ya que se contará con servicios capaces de respaldarlos, un buen ejemplo que tiene lugar es el internet de las cosas, donde todo tiene que estar interconectado en todo momento. De todo lo anteriormente mencionado se puede concluir que es necesaria una red que soporte el ritmo del avance tecnológico actual, además de la demanda por parte de los usuarios, y la red 5G con implementación mediante SDR y SDN es la que está más cerca de cumplir con todo lo requerido.

3. Planteamiento de problema

La tecnología actual es incapaz de cubrir la creciente demanda en términos de tasa de transmisión, latencia, ancho de banda, calidad de servicio, entre otros parámetros que caracterizan una red. Además, la implementación de una red de parámetros distribuidos acarrea grandes costos en cuanto hardware, capital, instalación, mantenimiento.

Un sistema de telecomunicaciones requiere una gran cantidad de hardware, este es suministrado por múltiples fabricantes los cuales usan diferentes especificaciones de diseño, causando problemas al momento de integrar dicho hardware en el sistema de telecomunicaciones.

La implementación de un método capaz de comunicar dos tecnologías (SDN y SDR) diseñadas para operar en diferentes capas del modelo OSI conlleva una gran dificultad debido a los tipos de protocolo que maneja cada capa. ¿Realizar un controlador de capa cruzada con base en los principios de SDN aplicados a SDR puede solucionar los problemas presentes de las redes que actualmente operan, cumpliendo con las expectativas que la base teórica de la red 5G propone?

4. Objetivos

General

Diseñar e implementar un controlador de capa cruzada donde se unifiquen los protocolos de radio definida por software y las redes definidas por software para que sirva como base para la tecnología 5G.

Específicos

- Comprender el funcionamiento y las características que se manejan en la radio definida por software, además de sus aplicaciones y los elementos que lo componen.
- Realizar un acercamiento con las redes definidas por software, su funcionamiento, sus características, ventajas, desventajas y sus aplicaciones.
- Diseñar un método capaz de realizar una integración entre las capas en las que se encuentran las redes definidas por software y las capas donde opera la radio definida por software
- Evaluar el comportamiento de una red que contenga las bases de la radio definida por software y las redes definidas por software.

Capítulo II

Marco referencial

1. Marco teórico

1.1. SDN (Software defined network)

La red definida por software (SDN) es una arquitectura emergente para diseñar y operar redes y servicios de comunicación. Proporciona una vista centralizada de la red que se puede configurar según la demanda y ha estimulado el desarrollo de aplicaciones de red sofisticadas debido a que realiza una separación del plano de control y el plano de datos. Si bien es una herramienta transformadora, la investigación se ha centrado principalmente en SDN en redes cableadas. Desafortunadamente, las redes inalámbricas no han recibido mucha atención en el contexto de SDN.

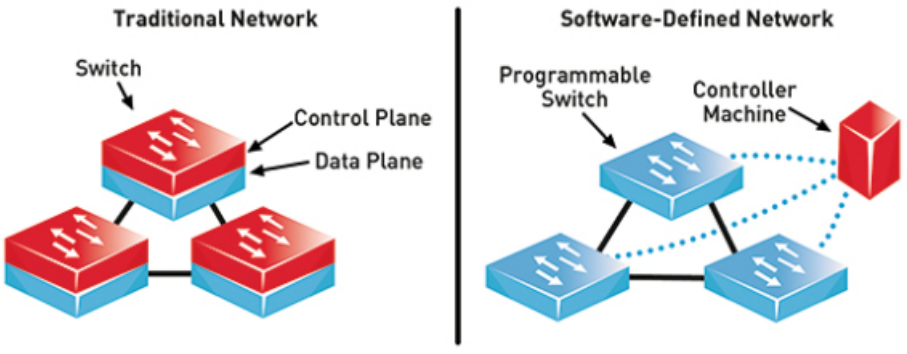


Figura II.1: Red convencional vs una red definida por software [1]

En la Figura II.1 se muestra el funcionamiento de una red tradicional, en la cual en cada nodo se encuentra el plano de control y de datos, además del switch, en una red definida por software un nodo central administra el plano de control y de datos, dejando en cada uno de los nodos únicamente un switch que sólo tiene la función de conmutar paquetes como le indique la máquina de control.

1.2. SDWN (Software defined wireless network)

Con las mismas bases de SDN, las Redes Inalámbricas Definidas por Software (SDWN), tienen como objetivo proporcionar un control centralizado de puntos de acceso (AP), que cumplen las instrucciones recibidas y siguen siendo responsables del envío y recepción del tráfico en los enlaces inalámbricos. SDWN es una rama de investigación emergente en el campo de las SDN, las cuales mejoran la flexibilidad, la capacidad de administración y la facilidad del uso de entornos de redes inalámbricas [2]. En la Figura II.2 se muestra una aproximación a lo que es la interfaz que compone a una SDWN.

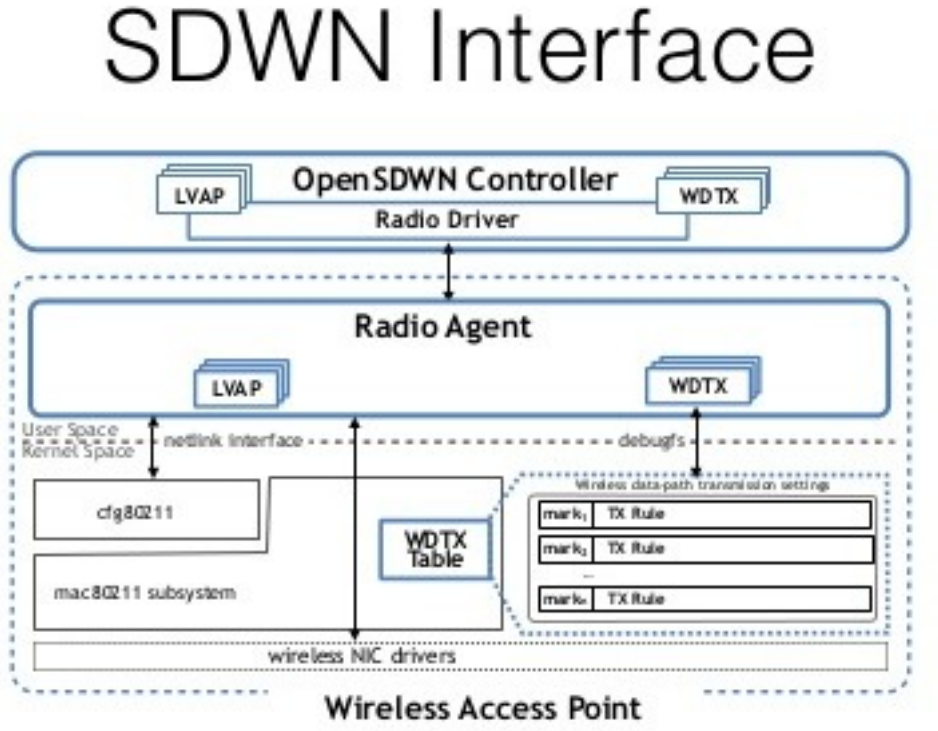


Figura II.2: Interfaz SDWN [2]

1.3. SDR (Software defined radio)

En su concepto más amplio, se conoce como software radio (SR) a aquellos radios multibanda capaces de soportar múltiples interfaces aire y protocolos a través del uso de antenas de banda ancha, conversiones de radiofrecuencia (RF), convertidores Analógico/-Digital (ADCs, del inglés Analog/Digital Converters) y Digital/Analógico (DACs). En los "software radios" ideales, todos los aspectos del radio (incluyendo la interfaz aire física) se definen por software que corre sobre procesadores de propósito general.

En muchos casos, esta implementación ideal no es realizable por razones que van desde el consumo energético hasta las estrictas y elevadas especificaciones del procesador, especialmente en altas frecuencias, de ahí que se siga el concepto de Radio Definido

por Software (SDR, del inglés Software Defined Radio), más práctico, el cual caracteriza los radios reconfigurables o reprogramables cuya funcionalidad se encuentra definida por software, capaces de implementar en el mismo hardware diferentes funcionalidades, utilizando para este fin los componentes electrónicos disponibles en el mercado, combinados convenientemente para suplir las limitaciones de los procesadores actuales. Como dichas funcionalidades se encuentran definidas en el software, resulta posible implementar nuevas tecnologías en un radio definido por software sencillamente realizando una actualización de software.

La radio definida por software (SDR) es una tecnología para la comunicación por radio, esta tecnología se basa en protocolos inalámbricos definidos por software, a diferencia de las soluciones basadas en hardware. Esto se traduce en el soporte de varias características y funcionalidades como la actualización y la actualización a través de la reprogramación, sin la necesidad de reemplazar el hardware en el que se implementan. Esto abre la puerta a la posibilidad de realizar dispositivos inalámbricos multibanda y multifuncionales [3].

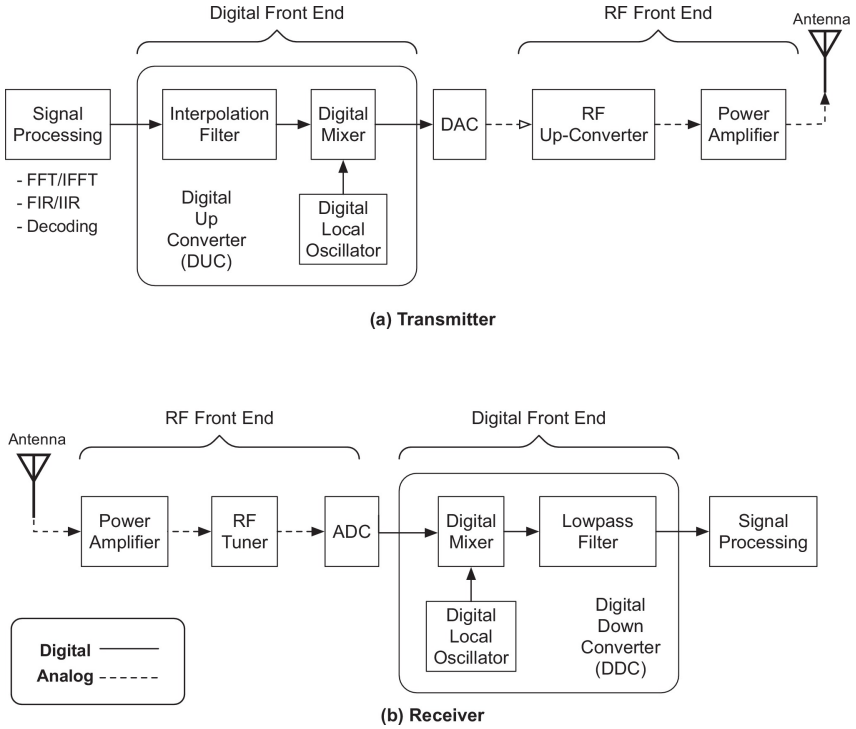


Figura II.3: Estructura básica de un sistema SDR [3]

En la Figura II.3 Se aprecia la estructura básica de un sistema de radio definido por software, en la parte (a) la estructura de un transmisor y en la parte (b) la de un receptor. Un equipo SDR contiene una serie de bloques básicos en su arquitectura que se pueden clasificar en tres secciones elementales: la sección de RF, la sección de FI y la sección de banda base.

La sección RF es la encargada de hacer la transmisión y/o recepción de la señal, en la sección FI se realiza la conversión Analógica-Digital o Digital-Analógica según sea el caso, el filtrado, y el tratamiento necesario de la señal, por último en la sección de banda

base se realiza el procesamiento de la señal en banda base a través de un microprocesador como una FPGA, DSP, etc.

1.4. Redes Centralizadas

En una red centralizada, todos los nodos son periféricos, salvo el central. Estos nodos únicamente pueden comunicarse a través del nodo central y sus canales. Si el nodo central sufre una caída, el resto de nodos dejan de tener flujo [4].

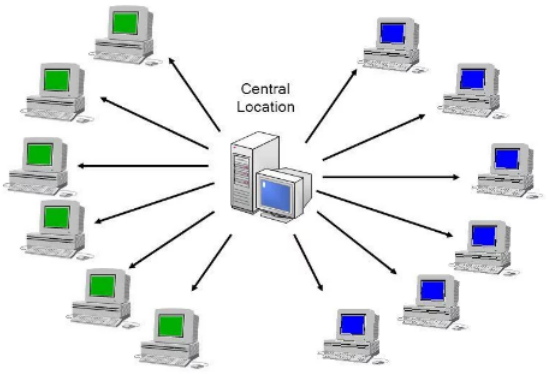


Figura II.4: Ejemplo red centralizada [5]

La red centralizada se rige por el principio del conocimiento, es decir que el receptor conoce el mensaje del emisor. Hasta ahora ha sido la más utilizada, sin embargo está siendo sustituida por la descentralizada y la distribuida, en la Figura II.4, se puede ver una máquina central, que regula y controla todo tipo de comunicación dentro de la red.

Redes P2P centralizadas Se basan en una arquitectura monolítica, es decir, todas las transacciones se hacen a través de un único servidor que a su vez hace de punto de enlace entre dos nodos. Igualmente almacena los nodos y los distribuye donde se almacenan los contenidos [4].

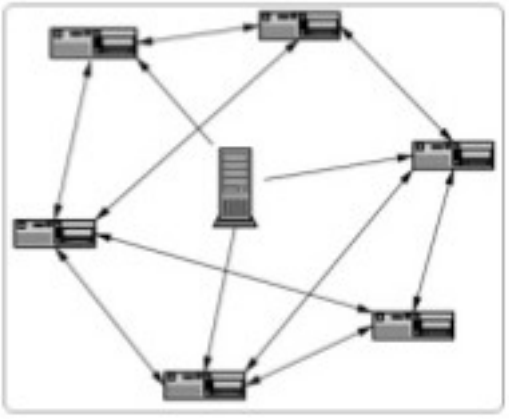


Figura II.5: Red P2P [6]

En una red P2P cualquier máquina puede conectarse con el servidor o con otra máquina como se puede ver en la Figura II.5.

1.5. Redes Distribuidas

Las redes distribuidas suponen un cambio en el paradigma ya que el poder de la decisión se convierte en automático en la mayoría [7].

Es un tipo de red cuya característica principal es la ausencia de un centro individual o colectivo. Los nodos se unen uno a otro de tal forma que ninguno de ellos tiene poder de filtro sobre la información que se transmite en la red, en consecuencia desaparece la idea de centro y periferia, características básicas en las redes centralizada y descentralizada.

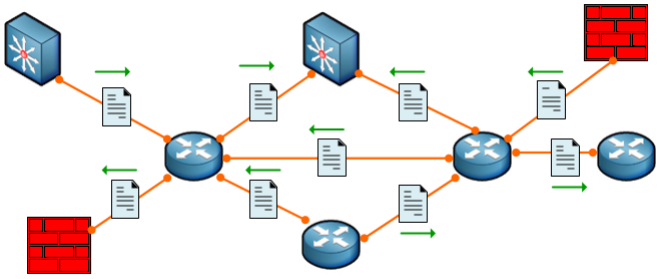


Figura II.6: Ejemplo red de conmutación distribuida [8]

Si cayera algún nodo, no desconectaría a ningún otro, por lo tanto la convierte en una red práctica, robusta y eficiente. La red se rige por el principio de la interacción, cada nodo es independiente y puede moverse libremente [4].

La Internet asienta sus bases sobre redes distribuidas en las cuales cada nodo es independiente y capaz de analizar la ruta más óptima de enrutamiento de un datagrama a su destino, y si es necesario cambiar dicha ruta. En la Figura II.6 se puede ver como cada uno de los nodos (routers) es capaz de elegir el direccionamiento que le da a cada paquete de datos.

1.6. Controlador Opendaylight

OpenDaylight (ODL) es una plataforma abierta modular para personalizar y automatizar redes de cualquier tamaño y escala. El Proyecto OpenDaylight surgió del movimiento SDN, con un claro enfoque en la capacidad de programación de la red. Fue diseñado desde el principio como base para soluciones comerciales que abordan una variedad de casos de uso en entornos de red existentes.

OpenDaylight es la plataforma de controlador SDN de código abierto más ampliamente implementada y en solo 6 años, OpenDaylight cuenta con 10 lanzamientos, más de 1000

autores/remitentes, mas de 1 billón de suscriptores al rededor del mundo [9].

La plataforma ODL está diseñada para permitir a los usuarios intermedios y proveedores de soluciones la máxima flexibilidad en la construcción de un controlador que se ajuste a sus necesidades. El diseño modular de la plataforma ODL permite que cualquier persona en el ecosistema ODL aproveche los servicios creados por otros; para escribir e incorporar los suyos; y para compartir su trabajo con otros. ODL incluye soporte para el conjunto más amplio de protocolos en cualquier plataforma SDN: OpenFlow, OVSDB, NETCONF, BGP y muchos más, que mejoran la programabilidad de las redes modernas y resuelven una variedad de necesidades de los usuarios.

1.6.1. Versiones OpenDayligth

OpenDayligth Lithium (Li) Es la tercera versión de OpenDaylight, la plataforma de código abierto para construir redes programables definidas por software. Con Lithium, los proveedores de servicios y las empresas pueden hacer la transición a SDN con un enfoque particular en ampliar la programabilidad de las redes inteligentes. Pueden componer sus propias arquitecturas de servicios o aprovechar una oferta comercial basada en OpenDaylight para ofrecer servicios de red dinámicos en un entorno en la nube, elaborar políticas dinámicas basadas en la intención y comenzar a virtualizar funciones con Service Function Chaining (SFC).

OpenDayligth Beryllium (Be) es la cuarta versión de OpenDaylight (ODL), la plataforma de código abierto líder para redes programables definidas por software. De múltiples proveedores, tradicional y ecológico, ODL es la plataforma SDN de facto de la industria, que admite un amplio conjunto de casos de uso y proporciona la base para las redes del futuro [10].

OpenDayligth Boro (Bo) Es la quinta versión de OpenDayligth (ODl), Boro aporta un fuerte enfoque práctico en dos tipos principales de implementaciones, con mejoras en la nube y el soporte de virtualización de funciones de red (NFV), así como la ingeniería de redes a gran escala. Las nuevas herramientas operativas, el rendimiento mejorado y la documentación mejorada completan la experiencia de implementación mejorada. Boro también proporciona nuevas herramientas y documentación para apoyar a los desarrolladores de aplicaciones, así como una mayor integración con marcos industriales más grandes, desde OPNFV y OpenStack hasta CORD y Atrium Enterprise.

OpenDayligth Carbono (C) Es la sexta versión de OpenDayligth (ODL), con Carbono, la visión de automatizar redes programables se enfoca y se expande a nuevos casos de uso. Las cadenas de herramientas fundamentales para la programación en la nube, NFV y el plano de gestión se han incorporado como componentes centrales de marcos de código abierto de nivel superior, como ONAP, OPNFV y OpenStack, así como implementaciones de diseños del mundo real de organismos de estándares como MEF. Estas nuevas pilas combinadas permiten cada vez más a los innovadores explorar productivamente nuevos casos de uso como IoT.

OpenDaylight Nitrógeno (Ni) Es la séptima versión de OpenDaylight (ODL), el enfoque principal para Nitrógeno es la implementación de Karaf 4; El componente OpenDaylight que permite al usuario elegir qué protocolos y servicios admitirá su controlador SDN. Con esta actualización, OpenDaylight mejora significativamente la gestión de las interdependencias entre sus paquetes de componentes.

OpenDaylight Oxígeno (O) Es la octava versión de OpenDaylight (ODL), con Oxígeno, OpenDaylight comienza el movimiento hacia un modelo de "distribución administrada", que desacopla los proyectos no centrales del proceso de lanzamiento principal y les permite evolucionar a su propio ritmo.

OpenDaylight Flúor (F) Es la novena versión de OpenDaylight (ODL), con Flúor, OpenDaylight ha hecho un movimiento significativo hacia la provisión de un paquete simplificado que elimina la necesidad de que los proveedores de soluciones realicen pruebas básicas e integración de los componentes principales, lo que les permite centrarse en la creación de capacidades diferenciadas y ofrecer una solución completa.

1.7. Protocolo OpenFlow

OpenFlow es un protocolo que permite a un servidor decirle a los conmutadores de red donde enviar paquetes. En una red convencional, cada conmutador tiene software propietario que le dice qué hacer. Con OpenFlow se centralizan las decisiones de migración de paquetes, de modo que la red se puede programar independiente de los conmutadores individuales y equipo del centro de datos [11].

En un conmutador convencional, la transferencia de paquetes (la trayectoria de datos) y el enrutamiento de alto nivel (la trayectoria de control) suceden en el mismo dispositivo. Un conmutador OpenFlow separa la trayectoria de datos de la trayectoria de control. La porción de trayectoria de datos reside en el propio conmutador; un controlador separado toma las decisiones de enrutamiento de alto nivel. El conmutador y el controlador se comunican por medio del protocolo de OpenFlow [11].

Un conmutador OpenFlow esta basado en el hecho de que la mayoría de los conmutadores y enrutadores Ethernet actuales contienen tablas de flujo para funciones como cortafuegos, calidad de servicio (QoS) o recogida de estadísticas. A pesar de que una tabla de flujo es diferente según el fabricante, existe un conjunto de funciones comunes. El dispositivo OpenFlow dispone de, al menos, tres partes: [12]

- Tabla de flujos: Cada entrada de la tabla dispone de campos en los que buscar coincidencias con los paquetes entrantes, contadores (para estadísticas de los paquetes) e instrucciones sobre qué hacer con los coincidentes.
- Canal seguro. Conecta el dispositivo al controlador, permitiendo el envío de comandos y paquetes entre ambos mediante el protocolo OpenFlow.

- Protocolo OpenFlow. Provee una manera abierta y estándar en la comunicación entre el conmutador y el controlador, permitiendo a este último insertar, eliminar, modificar y buscar en las entradas de la tabla de flujos a través del canal seguro.

1.7.1. Versiones de Openflow

El OpenFlowConsortium (un grupo colaborativo de investigadores universitarios y administradores de red liderado por miembros de Stanford y Berkeley), lanzó la primera implementación de referencia en noviembre de 2007, con la versión 0.1.0. Se continuó con versiones aún experimentales: 0.2.0, 0.8.0, 0.8.1, 0.8.2, 0.8.9, publicadas en 2008 y 0.9 a mediados de 2009 [12]. En diciembre de 2009 se lanzó la versión 1.0.0 del protocolo OpenFlow, la cual, hasta el momento, es la versión más ampliamente adoptada, que tiene en cuenta los campos de cabecera presentes en un paquete Ethernet mostrados en la Figura II.7:

Puerto entrante
Dirección MAC Origen
Dirección MAC Destino
Tipo Ethernet
Id VLAN
Prioridad CoS VLAN
IP Origen
IP Destino
Protocolo IP
Bits ToS (tipo de servicio) IP
Puerto TCP/UDP origen
Puerto TCP/UDP destino

Figura II.7: Campos de cabecera de paquete en la versión 1.0.0 del protocolo OpenFlow [12]

En la Figura II.8 se puede observar la estructura del conmutador OpenFlow con la especificación 1.0.0.

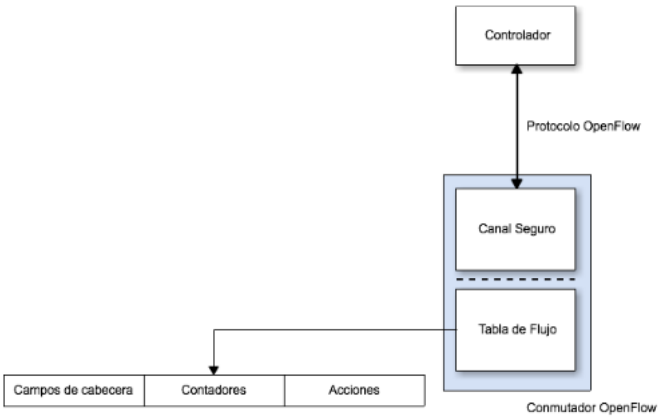


Figura II.8: Campos de cabecera de paquete en la versión 1.0.0 del protocolo OpenFlow [12]

La siguiente versión del protocolo OpenFlow es la especificación 1.1.0, en la cual, el conmutador tiene varias tablas de flujos conectadas entre ellas mediante procesamiento en 'pipeline' o tubería y una tabla de grupo. La tabla de grupo está diseñada para realizar operaciones que son comunes entre múltiples flujos, lo que significa que las acciones pertenecientes a un conjunto de ellos son agrupadas.

La especificación 1.2 incluye soporte a direcciones IPv6, así como la posibilidad de conectar un conmutador a múltiples controladores que pueden comunicarse también entre ellos, esto provee una recuperación más rápida durante fallos, asimismo, incorpora balance de carga. La especificación 1.3.0 incluye la posibilidad de controlar la tasa de paquetes mediante medidores de flujo, además se añaden conexiones auxiliares entre el conmutador y el controlador, también se incorporan cookies y campos de duración en las estadísticas. Por último, las especificaciones 1.4.0 y 1.5.0 añaden nuevas características como monitorización de flujos, "pipeline" o tubería sensible al tipo de paquetes y estadísticas extensibles en las entradas de las tablas de flujo [12]. En la Figura II.9, se puede observar la estructura de un conmutador OpenFlow con la especificación 1.5.0.

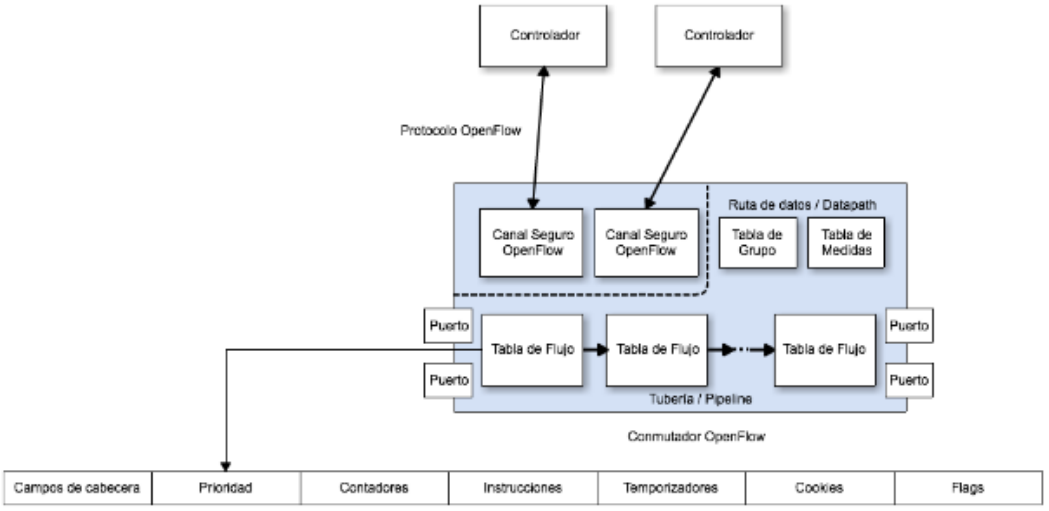


Figura II.9: Campos de cabecera de paquete en la versión 1.5.0 del protocolo OpenFlow [12]

1.8. Mininet

Mininet es un emulador de red , o quizás más precisamente un sistema de orquestación de emulación de red que ejecuta una colección de hosts finales, conmutadores, enrutadores y enlaces en un solo núcleo de Linux. Utiliza virtualización liviana para hacer que un solo sistema se vea como una red completa, ejecutando el mismo núcleo, sistema y código de usuario [13].

Al realizar una implementación con Mininet, los hosts virtuales, los conmutadores, los enlaces y los controladores de Mininet son reales, simplemente se crean utilizando software en lugar de hardware, y en su mayor parte su comportamiento es similar al de los elementos de hardware discreto. Por lo general, es posible crear una red Mininet que se parezca a una red de hardware, o una red de hardware que se parezca a una red Mininet, y ejecutar el mismo código binario y aplicaciones en cualquier plataforma [13].

1.8.1. Comandos de Mininet

Usando el comando Help se pueden visualizar todos los posibles comandos de la interfaz de Mininet como se puede apreciar en la figura II.10, a continuación se realiza una breve descripción de los comandos más usados para la simulación de una red.

```
mininet-wifi> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig
```

Figura II.10: Comandos Mininet. Fuente:Autor

- **EOF:** Comando para detener y salir de Mininet.
- **dpctl:** Es una utilidad de administración que permite cierto control sobre el conmutador OpenFlow, permitiendo agregar flujos a las tablas, consultar sus características y cambiar otras configuraciones.
- **dump:** Muestra la información de los nodos.
- **gterm y xterm:** Ambos comandos se emplean para abrir una ventana de comandos de un nodo.
- **help:** Comando para desplegar la ayuda genérica o de un comando en concreto.
- **intfs:** Lista las interfaces de los diferentes nodos.
- **iperf:** Realiza una prueba simple de TCP iperf entre dos hosts, permitiendo medir la velocidad de conexión máxima entre estos.
- **iperudp:** Realiza una prueba simple de UDP entre dos hosts, permitiendo medir la velocidad de conexión máxima entre estos.
- **link:** Inicia o termina la conexión entre dos nodos.
- **Links:** Muestra todas las conexiones establecidas en la red.
- **Pingall:** Se emplea para hacer un ping entre todos los hosts, devolviendo un resumen de todos los pings realizados.
- **Pingallfull:** Realiza un ping entre todos los hosts, devolviendo todos los resultados de las operaciones.
- **pingpair:** Comando útil para realizar pruebas entre los dos primeros hosts.
- **pingpairfull:** Realiza la misma función que el comando anterior, da como respuesta todos los resultados de esta operación.
- **ports:** Muestra la información de los puertos y las interfaces de los diferentes switch.
- **px:** Se emplea para ejecutar una declaración de Python.

- **py:** Se emplea para evaluar una expresión de Python.
- **source:** Se emplea para leer comandos de un archivo
- **switch:** Con este comando podemos inicializar o parar un switch.
- **time:** Permite medir el tiempo necesario para ejecutar un comando.

1.8.2. Mininet-WiFi

Mininet-WiFi es una bifurcación del emulador de red Mininet SDN y amplió la funcionalidad de Mininet al agregar estaciones de WiFi y puntos de acceso virtualizados basados en los controladores inalámbricos Linux estándar y el controlador de simulación inalámbrica 80211_hwsim. Esto significa que se han agregado nuevas clases para admitir la adición de estos dispositivos inalámbricos en un escenario de red Mininet y para emular los atributos de una estación móvil, como la posición y el movimiento en relación con los puntos de acceso [14].

Mininet-WiFi extiende la base del código Mininet agregando o modificando clases y scripts. Por lo tanto, Mininet-WiFi agrega una nueva funcionalidad y aún admite todas las capacidades de emulación SDN normales del emulador de red Mininet estándar [14].

1.8.3. Arquitectura y componentes

Los componentes principales que forman parte del desarrollo de Mininet-WiFi se ilustran en la figura II.11, donde se muestra que este emulador se divide en 2 espacios, el espacio del usuario y el espacio del kernel. En el espacio del kernel, el módulo mac80211_hwsim es responsable de crear interfaces virtuales de WiFi, importantes para estaciones y puntos de acceso, además, en el espacio del kernel, la gestión de subcapa de control de acceso a medios (MLME) se realiza en el lado de las estaciones, mientras que en el espacio del usuario el hostapd es responsable de esta tarea en el lado del punto de acceso (AP) [14].

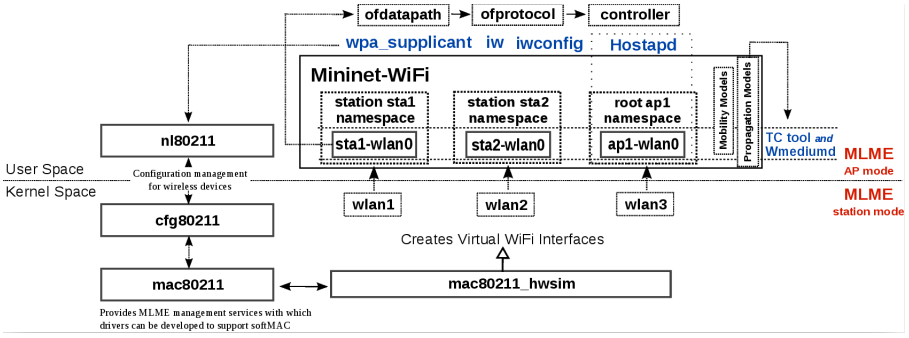


Figura II.11: Componentes de Mininet WiFi [14]

Mininet-WiFi también utiliza un par de utilidades como iw, iwconfig y/o wpa_supplicant. Los dos primeros se usan para la configuración de la interfaz y para obtener información

de las interfaces inalámbricas y el último se usa con Hostapd, para admitir WPA (acceso protegido de Wi-Fi), entre otras cosas. Además de ellos, otra utilidad fundamental es TC (Control de tráfico). El TC es un programa de utilidad de espacio de usuario utilizado para configurar el programador de paquetes de kernel de Linux, responsable de controlar la velocidad, el retraso, la latencia y la pérdida, aplicando estos atributos en las interfaces virtuales de estaciones y AP, representando con mayor fidelidad el comportamiento del mundo real [14].

Estaciones: Son dispositivos que se conectan a un punto de acceso mediante autenticación y asociación. En esta implementación, cada estación tiene una tarjeta inalámbrica. Como los hosts Mininet tradicionales están conectados a un punto de acceso, las estaciones pueden comunicarse con esos hosts [14].

Puntos de acceso: Son dispositivos que administran las estaciones asociadas. Virtualizado a través de Daemon hostapd y uso de interfaces inalámbricas virtuales para punto de acceso y servidores de autenticación. Mininet-WiFi actualmente incluye soporte para las implementaciones de referencia de espacio de usuario y Open vSwitch en los modos de kernel y espacio de usuario [14].

1.9. Miniedit

Editor de interfaz gráfica de usuario (GUI) para Mininet. Herramienta experimental para extender la funcionalidad de Mininet, es capaz de crear y ejecutar diferentes simulaciones de red, de igual manera, se configuran los diferentes elementos de red de manera simplificada y guarda las diferentes topologías. MiniEdit tiene una interfaz de usuario sencilla, presenta: un área de trabajo, herramientas de diseño; iconos de las herramientas tales como: equipos host, conmutadores OpenFlow habilitados, conmutador Ethernet de aprendizaje, router básico, netlink para crear enlaces entre los dispositivos de red y la herramienta fundamental, un controlador. En la Figura II.12 se observa la interfaz de MiniEdit para Mininet WiFi [2].

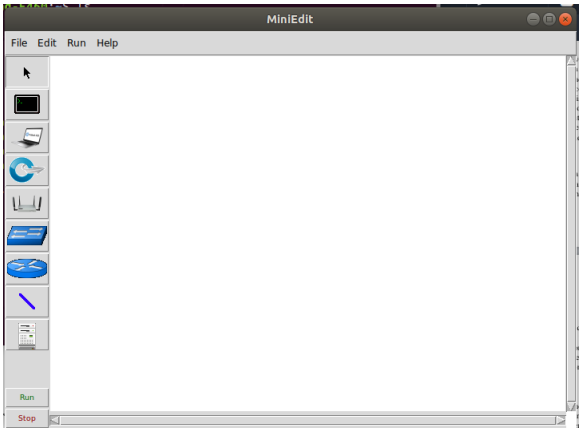


Figura II.12: Interfaz MiniEdit para Mininet Wi-fi. Fuente:Autor

Se puede observar de la Figura II.12, que en la parte superior se encuentran las opciones de trabajo para la red que se este implementando, en la parte izquierda de la ventana se

encuentran los componentes que se pueden utilizar, en la parte central se encuentra la zona de trabajo y finalmente en la parte inferior izquierda se encuentran las opciones de inicio y parada (RUN Y STOP respectivamente) de la emulación.

2. Estado del arte

Al ser una tecnología innovadora, en este momento la teoría acerca de las redes SDN/SDR es bastante escasa, sin embargo se presentará literatura con propuestas similares a las planteadas en este documento, para poder hacer un análisis critico del estado actual de la tecnología 5G.

2.1. Integración de SDR y SDN para 5G

En este trabajo se plantean los conceptos básicos que debería tener una red implementada con tecnología SDN/SDR, su rendimiento frente a sus predecesores, un posible diseño de controlador de capa cruzada y la arquitectura por capas que posiblemente tendrán las redes 5G [15], como se puede ver en la Figura II.13.

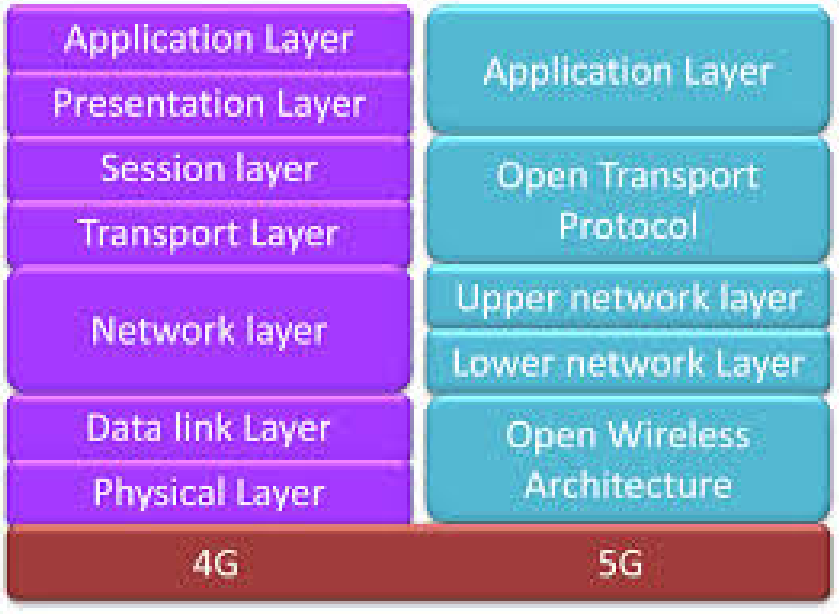


Figura II.13: Arquitectura por capas de redes 4G y redes 5G [15]

En este artículo también se da una visión general acerca de cómo se vería en la implementación una red híbrida entre SDN y SDR, como se ve en la Figura II.14

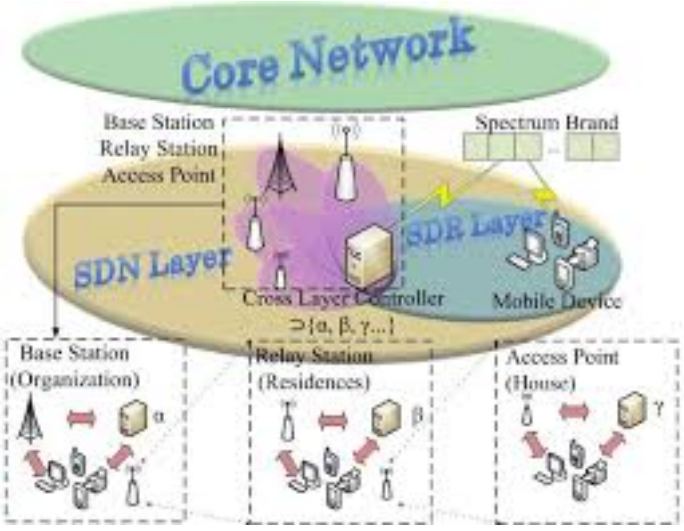


Figura II.14: Bosquejo de la arquitectura SDN/ SDR implementada [15]

Finalmente, en este artículo se presentan simulaciones con el rendimiento que podrían tener las redes SDN/SDR respecto a otras que no utilizan esta arquitectura en términos de uso de ancho de banda, latencia y tiempo de inserción, como se puede ver en las Figuras II.15, II.16 y II.17 respectivamente.

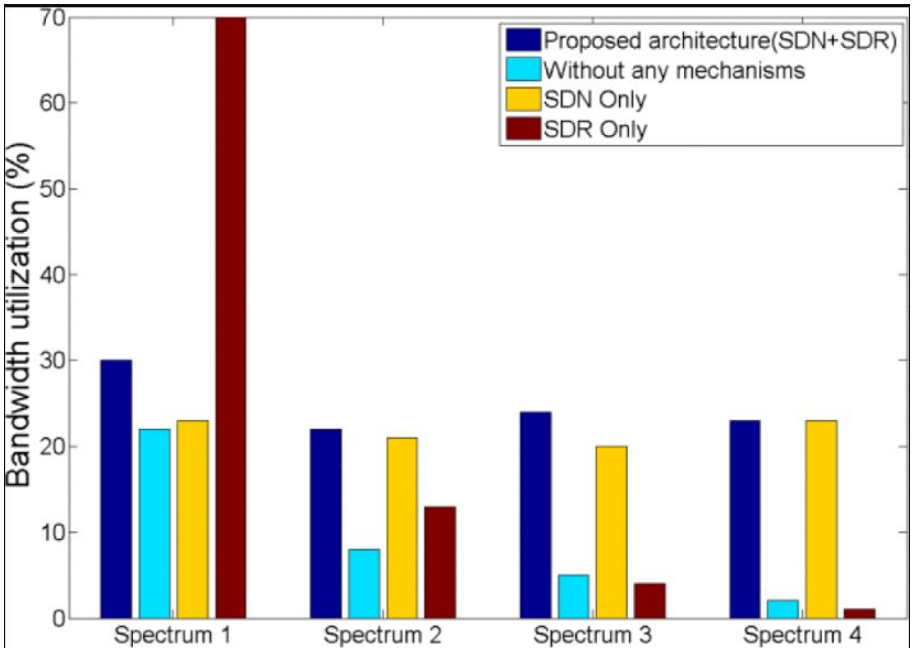


Figura II.15: Uso de ancho de banda vs espectro [15]

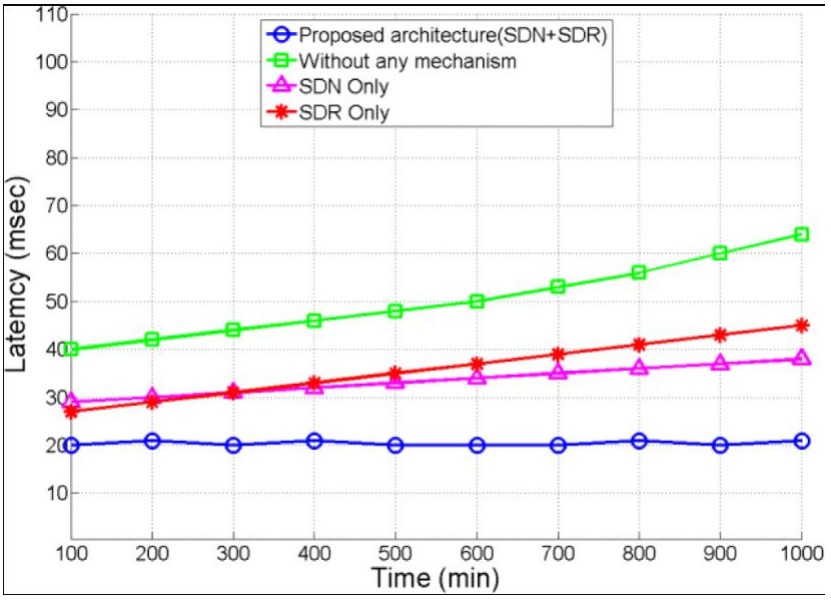


Figura II.16: Latencia vs tiempo en minutos [15]

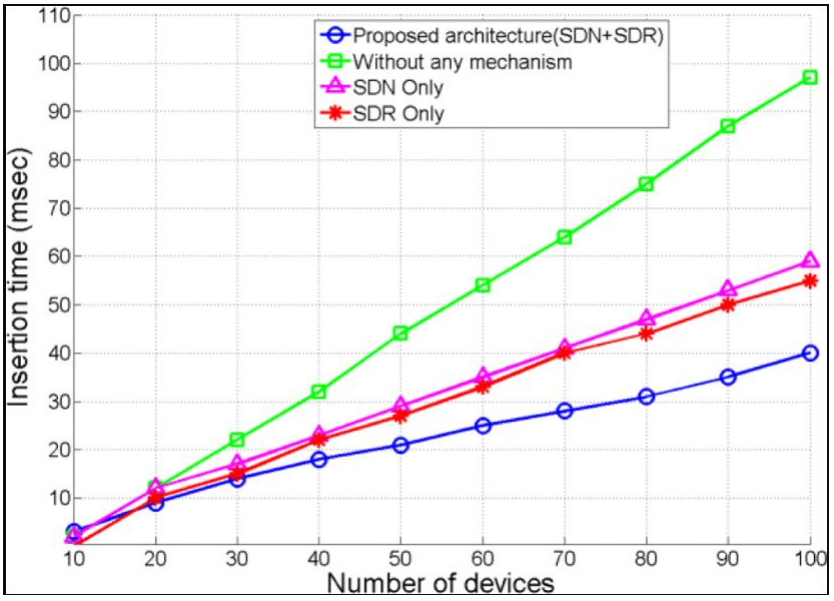


Figura II.17: Tiempo de inserción vs numero de dispositivos [15]

De lo anterior se puede decir que efectivamente una red híbrida SDN/SDR seria superior en todos los aspectos.

2.2. Asignación y programación de recursos de red para múltiples aplicaciones de control en SDN

En este artículo se propuso la idea de la asignación integrada de ancho de banda de enlace y tabla de flujo para aplicaciones de control múltiple en SDN. Lo anterior está sustentado en el conflicto existente entre la necesidad del control de flexibilidad de nivel de

flujo y la capacidad limitada de la tabla de flujo. También se utilizó el espacio de reenvío virtual (VFS), ya que VFS no solo puede implementar la asignación justa de ancho de banda de enlace y la programación de flujo de retardo mínimo en el plano de datos, sino también acelerar el reenvío de paquetes al buscar reglas de control en el plano de control.

Además se construyó un modelo de asignación conjunta basado en el precio, refiriéndose a la parte económica del recurso de red en la SDN introduciendo el precio para cada uno de los recursos, que puede obtener la asignación equitativa proporcional del ancho de banda del enlace y el retraso global mínimo al mismo tiempo. También se pretende diseñar una política popular de programación de flujos basada en la asignación equitativa proporcional del ancho de banda del enlace para lograr un retraso global mínimo.

También se evaluó el algoritmo de asignación de recursos con el tráfico real de Red de Educación e Investigación de China (CERNET) y uno del proveedor de servicios de Internet (ISP) en China, donde se pretende demostrar que el algoritmo puede garantizar la equidad de asignación de recursos de red y, además, realizar la maximización de la tasa de reenvío medio así como la minimización del tiempo de retraso medio [16]. La implementación que se llevo a cabo en este artículo se puede observar en la Figura II.18.

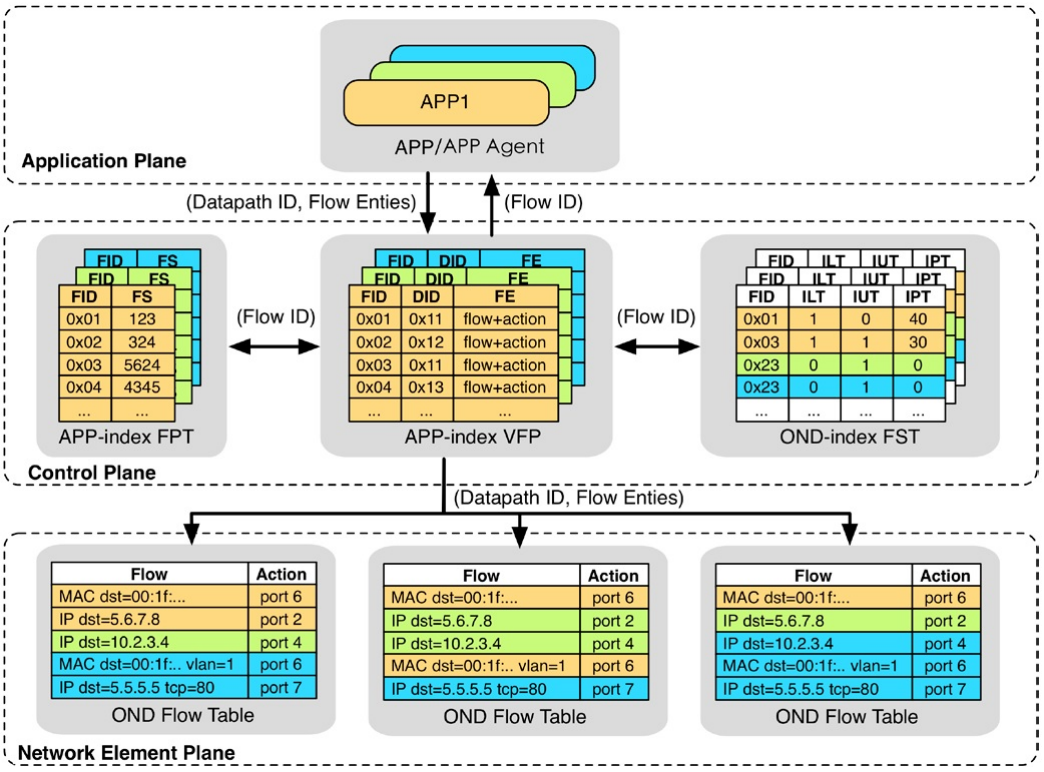


Figura II.18: Implementación y Diseño VFS [16]

Con un diseño ya implementado, se procedió a realizar pruebas de rendimiento en base a otros modelos existentes, como se puede evidenciar en las Figuras II.19 y II.20, todo se realizó tanto para CERNET como para ISP.

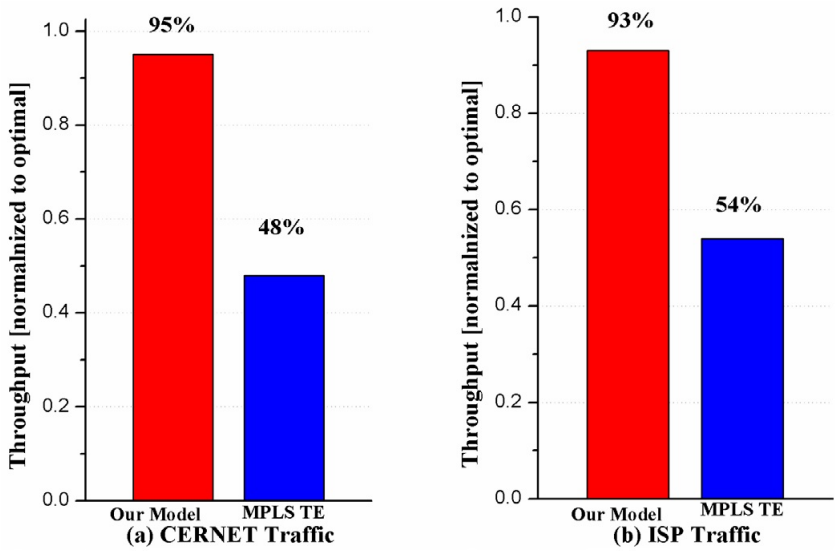


Figura II.19: Rendimiento de VFS en comparación a otro modelo existente (MPLSTE) [16]

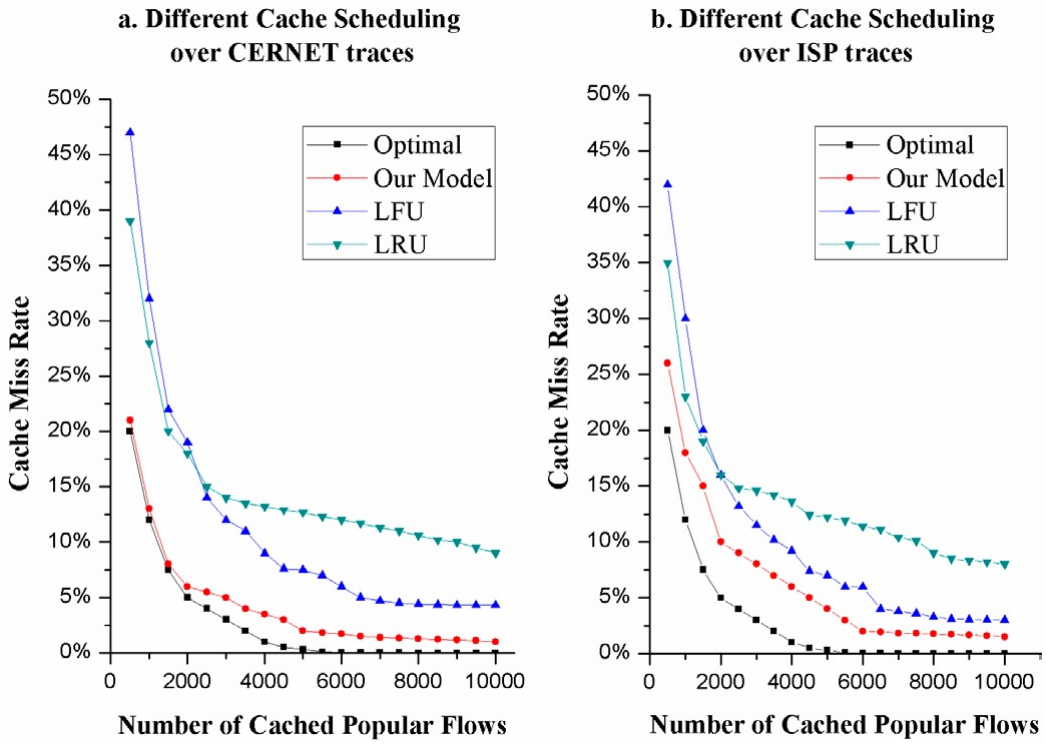


Figura II.20: Tasa de pérdida de caché en función de número de flujos populares en caché para el modelo propio, el modelo óptimo y otros modelos existentes [16]

Como se puede ver, en las Figuras II.19 y II.20, el desempeño del modelo planteado es superior a los demás, y se acerca al óptimo teórico, lo cual permite asumir que aplicaciones que contengan bases SDN mezcladas con otras tecnologías innovadoras serán superiores a las existentes actualmente.

Por último, en este trabajo se concluye que en la red IP tradicional, el ancho de banda de enlace es el recurso clave, ya que hay muchas aplicaciones que consumen ancho de banda, por ejemplo, video a la carta, como Netflix, Hulu. Sin embargo, en SDN, no solo el ancho de banda del enlace, pero también la capacidad de la tabla de flujo debe considerarse como un recurso valioso porque la capacidad de control de flujo de grano fino y específico es indispensable.

2.3. Un sistema de monitoreo de radiodifusión de FM de múltiples estaciones basado en SDR

En [17] se pretende utilizar la Radio definida por software y la OpenWebRX (que es un receptor de comunicación en línea para transmisiones analógicas) para monitorear remotamente el contenido de transmisión por radiodifusión en Indonesia. Se considera la aplicación de SDR necesaria ya que disminuye los costos totales, permite una implementación flexible y reconfigurable (a diferencia de la implementación por hardware), además que permite monitorear mas de 1 estación al tiempo (en este documento se tiene planeado utilizar el sistema 4 estaciones al mismo tiempo).

En [17] se justifica el uso de tecnología SDR en sistemas de radiodifusión con una explicación de los elementos que componen al SDR, además de la presentación de su diagrama de bloques, como se puede ver en la Figura II.21.

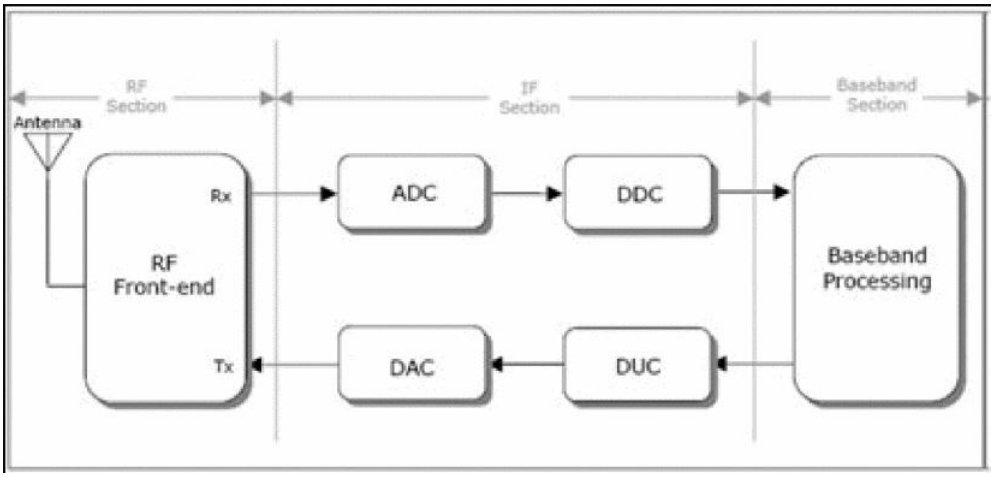


Figura II.21: Diagrama de bloques de SDR [17]

Donde según la Figura II.21 ,los elementos que componen a un SDR son:

- El front-end de Radio Frecuencia (RF) funciona como un transmisor o receptor de señales de radiofrecuencia a través de un acoplador y convierte las señales de RF en Frecuencia Intermedia (IF).
- El bloque de Conversión analógica a digital (ADC) en el lado del receptor funciona para convertir señales analógicas en señales digitales, mientras que de Conversión digital a analógica (DAC) en el lado del transmisor funciona para convertir señales

digitales en señales analógicas. ADC y DAC son interfaces entre las partes analógicas y digitales de un sistema de radio.

- El bloque de conversión digital descendente (DDC) funciona como un convertidor descendente digital en el receptor, mientras que el convertidor digital ascendente (DUC) funciona como un convertidor ascendente digital en el lado del transmisor. DDC / DUC procesa la modulación y la demodulación digital en el nivel IF, y con frecuencia también se denomina sintonización digital.

Luego de haber explicado SDR, en [17], se procede a explicar el OpenWebRX, el cual es es un receptor de comunicación en línea para transmisiones analógicas,con una interfaz de usuario web en la que está disponible la visualización en cascada en tiempo real. El software es utilizado debido a que tiene la capacidad de recibir múltiples estaciones al mismo tiempo, además que el código del programa está bajo una licencia de código abierto, por lo que se puede usar y modificar según sea necesario, en la Figura II.22 se muestra la interfaz de OpenWebRX.

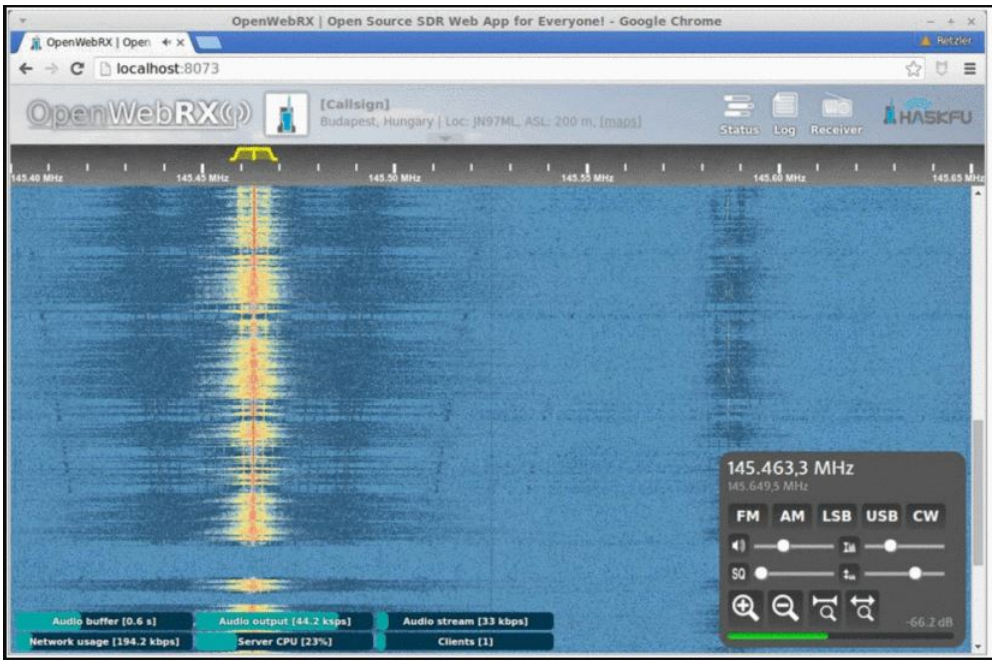


Figura II.22: Interfaz OpenWebRX [17]

Ya con los conceptos de SDR claros, en [17] se generó una propuesta de diseño para el sistema con implementación SDR y OpenWebRX , el cual se puede ver en la Figura II.23.

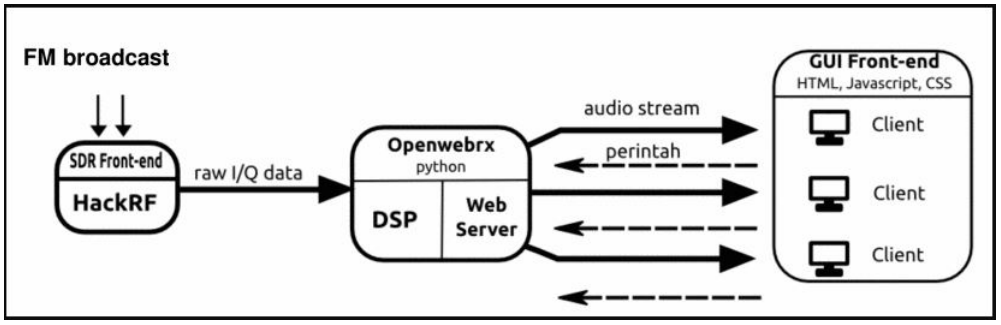


Figura II.23: Diseño sistema con implementación SDR y OpenWebRX [17]

Ya con un diseño implementado, se realizaron pruebas de rendimiento del sistema, en los cuales, se determinaron ciertos requisitos que se deben cumplir tanto en capacidad de computo, como en velocidad de internet, esto se puede ver en las Figuras II.24 y II.25 respectivamente.

#of Stations	CPU1	CPU2	CPU3	CPU4	Total CPU	Total CPU (GHz)	RAM (MB)
0	13.30%	13.30%	13.10%	15.50%	13.80%	0.45	563.2
1	34.30%	34.00%	30.90%	31.60%	32.70%	1.08	582.3
2	50.50%	47.50%	50.50%	48.50%	49.25%	1.63	591.5
3	57.70%	58.60%	58.80%	58.60%	58.43%	1.93	597.1
4	68.70%	71.10%	67.70%	70.40%	69.48%	2.29	601.9
5	77.20%	80.20%	80.80%	78.40%	79.15%	2.61	606.7
6	94.90%	95.10%	95.90%	94.80%	95.18%	3.14	617.8
7	99.00%	95.10%	99.00%	98.00%	97.78%	3.23	658.7
8	97.00%	100.00%	93.00%	99.00%	97.25%	3.21	681.5

Figura II.24: Capacidad de cómputo [17]

Minutes	Data received (MB)	Increase Data Received (MB)	Data Sent (MB)	Increase Data Sent (MB)
0	0.5215		0.0373	
5	2.1	1.5785	10.6	10.5627
10	3.6	1.5	19.0	8.4
15	5.2	1.6	27.7	8.7
20	6.7	1.5	36.8	9.1
25	8.2	1.5	45.2	8.4
30	9.8	1.6	53.8	8.6
Average		1.55		8.96

Figura II.25: Velocidad de internet [17]

Para finalizar en [17] se concluye que sobre la base de las pruebas y el análisis de los resultados de la implementación, el sistema se puede utilizar para recibir transmisiones

de FM en las frecuencias 87.7 - 107.7 MHz que utilizan la tecnología SDR. El sistema se puede utilizar para recibir hasta cuatro (4) emisoras de FM al mismo tiempo. El sistema se puede utilizar para grabar audio desde una emisora de FM. Los clientes pueden acceder al sistema a través de los últimos navegadores web Mozilla Firefox y Google Chrome utilizando una conexión de internet remota.

2.4. Un marco basado en SDN / NFV para la gestión y el despliegue de una red central 5G basada en servicios

En [18] el objetivo principal es el de proponer una arquitectura de administración novedosa para una red central basada en servicios 5G basada en virtualización de funciones de red (NFV) y SDN. Además , se plantean problemas con el tráfico de datos de las redes actuales, la creciente demanda de servicios además de los costos que acarrea toda la implementación del hardware que compone a una red, con todo esto, en la Figura II.26 dan a conocer las ventajas de una implementación SDN/NFV.

	Technology adoption	Customized network slicing	Flexibility	Scalability	Backhaul cost
Softcell [14]	SDN	✓	Low	High	High
MobileFlow [16]	SDN	-	Low	Low	High
Softepc [17]	NFV	-	Low	Low	High
Softair [19]	SDN+NFV	✓	Low	High	Low
Softnet [20]	SDN+NFV	-	Low	High	High
Proposed framework	SDN+NFV	✓	High	High	Low

Figura II.26: Comparación entre el modelo SDN/NFV y otros modelos existentes [18]

También se trata en [18] el cómo se podría implementar la arquitectura SDN/NFV, desde las capas que se utilizarían hasta los servicios que podría proveer, además de que ilustran todo en la Figura II.27.

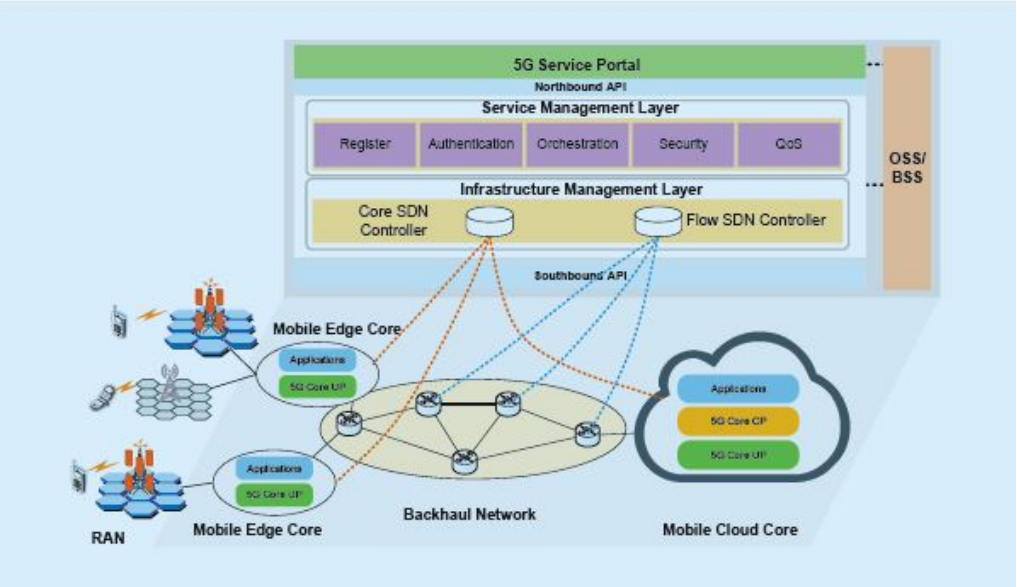


Figura II.27: Arquitectura SDN/NFV [18]

En [18] también se trata lo que es la arquitectura de corte de red, planteando un modelo que garantizará un buen servicio, como se muestra en la Figura II.28, también es presentado un desarrollo matemático con el cual se evalúan parámetros de redes.

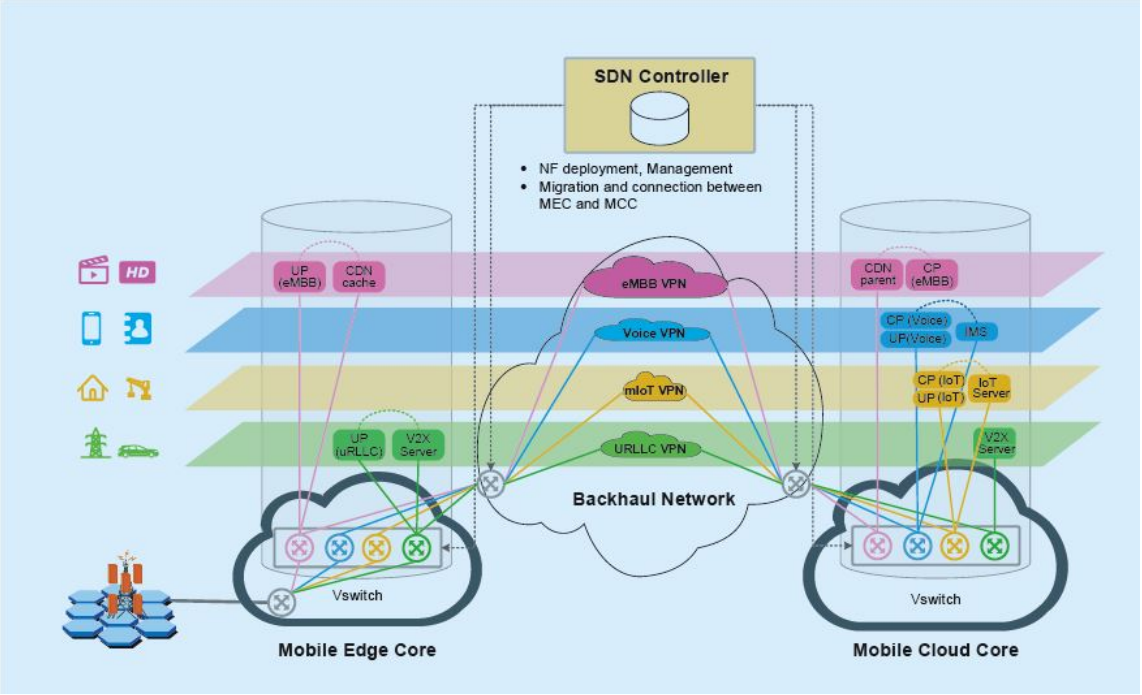


Figura II.28: Arquitectura de corte de red [18]

Por último se concluye con varias simulaciones evaluando el comportamiento del modelo SDN/NFV, entre ellas los costos en función del precio de la energía de servidores de borde, y los costos totales en función del tamaño de la solicitud de la red, esto se evidencia en la

Figuras II.29 y II.30 respectivamente.

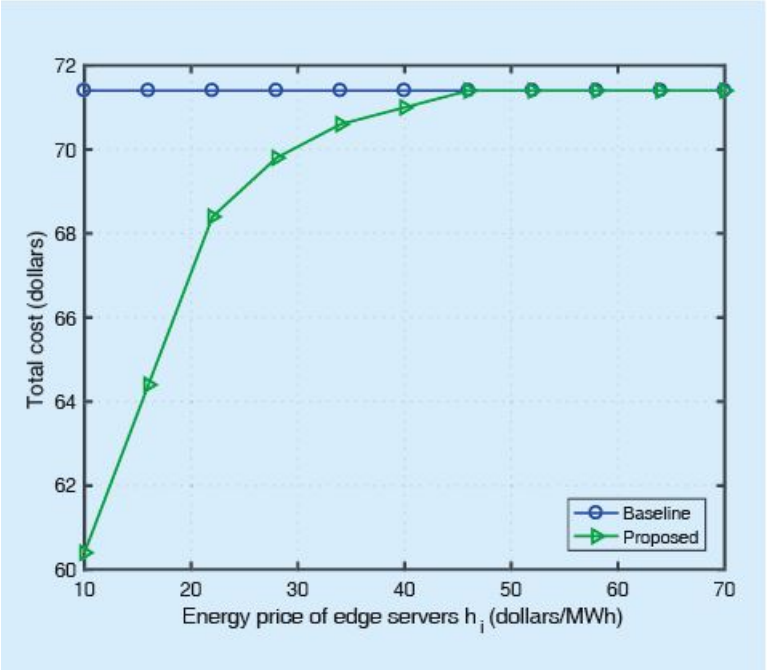


Figura II.29: Costos en función del precio de la energía de servidores de borde [18]

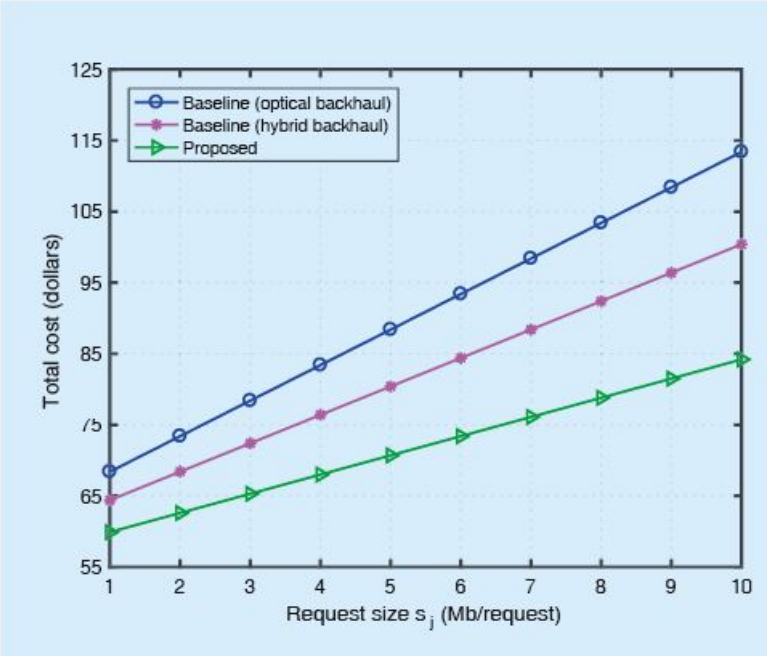


Figura II.30: Costos totales en función del tamaño de la solicitud de la red [18]

Donde, para ambos casos en [18], la curva representativa del modelo SDN/NFV es la de color verde, así que se puede evidenciar que aplicaciones que contengan tecnología SDN combinada con otro tipo de tecnología (en este caso NFV) tendrán un rendimiento superior a otros tipos de tecnología.

2.5. CrossFlow: una arquitectura de capas cruzadas para SDR utilizando los principios de SDN

En [19] se tiene como objetivo presentar un enfoque basado en principios para construir una arquitectura de capas cruzadas utilizando los componentes básicos de SDR y los principios de redes definidas por software (SDN), esto es debido a que se podría tener un modelo que admita aplicaciones de varios protocolos.

La arquitectura "Crossflow" que se propone en este artículo estaría diseñada de tal forma que sea capaz de cumplir con las siguientes aplicaciones:

- Adaptación de la capa física que incluye
  - Salto de frecuencia para gestionar de manera eficiente la conmutación entre canales para la utilización efectiva del espectro y evitar la interceptación no autorizada.
  - Control de potencia de transmisión para compensar eficientemente la confiabilidad de la comunicación y la interferencia causada a otros usuarios.
  - La modulación y codificación adaptativas para el intercambio y la confiabilidad de la comunicación, y se adaptan a las variaciones del canal.
- El aprovisionamiento de calidad de servicio (QoS) que permite la especialización de las políticas de QoS a través del control de acceso medio, la regulación, el control de admisión, la programación y el control de errores (por ejemplo, ARQ y FEC).
- Enrutamiento adaptable para permitir que un controlador CrossFlow distribuido, con su visión global de la red, cambie dinámicamente entre protocolos de enrutamiento reactivo y proactivo según las condiciones de la red y las restricciones de la aplicación.
- Control entre capas para permitir la optimización conjunta de parámetros, algoritmos y protocolos en todas las capas de la pila de protocolos.

El siguiente paso tomado en [19] fue el de explicar el software que se iba a usar en la construcción de un controlador de capa cruzada, en este caso seria GNU Radio. Para finalmente presentar un diagrama con el diseño de un controlador de capa cruzada, como se puede ver en la Figura II.31.

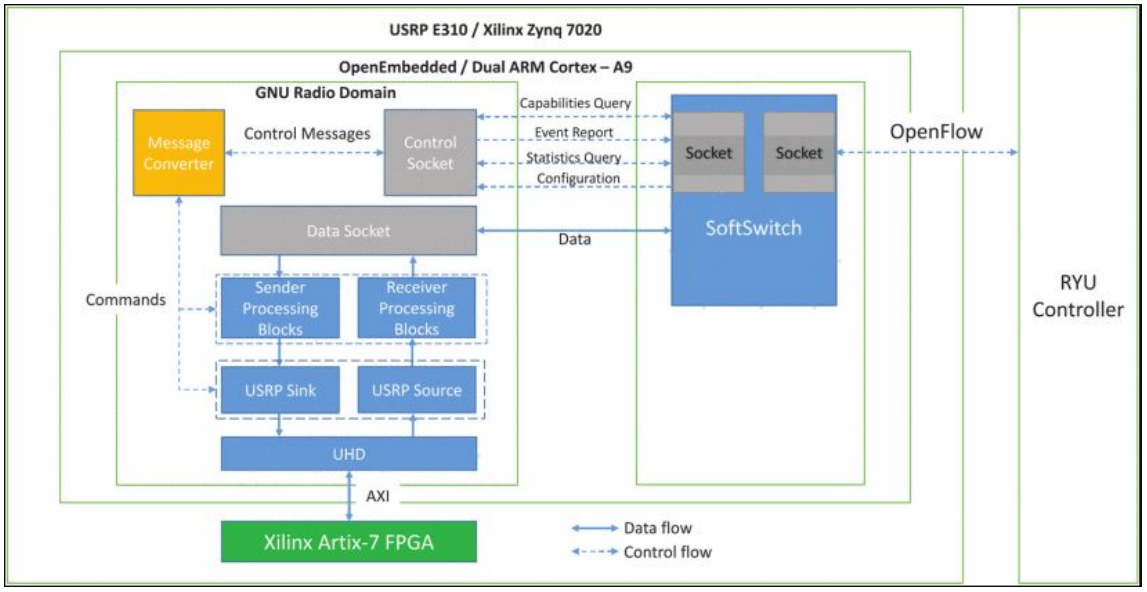


Figura II.31: Diseño controlador capa cruzada [19]

Donde:
Se utilizó USRP 310 de Ettus Research como la plataforma de hardware para el modelo. E310 usa Zynq 7020, que tiene un procesador ARM Cortex A9 de doble núcleo y FPGA en el mismo chip. Con una importante potencia de procesamiento presente, se planeó cargar tanto los bloques de GNU Radio como el plano de datos del software CPqD SoftSwitch en el dispositivo. El SoftSwitch que en pocas palabras es el dispositivo que separa los elementos de la red (Hardware) del control de la misma (Software), actuará como un agente de OpenFlow con extensiones personalizadas. El SoftSwitch también actuará como interfaz entre los bloques de GNU Radio y el controlador Ryu SDN. Para establecer la comunicación entre ofsoftswitch y los bloques de radio GNU, se abrieron sockets en ambos dominios para crear un canal de IPC. El dominio de GNU Radio tendrá una implementación de bloque de mensajes personalizada, que traducirá el mensaje de ofsoftswitch a un formato de mensaje comprendido por el bloque. Para los bloques USRP Source y USRP Sink, que obtienen y envían muestras desde y hacia el USRP Hardware Driver (UHD), respectivamente, el formato del mensaje es Polymorphic Types (PMT).

2.6. Integración de la virtualización de la función de red con SDR y SDN para redes 4G / 5G

En este documento se plantea como problema inicial que las redes móviles 4G no serán capaces de suplir las necesidades previstas para 2020, dado que la eficiencia del enlace inalámbrico se está acercando a los límites fundamentales [20].

Se plantea la búsqueda de nuevos paradigmas para afrontar dichos problemas, uno de ellos la integración de NFV (virtualización de la función de red), con la radio definida por software(SDR) y las redes definidas por software (SDN)

En este documento se profundiza en el concepto de NFV, SDR y SDN, analizando la estandarización que aplica para estas técnicas.

En este documento plantean una arquitectura híbrida de un sistema en el cual se integra NFV, SDN y SDR, como lo podemos ver en la Figura II.32.

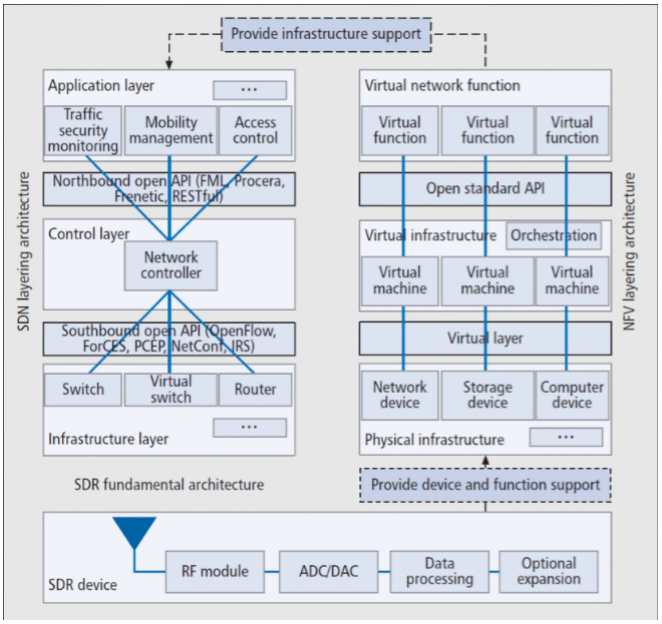


Figura II.32: Arquitectura de integración SDN SDR NFV [20]

Cada una de estas tecnologías aportan ventajas, por ejemplo: SDR supera las desventajas del hardware y ayuda a dividir la funcionalidad de control y la funcionalidad de servicio. Los dispositivos SDR brindan soporte de dispositivo y función a la arquitectura NFV, mientras que NFV brinda soporte de infraestructura a SDN.

Para llegar a cubrir la demanda planteada para las redes 5G, la red de comunicación inalámbrica debe ser capaz de satisfacer enormes cantidades de demandas de servicios diversificados de diferentes velocidades de datos de los usuarios. Como solución, el concepto de redes heterogéneas inteligentes (HetNets) se propone como un esquema jerárquico de utilización de recursos de red que explota las tecnologías tanto celulares como WLAN. Los HetNets en la naturaleza requieren el soporte de las tecnologías NFV, SDN y SDR en diferentes niveles, en la Figura II.33.

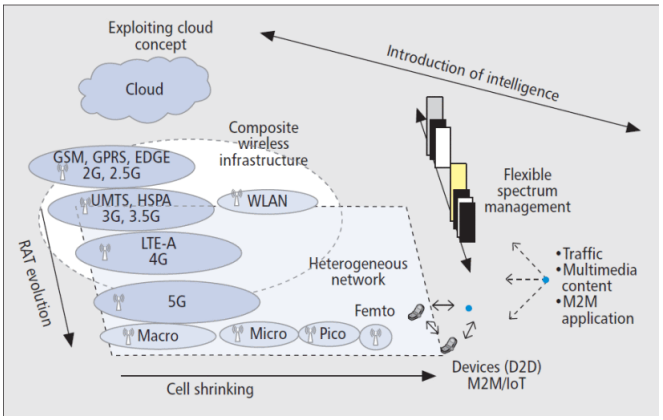


Figura II.33: Evolución en las redes móviles [20]

Cada una de las tecnologías cumplen funciones bien definidas en la integración, SDR es responsable de los problemas de la capa física, como la modulación, el procesamiento de señales, la codificación de canales; y encapsulación de datos. SDN se centra en la programación de elementos de red para garantizar el transporte de extremo a extremo, así como en la conmutación y enrutamiento de paquetes de datos con QoS garantizada. NFV organiza y combina las entidades funcionales de la red para que el servicio y el dispositivo se puedan implementar rápidamente.

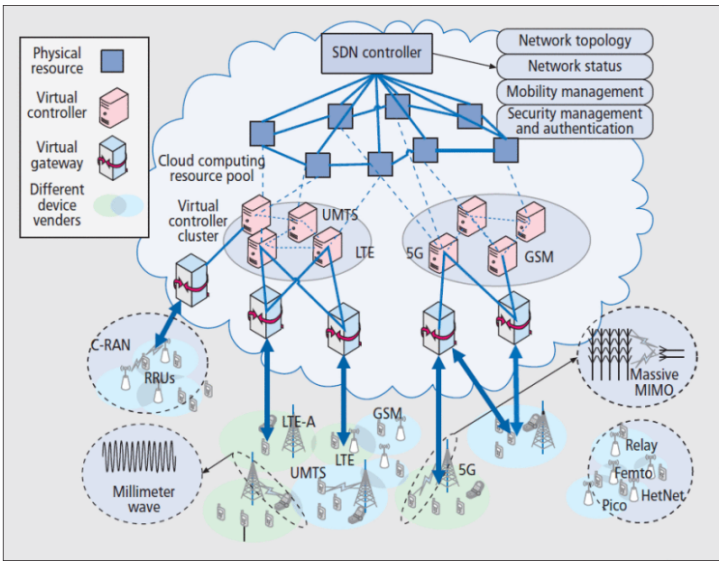


Figura II.34: Integración de tecnologías [20]

En la figura II.34 se puede ver la proyección de cómo serán las redes móviles en un futuro no muy lejano, con la aplicación de algunas tecnologías aún en experimentación, como la implementación de ondas milimétricas o MIMO masivo.

Capítulo III

Implementación del controlador y configuración del entorno de prueba

1. Instalación de complementos necesarios

En esta sección del documento se evidenciará todo el proceso realizado para cumplir con los objetivos planteados anteriormente.

Lo primero que se mostrará en esta sección sera las temáticas que competen a las fases 1 y 2 del trabajo, dentro de este contenido se podrá evidenciar la instalación del controlador SDN Opendaylight(ODL) y el uso de su interfaz gráfica DLUX, la instalación de Mininet, Mininet-WiFi, acceso a la interfaz gráfica Miniedit y posteriormente el uso de Mininet-WiFi mediante scrypts de python.

1.1. Instalación Opendaylight(ODL)

El controlador ODL puede instalarse de dos maneras diferentes: clonando e instalando los repositorios desde Github o descargando directamente el archivo que contiene el controlador, descomprimirlo e instalarlo. En este trabajo se utilizó la instalación mediante la descarga del archivo. En la página oficial de ODL, en la cual hay una opción de "descargas" en la sección para desarrolladores, tal como se resalta en el óvalo de color rojo de la Figura III.1.

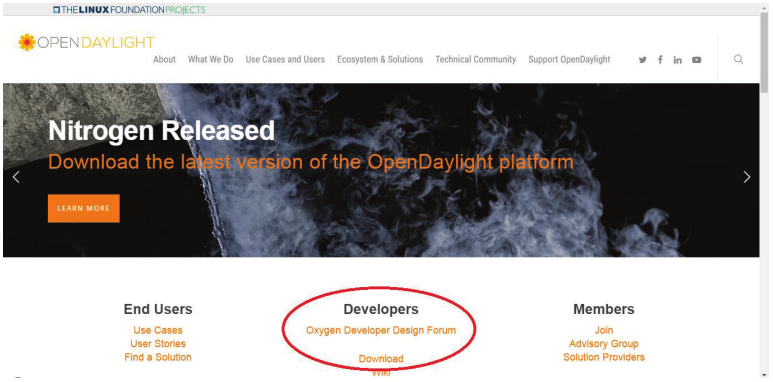


Figura III.1: Página oficial de Opendaylight. Fuente: Autor.

El siguiente paso en la instalación, es el de escoger la distribución deseada de Opendaylight, para este caso se escogió la distribución 0.4.0-Beryllium/ ya que es la recomendada por el manual de la página de ODL. Esto se puede ver en la Figura III.2.

Name	Last Modified	Size	Description
Parent Directory			
0.2.0-Helium/	Fri Dec 19 15:21:23 UTC 2014		
0.2.1-Helium-SR1.1/	Thu Jan 29 23:34:59 UTC 2015		
0.2.1-Helium-SR1/	Thu Jan 29 23:34:59 UTC 2015		
0.2.2-Helium-SR2/	Thu Jan 29 23:34:59 UTC 2015		
0.2.3-Helium-SR3/	Tue Aug 11 02:31:49 UTC 2015		
0.2.4-Helium-SR4/	Tue Aug 11 02:31:49 UTC 2015		
0.3.0-Lithium/	Tue Aug 11 02:31:49 UTC 2015		
0.3.1-Lithium-SR1/	Thu Oct 15 15:20:13 UTC 2015		
0.3.2-Lithium-SR2/	Thu Oct 15 15:20:13 UTC 2015		
0.3.3-Lithium-SR3/	Thu Dec 03 18:58:02 UTC 2015		
0.3.4-Lithium-SR4/	Thu Mar 03 23:00:22 UTC 2016		
0.4.0-Beryllium/	Thu Feb 18 18:37:48 UTC 2016		
0.4.1-Beryllium-SR1/	Wed Mar 23 02:20:31 UTC 2016		
0.4.2-Beryllium-SR2/	Thu May 12 09:56:33 UTC 2016		
0.4.3-Beryllium-SR3/	Thu Aug 04 20:07:11 UTC 2016		
0.4.4-Beryllium-SR4/	Wed Nov 02 12:25:23 UTC 2016		

Figura III.2: Distribuciones disponibles de ODL. Fuente: Autor.

Lo siguiente a realizar será seleccionar el archivo que contiene el controlador deseado, en este caso es el archivo distribution-karaf-0.4.0-Beryllium.tar.gz .

Name	Last Modified	Size
Parent Directory		
distribution-karaf-0.4.0-Beryllium.pom	Tue Feb 16 12:13:38 UTC 2016	14477
distribution-karaf-0.4.0-Beryllium.pom.asc	Tue Feb 16 13:26:29 UTC 2016	473
distribution-karaf-0.4.0-Beryllium.pom.asc.md5	Wed Feb 17 10:14:06 UTC 2016	32
distribution-karaf-0.4.0-Beryllium.pom.asc.sha1	Wed Feb 17 10:14:06 UTC 2016	40
distribution-karaf-0.4.0-Beryllium.pom.md5	Wed Feb 17 10:14:06 UTC 2016	32
distribution-karaf-0.4.0-Beryllium.pom.sha1	Wed Feb 17 10:14:06 UTC 2016	40
distribution-karaf-0.4.0-Beryllium.tar.gz	Tue Feb 16 12:13:52 UTC 2016	293750321
distribution-karaf-0.4.0-Beryllium.tar.gz.asc	Tue Feb 16 13:26:29 UTC 2016	473
distribution-karaf-0.4.0-Beryllium.tar.gz.asc.md5	Wed Feb 17 10:14:06 UTC 2016	32
distribution-karaf-0.4.0-Beryllium.tar.gz.asc.sha1	Wed Feb 17 10:14:06 UTC 2016	40
distribution-karaf-0.4.0-Beryllium.tar.gz.md5	Wed Feb 17 10:14:06 UTC 2016	32
distribution-karaf-0.4.0-Beryllium.tar.gz.sha1	Wed Feb 17 10:14:06 UTC 2016	40
distribution-karaf-0.4.0-Beryllium.zip	Tue Feb 16 12:14:02 UTC 2016	295610182
distribution-karaf-0.4.0-Beryllium.zip.asc	Tue Feb 16 13:26:29 UTC 2016	473
distribution-karaf-0.4.0-Beryllium.zip.asc.md5	Wed Feb 17 10:14:06 UTC 2016	32
distribution-karaf-0.4.0-Beryllium.zip.asc.sha1	Wed Feb 17 10:14:06 UTC 2016	40
distribution-karaf-0.4.0-Beryllium.zip.md5	Wed Feb 17 10:14:06 UTC 2016	32
distribution-karaf-0.4.0-Beryllium.zip.sha1	Wed Feb 17 10:14:06 UTC 2016	40

Figura III.3: Distribuciones disponibles para la versión 0.4.0-Beryllium de ODL. Fuente: Autor.

Antes de continuar, es necesario verificar la versión de java con la que cuenta el computador, para la distribución de ODL utilizada se requiere por lo menos una versión de java 1.8.0, como se puede ver en la Figura III.4, esta es la versión utilizada en este trabajo.

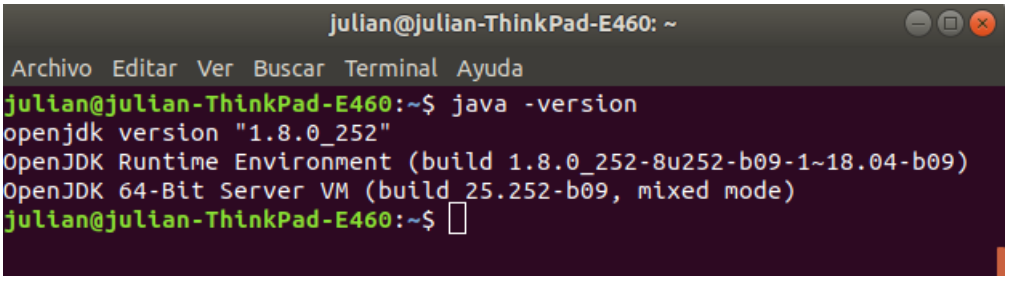


Figura III.4: Versión Java. Fuente: Autor.

Continuando con la instalación de ODL, se procede a descomprimir el archivo descargado, esto se realiza con el siguiente comando:

```
tar zxvf distribution-karaf-0.4.0-Beryllium.tar.gz
```

El último paso a realizar en la instalación de ODL se puede observar en los siguientes comandos, con los cuales se accede a la carpeta donde se encuentra el controlador y se ejecuta el mismo respectivamente.

```
cd distribution-karaf-0.4.0-Beryllium
sudo ./bin/karaf
```

Si se realizó la instalación correctamente, al iniciar el controlador debe aparecer un aviso similar al mostrado en la Figura III.5.

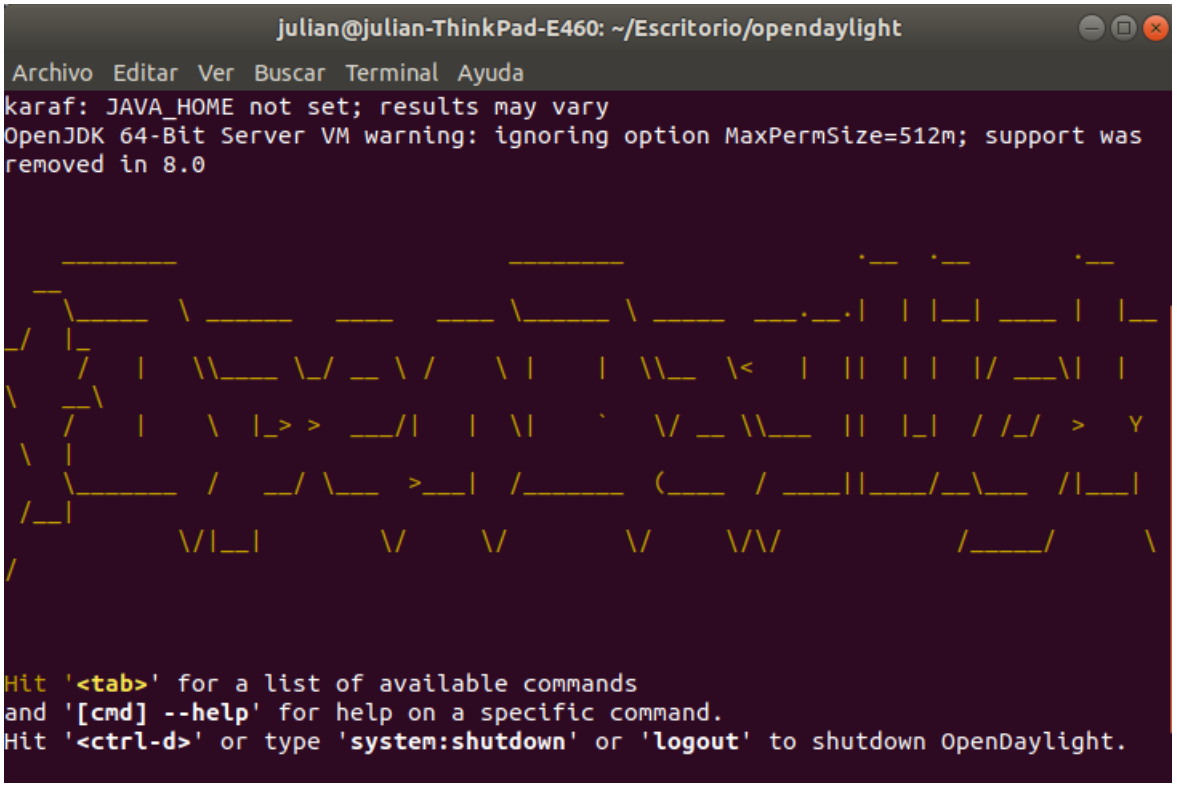


Figura III.5: Inicialización de l controlador Opendaylight. Fuente: Autor.

Además de tener el controlador funcionando o inicializado, es necesario instalarle ciertos complementos que permiten la conexión entre ODL y Mininet, además de la interfaz gráfica de ODL, este comando se puede ver en la Figura III.6.

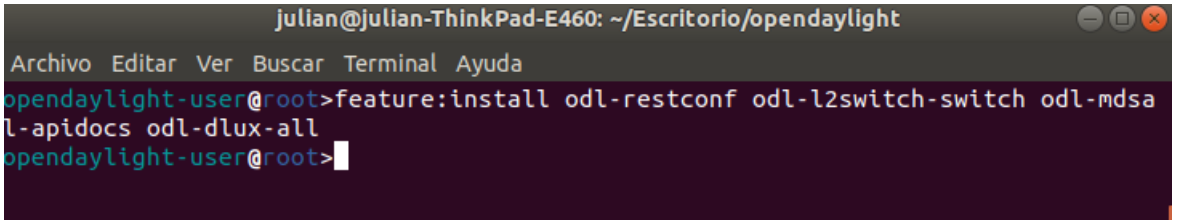


Figura III.6: Instalación de complementos para ODL. Fuente: Autor.

Los complementos instalados fueron los siguientes:

- odl-restconf: permite el acceso a RESTCONF API
- odl l2switch-switch: provee funcionalidades de red similares a un switch Ethernet.
- odl-mdsal-apidocs: permite el acceso a la Yang API.
- odl-dlux-all: la interfaz de usuario gráfica de OpenDayLight.

1.2. Instalación Mininet

Para la instalación de Mininet se siguió la guía ofrecida por la página oficial de Mininet, la cual consta de los siguientes pasos:

Entrar a la página oficial de Mininet, como se muestra en la Figura III.7

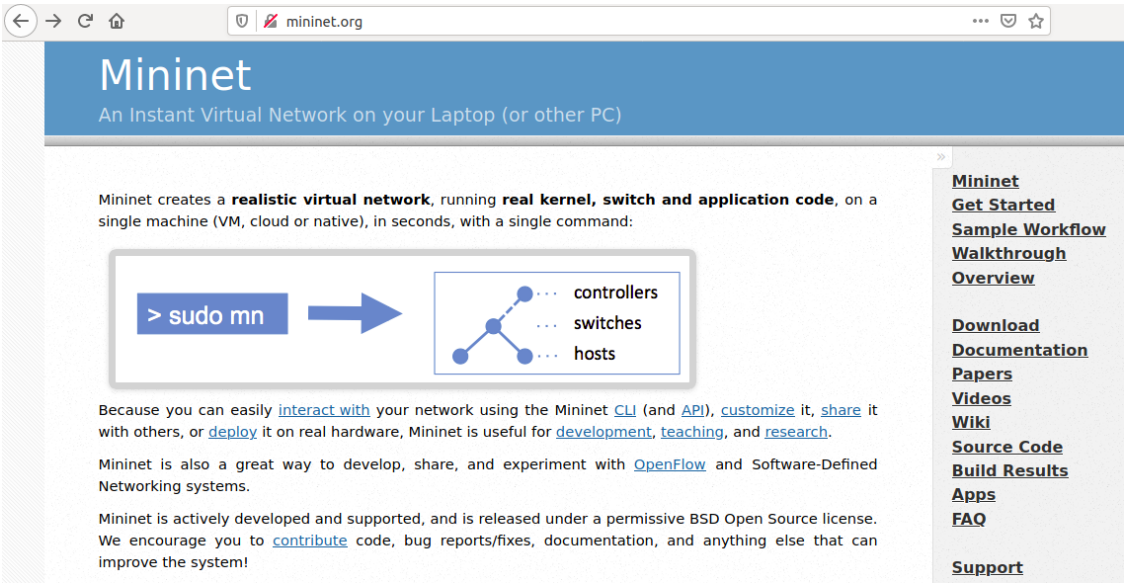


Figura III.7: Página oficial de Mininet.Fuente: Autor.

El siguiente paso es ir a la sección de descargas (Download), esto se puede ver en la Figura III.8, donde mininet ofrece 4 diferentes tipos de descargas, en este caso se uso la opción 2 ya que no se pretende trabajar en máquina virtual sino en un sistema nativo Ubuntu.

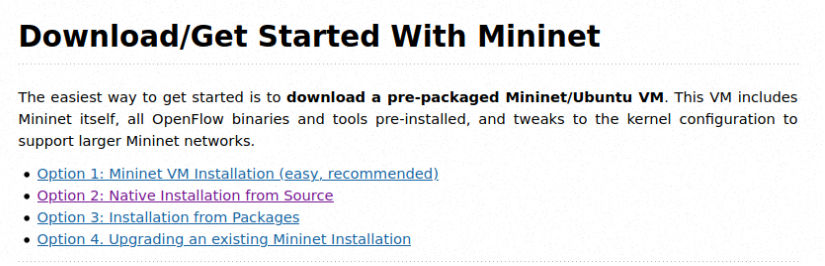


Figura III.8: Opciones de descarga de Mininet. Fuente: Autor.

Los siguientes pasos serán sobre la ventana de comandos, como se puede evidenciar en el siguiente comando.

git clone git://github.com/mininet/mininet

El anterior comando sirve para instalar de forma nativa la última versión de mininet disponible, pero si se desea especificar la versión, es necesario aplicar los siguientes comandos, presentados en la Figura III.9.

```
cd mininet
git tag  # list available versions
git checkout -b 2.2.1 2.2.1  # or whatever version you wish to install
cd ..
```

Figura III.9: Comandos para la especificación de la versión de mininet a instalar. Fuente: Autor.

En los anteriores comandos respectivamente se va a la carpeta que contiene a mininet, luego se abre una lista de opciones con la versión del mismo, luego se escoge la versión deseada (en este caso 2.2.1) y finalmente se sale de la carpeta de mininet.

Ya con la versión deseada de mininet a instalar, se procede a instalar los complementos del mismo, para este caso se instalaron todos los complementos disponibles para este software para evitar futuros problemas y además por simplicidad, el comando utilizado fue el siguiente.

mininet/util/install.sh -a

Ya teniendo todo instalado se procede a realizar una prueba con el software pare verificar que todo haya quedado bien instalado, para esta prueba se crea una topología simple y por defecto que trae Mininet, con el siguiente comando.

sudo mn –test pingall

Si todo quedó bien instalado, debería abrirse mininet y crear una topología simple con 2 host conectados a 1 switch y se habrá realizado una prueba enviando paquetes de un host a otro de manera exitosa, como se muestra en la Figura III.10

```
julian@julian-ThinkPad-E460:~/Escritorio/PythonTesis$ sudo mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

Figura III.10: Prueba final de la instalación de Mininet. Fuente: Autor.

1.3. Instalación de Mininet-WiFi

El proceso para la instalación de Mininet-WiFi es bastante sencilla, ya que este es solo una extensión de mininet, solo que aplicando redes WiFi como su nombre lo indica, pero para este proyecto es fundamental y de bastante utilidad contar con esta herramienta.

Lo primero a realizar será ingresar en el siguiente comando el cual descarga la información requerida para instalar Mininet-WiFi, y además crea la carpeta donde se albergará el software.

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

Lo siguiente a realizar será buscar la carpeta donde se encuentra el software, con el siguiente comando.

```
cd mininet-wifi
```

Por último, de manera análoga a como se hizo con Mininet, se instalan los complementos de Mininet-WiFi, para este caso se instalaron todos los complementos ya que es muy posible que todos sean necesarios en algún punto del proyecto. El comando utilizado para la instalación fue el siguiente.

```
sudo util/install.sh -a
```

Teniendo ya todo instalado, lo siguiente que se debe realizar es, comprobar el correcto funcionamiento de Mininet-WiFi , y para esto se usa la topología por defecto del software, en donde se crean dos estaciones móviles (que para este trabajo se tomarán como las SDR a controlar ya que cumplen con la definición antes propuesta de la misma) conectadas vía WiFi a un punto de acceso o Access Point y se realiza un ping entre estaciones, como se puede ver en la Figura III.11 donde se crea la red, y en la Figura III.12 donde se comprueban las conexiones con un ping entre estaciones.

```
julian@julian-ThinkPad-E460:~/mininet-wifi$ sudo mn --wifi
*** Adding stations:
sta1 sta2
*** Adding access points:
ap1
*** Configuring wifi nodes...
*** Creating network
*** Adding controller
*** Adding hosts:

*** Adding switches:

*** Adding links:
(sta1, ap1) (sta2, ap1)
*** Starting controller(s)
c0
*** Starting L2 nodes
ap1 ...
*** Starting CLI:
mininet-wifi>
```

Figura III.11: Inicialización de la red por defecto de Mininet-WiFi. Fuente: Autor.

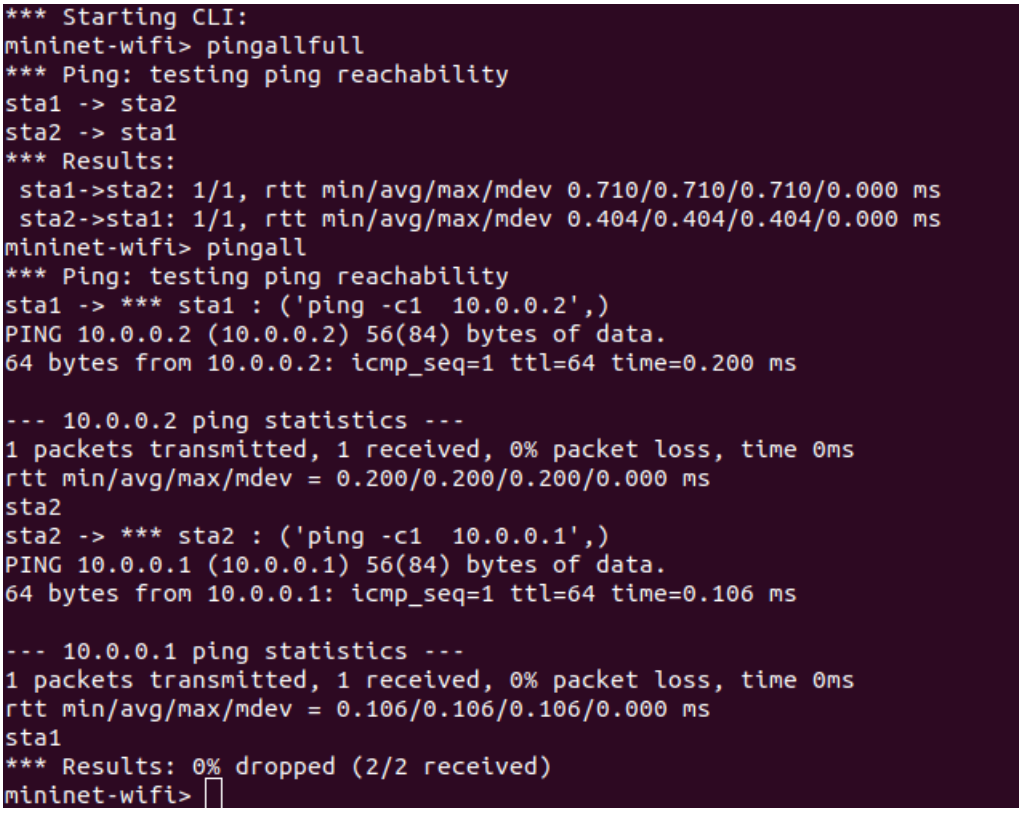


Figura III.12: Prueba de la red por defecto de Mininet-WiFi. Fuente: Autor.

Como se puede observar, todo quedó bien instalado ya que ningún paquete es descartado al realizar la prueba de conexión entre estaciones.

1.4. Interfaz Gráfica Miniedit

En esta parte del proyecto se explicará un poco el manejo de la interfaz gráfica utilizada en Mininet-WiFi y varias herramientas de configuración utilizadas, lo primero es mostrar cómo se ve una red inalámbrica utilizando directamente la interfaz gráfica Miniedit, esto se muestra en la Figura III.13, donde los símbolos que parecen computadores son las estaciones móviles, el símbolo de la mitad es el Access Point y el símbolo que está en la parte superior derecha es el controlador c0.

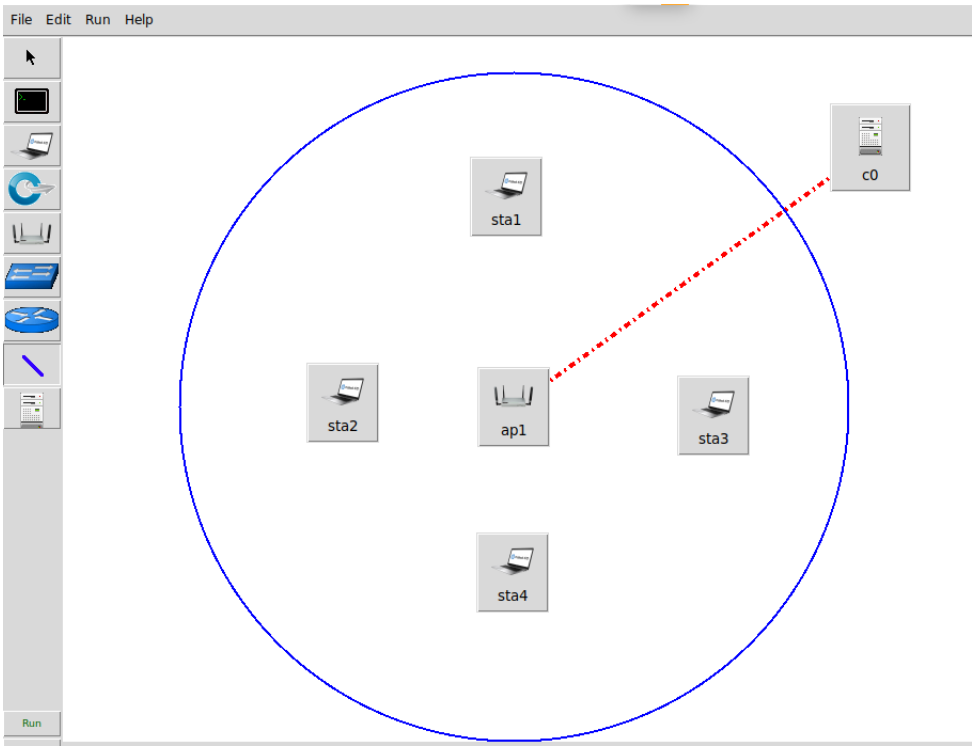


Figura III.13: Ejemplo de Topología utilizando Miniedit. Fuente: Autor.

Lo primero a configurar serán las preferencias generales de la red, en la parte superior izquierda de la Figura III.13 donde dice **Edit** es posible realizar dichos cambios, como se ilustra en la Figura III.14, donde es posible variar las direcciones de protocolo IP a utilizar, la versión del protocolo OpenFlow, el tipo de switch a utilizar, etc.

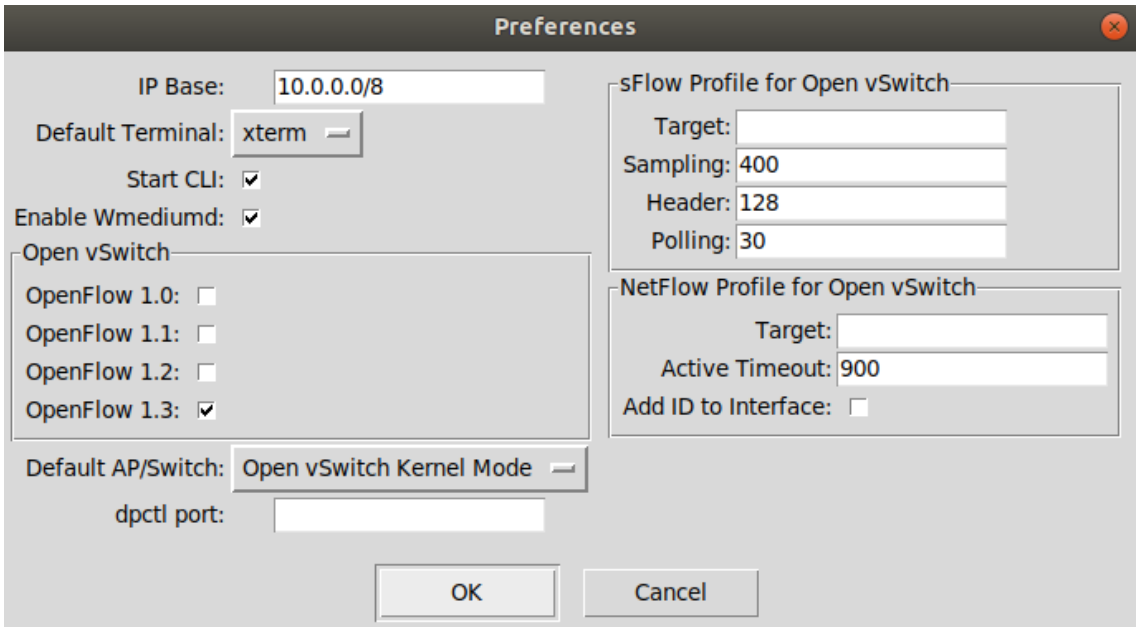


Figura III.14: Preferencias de la red en Miniedit. Fuente: Autor.

Teniendo configuradas las opciones generales de la red, se procede a configurar el controlador SDN, y esto se ilustra en la Figura III.15, donde se puede configurar desde el puerto del controlador hasta su dirección IP, se debe agregar que para una instalación del controlador en un sistema nativo Ubuntu, este queda almacenado en el localhost del ordenador, por lo que la IP asignada por defecto es 127.0.0.1, mientras que para la configuración del puerto es necesario tener en cuenta el controlador utilizado, para este caso es ODL por lo que el puerto a utilizar puede ser el 6633 o el 6653, por último, si se quiere vincular directamente el controlador con mininet es necesario que el tipo de controlador sea remoto.

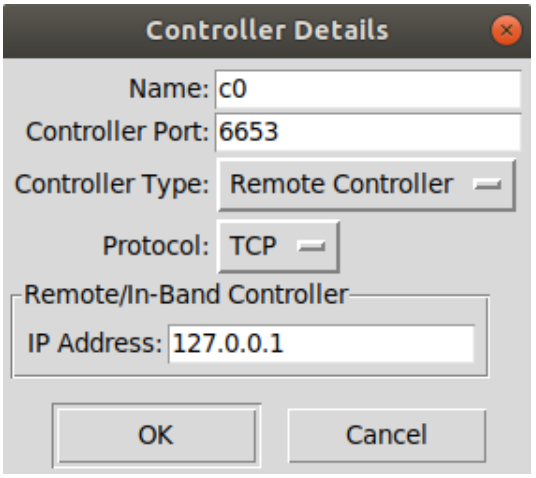


Figura III.15: Configuración del controlador SDN en Miniedit. Fuente: Autor.

Con el controlador totalmente configurado, se procede a configurar lo que sería el Access Point o punto de acceso, como se muestra en la Figura III.16, para el Access Point, se puede modificar desde las Wlans utilizadas, el canal usado en la transmisión, el modo de uso, el tipo de Access Point, etc. Para cumplir con los objetivos de este proyecto, los parámetros de interés serán el canal o Channel, el modo o Mode, y el rango de la señal o Signal Range tanto en los Access Point como en las estaciones de usuario.

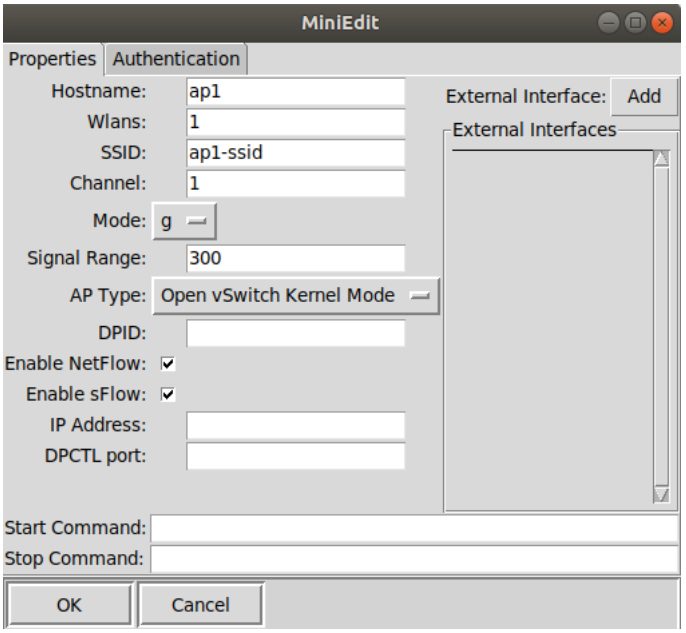


Figura III.16: Configuración del Access Point en Miniedit. Fuente : Autor.

Por último, se procede a configurar los parámetros de las estaciones uno por uno, en la Figura III.17 se muestra un ejemplo de la ventana de configuración, donde el único parámetro de interés para este caso será el rango de la señal.

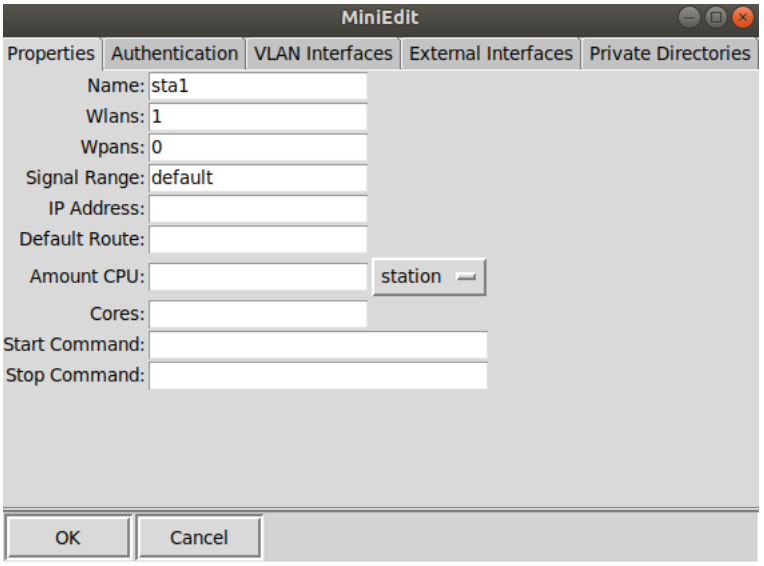


Figura III.17: Configuración de la Estación Móvil en Miniedit. Fuente: Autor.

Miniedit también ofrece la opción de generar un script de python con la topología utilizada, como se puede ver la parte superior izquierda de la Figura III.13 en la opción File, que al seleccionarla abrirá una ventana como la mostrada en la Figura III.18.

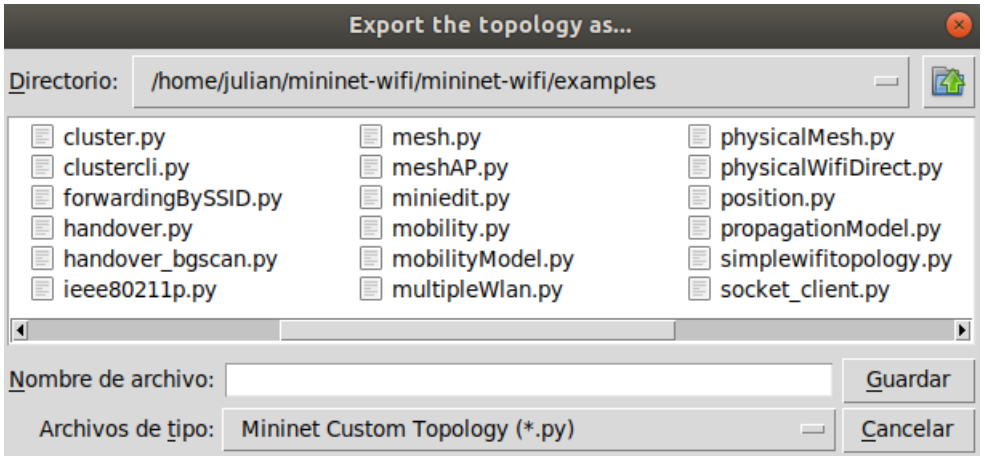


Figura III.18: Ventana para exportar la topología como un script de Python. Fuente: Autor.

1.5. Conexión entre Opendaylight y Mininet-WiFi

En esta parte del documento se hace un breve instructivo de cómo conectar el controlador con Mininet-WiFi ya que este ofrece varias herramientas como la interfaz gráfica de usuario que facilitan el manejo de las redes SDN y la visualización de información. Los pasos necesarios para conectar ODL con Mininet-WiFi utilizando una topología por defecto de mininet son los siguientes:

1. Lo primero que se debe realizar en este instructivo será la inicialización del controlador como anteriormente se había explicado en la sección de instalación de Opendaylight, donde al haber realizado todo correctamente debería aparecer una ventana indicando que se inicializó el controlador, similar a la mostrada en la Figura III.5.

2. Para este caso se utilizará la topología por defecto que genera Mininet-WiFi, la cual consta de dos estaciones y un punto de acceso o access point, el comando ejecutado fue el siguiente.

   sudo mn –wifi –controller=remote,ip=127.0.0.1,port=6653 –mac –switch=ovsk, protocols=OpenFlow13

   Para poder utilizar el controlador ODL es necesario especificar la dirección IP del mismo, la cual depende del método de instalación, previamente mencionado, el puerto del controlador, el cual para ODL se encuentra entre el puerto 6633 y el puerto 6653 y por último la versión de Openflow deseada, para este caso es la 1.1.3.

   Luego de haber ingresado el comando, el resultado obtenido debe ser como el mostrado en la Figura III.19

```
julian@julian-ThinkPad-E460:~/mininet-wifi$ sudo mn --wifi --controller=remote,i
p=127.0.0.1,port=6653 --mac --switch=ovsk,protocols=OpenFlow13
*** Adding stations:
sta1 sta2
*** Adding access points:
ap1
*** Configuring wifi nodes...
*** Creating network
*** Adding controller
*** Adding hosts:

*** Adding switches:

*** Adding links:
(sta1, ap1) (sta2, ap1)
*** Starting controller(s)
c0
*** Starting L2 nodes
ap1 ...
*** Starting CLI:
mininet-wifi>
```

Figura III.19: Inicialización por defecto Mininet-wifi utilizando controlador ODL. Fuente: Autor.

3. Ya con la red inicializada, es necesario realizar una prueba de conexión con el objetivo de verificar el correcto funcionamiento del controlador, esto se puede ver en la Figura III.20, donde se utilizaron los comandos pingall y pingallfull para probar la conexión entre estaciones.

```
mininet-wifi> pingallfull
*** Ping: testing ping reachability
sta1 -> sta2
sta2 -> sta1
*** Results:
 sta1->sta2: 1/1, rtt min/avg/max/mdev 0.581/0.581/0.581/0.000 ms
 sta2->sta1: 1/1, rtt min/avg/max/mdev 0.242/0.242/0.242/0.000 ms
mininet-wifi> pingall
*** Ping: testing ping reachability
sta1 -> *** sta1 : ('ping -c1  10.0.0.2',)
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=0.390 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.390/0.390/0.390/0.000 ms
sta2
sta2 -> *** sta2 : ('ping -c1  10.0.0.1',)
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.277 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.277/0.277/0.277/0.000 ms
sta1
*** Results: 0% dropped (2/2 received)
mininet-wifi>
```

Figura III.20: Prueba de conexión de la red con el controlador ODL. Fuente: Autor.

En efecto, se puede evidenciar que el controlador está funcionando correctamente,

ya que hay comunicación entre estaciones y no se presenta pérdida de paquetes.

4. Una última prueba que es posible realizar, es , utilizando la interfaz gráfica del controlador, verificar la topología creada, en este caso esta constaría como se mencionó anteriormente de dos estaciones y un punto de acceso. Para realizar esta prueba se debe realizar lo siguiente.

   Lo primero seria ingresar a la siguiente dirección desde el navegador:

   http://127.0.0.1:8181/index.html

   Ya con esta dirección debería aparecer una ventana como la mostrada en la Figura III.21, donde el usuario y la clave serán la palabra admin.

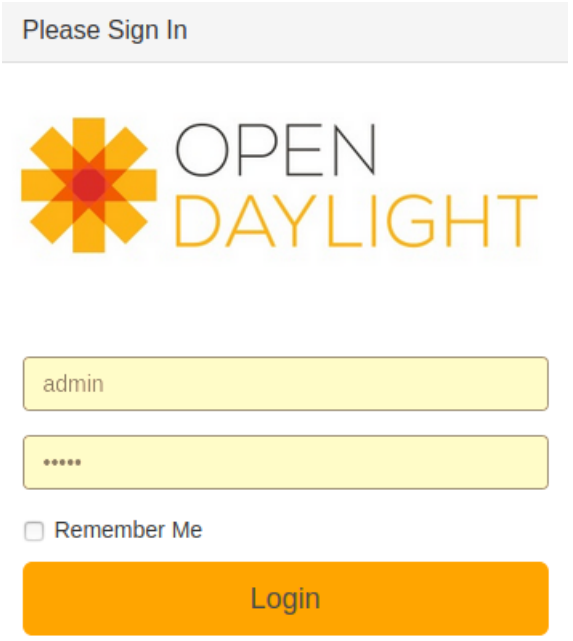


Figura III.21: Página de inicio interfaz gráfica Dlux. Fuente: Autor.

Después de ingresar el usuario y la contraseña, si todo esta funcionando correctamente aparecerá una ventana similar a la Figura III.22 con la topología creada.

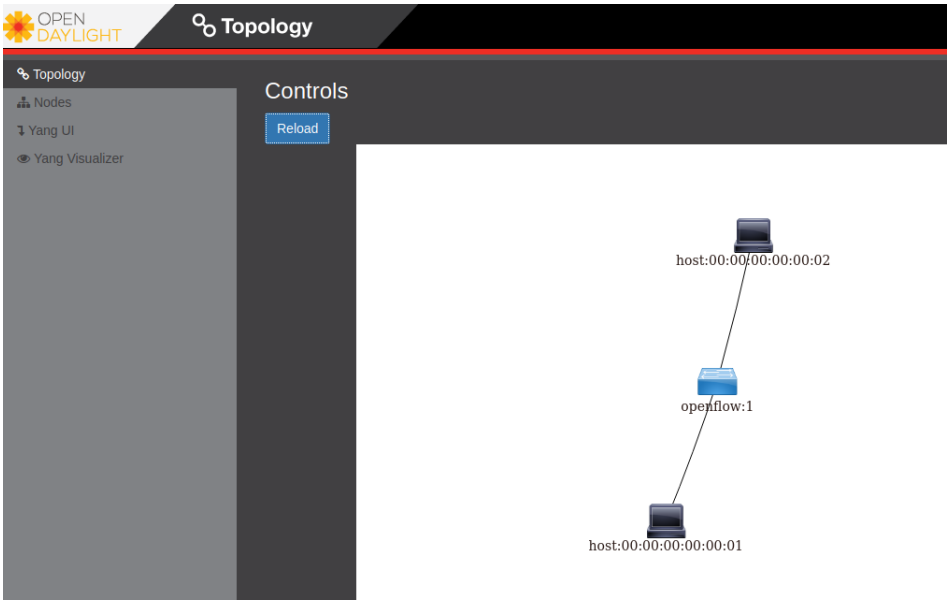


Figura III.22: Red creada y visualizada mediante la interfaz gráfica de ODL. Fuente: Autor.

1.6. Uso de Mininet-WiFi en conjunto con scripts de Python

La declaración de cualquier red en Mininet-WiFi se hace en lenguaje Python, por esta razón, cualquier modificación que se quiera realizar a la estructura de la red se puede realizar a partir de comandos Python o dentro de un script, en este se define la topología de la red, conexiones, métodos de enrutamiento, dispositivos de la red (host, puntos de acceso, switch, entre otros), a continuación se enuncian algunos de los comandos usados en la creación de la red y su utilidad.

- **net=Mininet(topo):** Crea una clase net tipo Mininet, con la topología establecida anteriormente.
- **addSwitch():** Agrega un Switch a la clase net, los parámetros y configuración del Switch se establecen dentro de los parámetros de entrada de la función.
- **addHost():** Agrega un host a la clase net, esta función tiene como parámetros de entrada características de la estación como el nombre, dirección IP, Posición espacial en un eje coordenado y demás.
- **addLink():** Con esta función de la clase net se agrega una conexión entre dos nodos, se pueden modificar algunos parámetros de interés como ancho de banda, delay y pérdidas en el enlace.

Para el desarrollo del documento se utilizó el script de pyhton Red2WE.py de la sección de anexos, donde se genera una red con 4 estaciones, 1 punto de acceso y un controlador, como se muestra en la Figura III.23

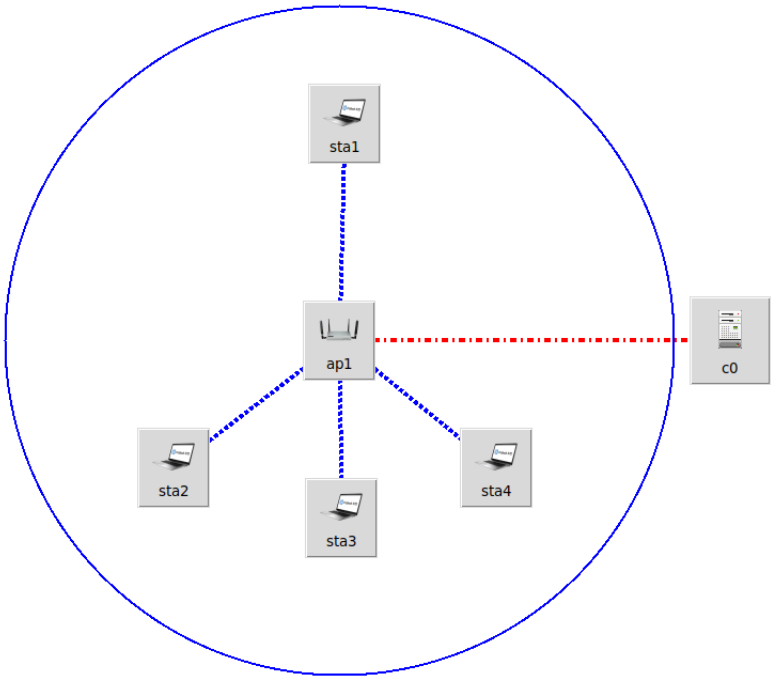


Figura III.23: Red generada por el script de Python utilizado. Fuente: Autor.

Capítulo IV

Evaluación y Resultados

En esta sección del documento se encuentra todo lo relacionado con la evaluación de controlador de capa cruzada en el entorno de operación Mininet-WiFi, en donde se utilizará un controlador SDN Opendaylight, y para la parte de SDR se hará una analogía donde estos serán representados por las estaciones y el punto de acceso. El primer aspecto a tratar será todo lo relacionado con la potencia de transmisión, luego se tratará el ancho de banda y la tasa de transmisión y finalizando con la frecuencia de trabajo tanto del punto de acceso como de las estaciones utilizadas.

1. Potencia

Para cambiar la magnitud de la potencia transmitida de los puntos de acceso o de las estaciones, Mininet-WiFi ofrece varios métodos con los cuales se puede lograr dicho cambio, los cuales son: cambiando el rango de alcance de las estaciones, cambiando la ganancia de la antena y directamente cambiando su potencia de transmisión, como se muestra a continuación.

1.1. Rango de la señal

El primer método encontrado para variar la potencia fue mediante la variación del rango de la señal transmitida, Mininet-WiFi ofrece una gráfica en la cual ilustra dichos rangos según el modelo de propagación utilizado, para el script Red2WE.py se utilizó el modelo de pérdidas de propagación de Friis con pérdidas de 2, dichos rangos se cargan con un valor inicial, y mediante el script rango.py que se muestra en los anexos, se modifican según lo requiera el usuario. Para poder cargar dicho archivo .py mientras se encuentra mininet-WiFi en simulación, es necesario utilizar el siguiente comando.

source /direcciondelarchivo/nombreDelArchivo.py

Donde, para este caso el nombre del archivo es rango.py como ya se había mencionado. En la Figura IV.1 se muestra una configuración inicial de los rangos de operación de los SDR, donde el punto de acceso está denotado como **ap1** y las estaciones de usuario como **stx** donde la x representa el número de la estación.

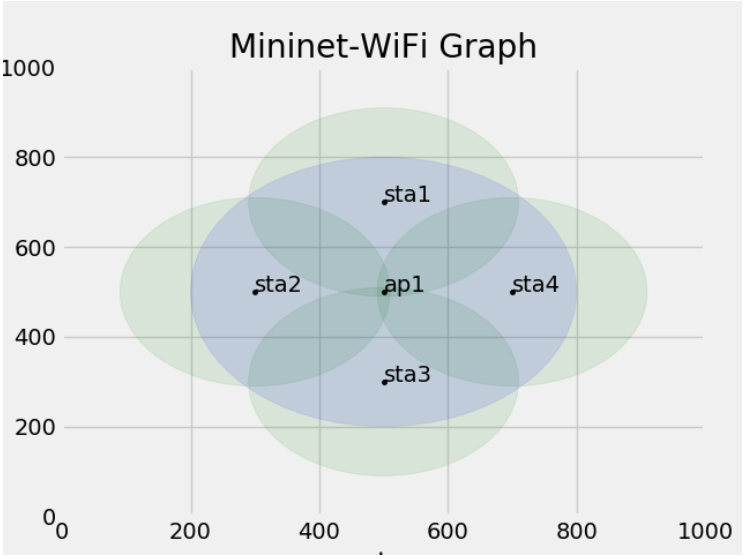


Figura IV.1: Red generada por el script de Python utilizado. Fuente: Autor.

Para el escenario mostrado en la Figura IV.1, se mostrará en la figura IV.2 la información de la potencia de su señal antes de realizar cambios en el rango del punto de acceso.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to ce:28:ff:64:5d:90 (on sta1-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 7395 bytes (170 packets)
        TX: 94 bytes (2 packets)
        signal: -78 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
```

Figura IV.2: Información del enlace entre el punto de acceso y la estación 1. Fuente: Autor.

Además de poder evidenciar el estado del enlace, se hizo una prueba de conectividad para verificar el funcionamiento del controlador y la tasa de pérdida de paquetes, esto se puede evidenciar en la Figura IV.3, donde claramente se puede observar que dichas pérdidas para este caso son del 0 % , es decir, que no hay pérdida de paquetes y que todo funciona correctamente. Cabe aclarar que con este test también es posible saber el retardo al enviar paquetes entre estaciones, lo cual será muy útil más adelante al momento de evaluar el rendimiento de las redes a utilizar.

```
--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 5.361/5.361/5.361/0.000 ms
sta2
sta2 -> *** sta2 : ('ping -c1  10.0.0.4',)
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.742 ms

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.742/0.742/0.742/0.000 ms
sta4 *** sta2 : ('ping -c1  10.0.0.1',)
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=2.51 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.514/2.514/2.514/0.000 ms
sta1 *** sta2 : ('ping -c1  10.0.0.3',)
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=3.09 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.096/3.096/3.096/0.000 ms
sta3
*** Results: 0% dropped (12/12 received)
mininet-wifi>
```

Figura IV.3: Prueba 1 de tasa de pérdida de paquetes en la conexión entre estaciones. Fuente: Autor.

Para poder evidenciar el efecto que se tiene sobre la señal el cambiar el rango, se redujo el rango del punto de acceso ap1 en gran medida, como se puede ver en la Figura IV.4.

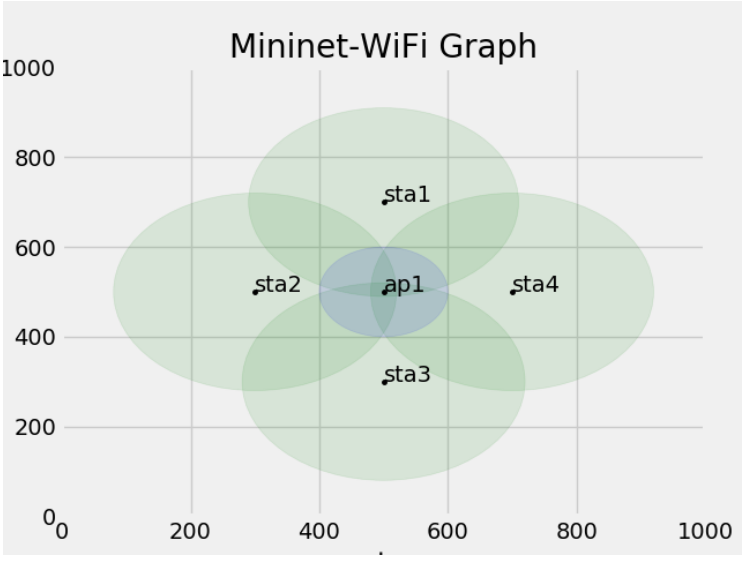


Figura IV.4: Variación de Rango de ap1. Fuente: Autor.

Luego de haber modificado el rango de ap1, se procede a realizar de nuevo un diagnóstico del estado del enlace entre las estaciones y el punto de acceso para verificar la variación en la señal, los resultados se pueden observar en la Figura IV.5

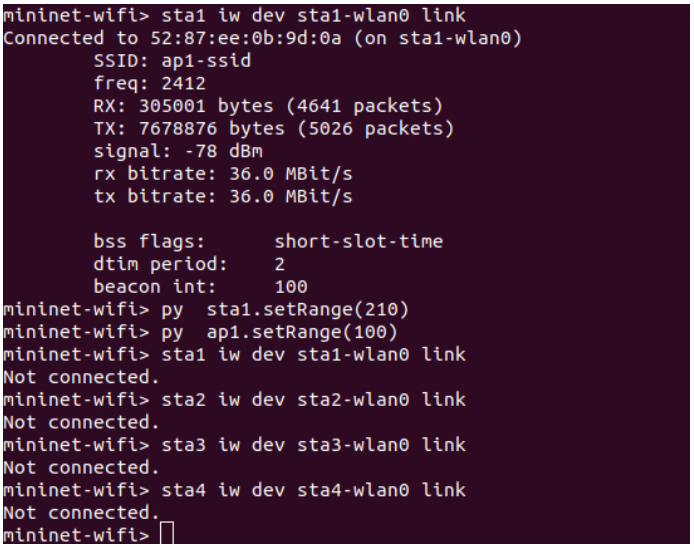


Figura IV.5: Estado del enlace entre estaciones y el punto de acceso luego del cambio de rango de ap1. Fuente: Autor.

Al parecer al disminuir tanto el rango de ap1 se pierde la conexión con las estaciones, ya que aunque se puede observar cómo los rangos de las estaciones llegan al punto de acceso, este al ser tan pequeño no logra establecer un enlace. Se puede afirmar que hubo cambio en la potencia de ap1 ya que claramente en telecomunicaciones móviles, la potencia requerida para cubrir un área es proporcional al tamaño de dicha área.

Para este caso, se volvió a realizar la prueba de pérdida de paquetes para verificar si efectivamente se había perdido la conexión entre estaciones y el punto de acceso, esto se muestra en la Figura IV.6.

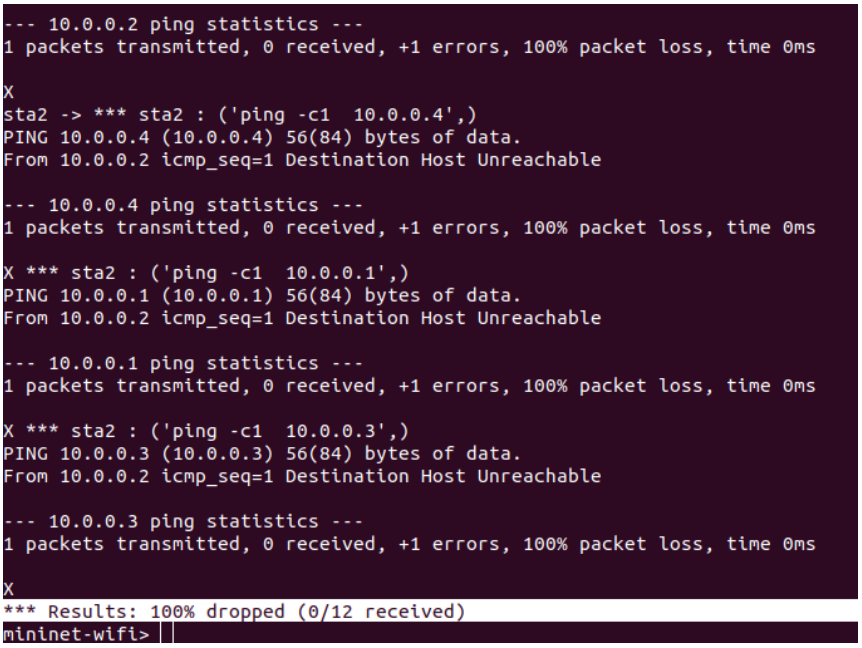


Figura IV.6: Prueba 2 de tasa de pérdida de paquetes en la conexión entre estaciones. Fuente: Autor.

Al observar la Figura IV.6, el resultado marca que las pérdidas de paquetes están en el 100 %, lo cual confirma sin lugar a dudas que al haber disminuido tanto el rango del punto de acceso se perdió la conexión entre estaciones por lo cual todos los paquetes enviados en la prueba son descartados. Además, se muestra que no es posible evaluar el retardo en el envió de paquetes ya que existe un aviso que indica que no se pueden alcanzar las estaciones o "destinos".

1.2. Ganancia de la antena

Otro aspecto capaz de modificar la potencia en el enlace es la Ganancia de la antena utilizada en los puntos de acceso y las estaciones, para esto se utilizó de manera análoga al rango un script de python llamado GananciaAntena.py el cual se encuentra en la sección de anexos. El escenario utilizado para estas pruebas es el mostrado en la Figura IV.7.

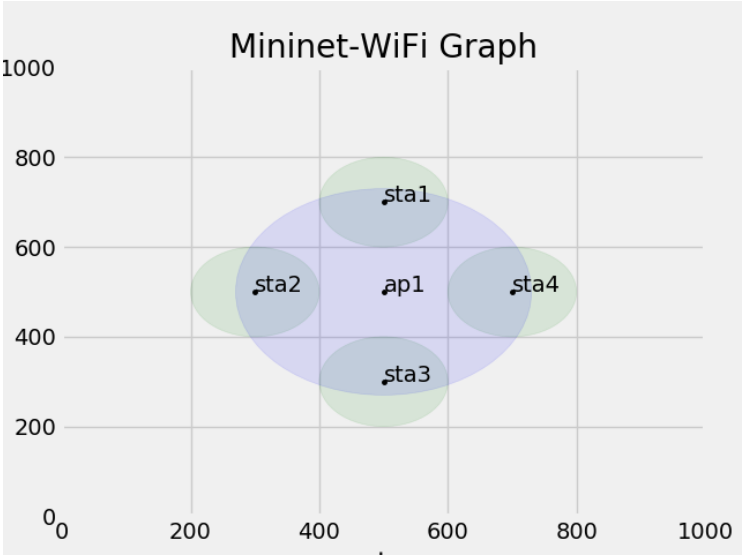


Figura IV.7: Escenario para pruebas con ganancia de antena. Fuente: Autor.

Para este escenario, se modificarán las ganancias tanto de ap1 como de sta1 para verificar el correcto funcionamiento en los enlaces y que corresponda con los cambios realizados, lo primero es verificar cúales son las ganancias por defecto que se tienen en el script de Red2WE.py, como se muestra en la Figura IV.8.

```
*** Starting CLI:
mininet-wifi> py ap1.wintfs[0].antennaGain
5
mininet-wifi> py sta1.wintfs[0].antennaGain
5
mininet-wifi> py sta2.wintfs[0].antennaGain
5
mininet-wifi> py sta3.wintfs[0].antennaGain
5
mininet-wifi> py sta4.wintfs[0].antennaGain
5
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 8a:65:f8:0d:e8:2b (on sta1-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 29301 bytes (668 packets)
        TX: 196 bytes (3 packets)
        signal: -78 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi>
```

Figura IV.8: Ganancias por defecto de las estaciones y el punto de acceso. Fuente: Autor

Como se puede ver en la Figura IV.8, las ganancias por defecto para todos los elementos es de 5, luego de esto se procede a verificar la potencia de la señal en el enlace entre la estación 1 o sta1 y el punto de acceso o ap1, que para este caso es de -78dBm. Lo

siguiente a realizar será el cambio en la ganancia del punto de acceso, para esta prueba se incrementó la ganancia pasando de 5 a 15, como se puede observar en la Figura IV.9.

```
mininet-wifi> py  ap1.setAntennaGain(15)
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 8a:65:f8:0d:e8:2b (on sta1-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 87633 bytes (1994 packets)
        TX: 400 bytes (5 packets)
        signal: -68 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
```

Figura IV.9: Estado del enlace entre sta1 y ap1 luego de aumentar la ganancia en ap1. Fuente: Autor

Como se esperaba, al aumentar la ganancia en ap1, aumenta la potencia de la señal, en este caso, pasa de -78dBm a -68dBm, lo que supone un aumento de 10 dB en la señal recibida. La siguiente prueba realizada fue, aumentar la ganancia de nuevo, pero esta vez, de sta1, esto con el objetivo de observar que pasa con el enlace, esto se muestra en la Figura IV.10.

```
mininet-wifi> py ap1.wintfs[0].antennaGain
15
mininet-wifi> py  sta1.setAntennaGain(15)
mininet-wifi> py sta1.wintfs[0].antennaGain
15
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 8a:65:f8:0d:e8:2b (on sta1-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 127051 bytes (2890 packets)
        TX: 502 bytes (6 packets)
        signal: -58 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi>
```

Figura IV.10: Estado del enlace entre sta1 y ap1 luego de aumentar la ganancia en sta1. Fuente: Autor.

De nuevo, al aumentar la ganancia de una antena, se puede observar como aumenta aun más la potencia de la señal pasando de -68dBm a -58dBm, lo que evidentemente es un gran cambio en la potencia de la señal. Por último, se realizará una última prueba donde, se disminuirá la ganancia en sta1, pasando de una ganancia de 15 a una de 10, y se verificará el estado del enlace entre sta1 y ap1, esto se evidencia en la Figura IV.11.

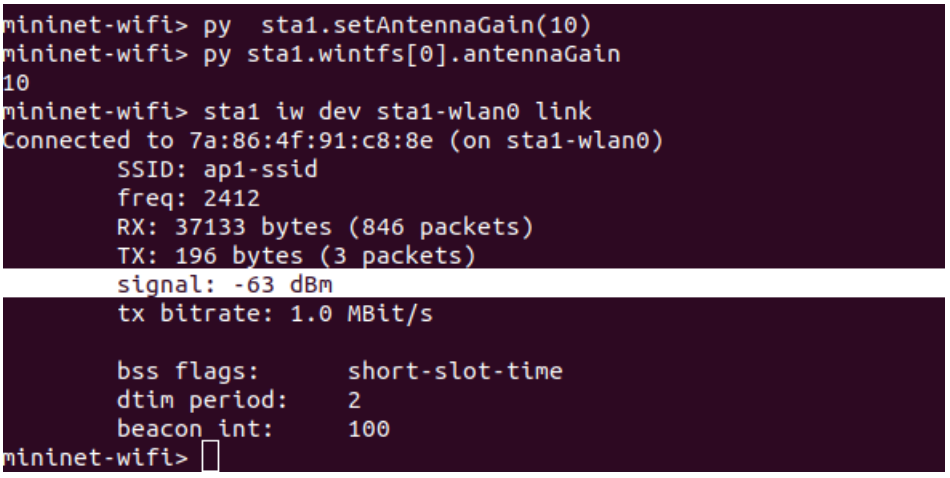


Figura IV.11: Estado del enlace entre sta1 y ap1 luego de disminuir la ganancia en sta1. Fuente: Autor.

Claramente, se puede observar como, al disminuir la ganancia en sta1, la potencia de la señal en el enlace disminuye, pasando de -58dBm a -63dBm.

Todas la pruebas realizadas anteriormente son análogas para cualquier estación en la cual se requiera realizar un cambio en la ganancia de la antena para modificar la potencia del enlace entre estaciones y puntos de acceso.

1.3. Potencia de la señal

El último método para cambiar la potencia de transmisión ofrecida por Mininet-WiFi, es, directamente variar la potencia con la que transmite la antena de las estaciones y los puntos de acceso, para este método se realizarán pruebas similares a como se hizo con la ganancia de la antena y se verificaron resultados. En este punto se utilizó además del script que crea la topología de red, el script PotenciaTx.py, el cual se encuentra en la sección de anexos. El escenario inicial en estas pruebas será el mismo utilizado en la parte de ganancia de antenas, como se muestra en la Figura IV.7.

Lo primero que se debe realizar para estas pruebas, es comprobar la potencia por defecto que ofrece el script Red2WE.py, esto se ilustra en la Figura IV.12.

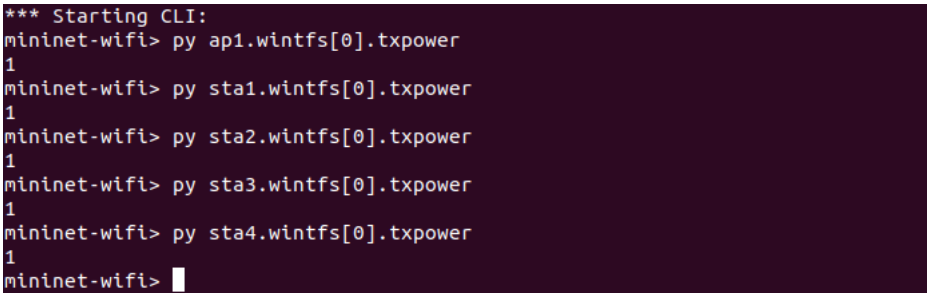


Figura IV.12: Potencias de transmisión por defecto para puntos de acceso y estaciones. Fuente: Autor.

En esta ocasión, la potencia por defecto es de 1 para todos los elementos. Las pruebas realizadas se hicieron con la estación 3 o sta3, esto con el objetivo de evidenciar que el comportamiento de todas la estaciones es igual sin importar la que se escoja, el estado del enlace inicial entre sta3 y ap1 se puede observar en la Figura IV.13.

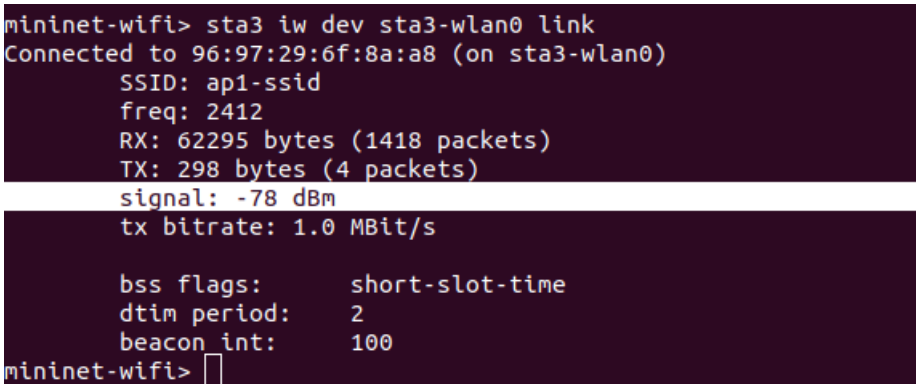


Figura IV.13: Estado inicial del enlace entre sta3 y ap1. Fuente: Autor.

La primera prueba a realizar será el aumento de la potencia en ap1, pasando de 1 a 5, como se puede ver en la Figura IV.14.

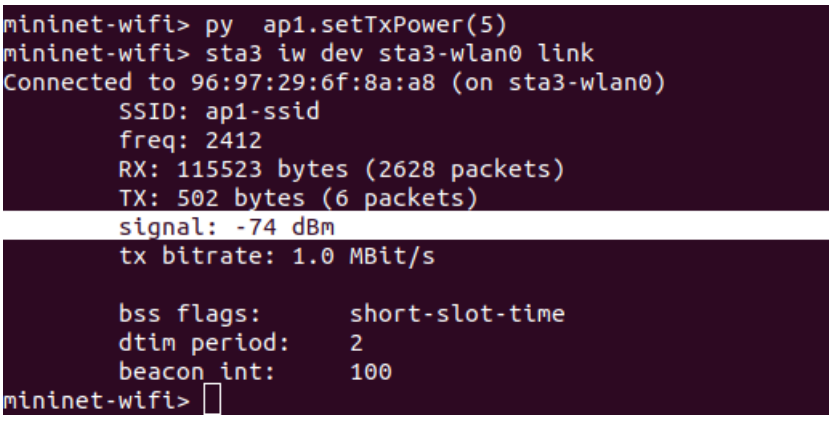


Figura IV.14: Estado inicial del enlace entre sta3 y ap1. Fuente: Autor.

Como se esperaba, al aumentar la potencia de transmisión en ap1, aumenta la señal en el enlace entre sta3 y ap1. La siguiente prueba es el aumento en la potencia en sta3, pasando de -78dBm a -74dBm, como se muestra en la Figura IV.14.

```
mininet-wifi> py sta3.wintfs[0].txpower
1
mininet-wifi> py  sta3.setTxPower(5)
mininet-wifi> py sta3.wintfs[0].txpower
5
mininet-wifi> sta3 iw dev sta3-wlan0 link
Connected to 56:67:3b:46:da:17 (on sta3-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 21387 bytes (488 packets)
        TX: 94 bytes (2 packets)
        signal: -74 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi>
```

Figura IV.15: Estado del enlace entre sta3 y ap1 al aumentar la potencia en sta3. Fuente. Autor.

En esta parte, es curioso observar como, aunque se aumente la potencia en sta3 pasando de una potencia de 1 a una de 5 como se puede observar en la Figura IV.15, el estado del enlace no cambió, es decir, se mantuvo en -74dBm, a diferencia a como lo hizo con ap1 y como sucedió anteriormente al modificar la ganancia en la antena en las estaciones, esto se puede deber simplemente a la estructura con la que se desarrolló Mininet-WiFi.

La última prueba realizada fue, disminuir la potencia transmitida en ap1, pasando de 5 a 3, y como se esperaba, la señal del enlace disminuyó, pasando de -74dBm a -76dBm, esto se muestra en la Figuras IV.15.

```
mininet-wifi> py  ap1.setTxPower(3)
mininet-wifi> py ap1.wintfs[0].txpower
3
mininet-wifi> sta3 iw dev sta3-wlan0 link
Connected to fa:6d:f0:42:b1:7b (on sta3-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 18923 bytes (432 packets)
        TX: 94 bytes (2 packets)
        signal: -76 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi>
```

Figura IV.16: Estado del enlace entre sta3 y ap1 al disminuir la potencia de ap1. Fuente: Autor.

2. Velocidad o tasa de transmisión

Además de la potencia, otro elemento de las SDR que se planteó modificar en un inicio fue la tasa de transmisión que estas manejaban, ya que al variar este parámetro, se varia indirectamente la modulación utilizada en la transmisión, lo cual es básicamente uno de los objetivos de este trabajo. Para lograr variar la velocidad de transmisión se revisaron las herramientas ofrecidas por Mininet-WiFi en las cuales se descubrió que este software permite la operación de las estaciones y de los puntos de acceso en los estándares IEEE 802.11g, IEEE 802.11a, IEEE 802.11b y IEEE 802.11n cuyas tasas de transmisión teóricas máxima y frecuencias de trabajo, se pueden observar en la Tabla 4.1 [21].

Estándar	**Frecuencia de Trabajo (GHz)**	**Velocidad de transmisión máxima teórica (Mbps)**
IEEE 802.11g	2.4	54
IEEE 802.11b	2.4	11
IEEE 802.11a	5	54
IEEE 802.11n	2.4 y 5	600

Tabla 4.1: Estándares WiFi, velocidad de transmisión máxima teórica y frecuencias de trabajo [21].

Conociendo las tasas de transmisión máximas para cada estándar ofrecido por Mininet-WiFi, lo siguiente es realizar las pruebas de cambio de velocidades de transmisión en la red creada en el script Red2WE.py en la cual como ya se había mencionado, posee 4 estaciones, 1 punto de acceso y 1 controlador SDN.

Para la primera prueba se utilizó el estándar IEEE 802.11g, es decir, que se dejó operando ap1, sta1,sta2,sta3 y sta4 en la banda de frecuencias que rondan los 2.4GHz y con una velocidad máxima de transmisión teórica de 54Mbps. Las condiciones iniciales para este experimento se pueden observar a continuación en la Figura IV.17.

```
julian@julian-ThinkPad-E460:~/Escritorio/PythonTesis$ sudo python Red2WE.py
Ingrese el modo de Trabajo de a,b,g,n: g
Ingrese el canal de Trabajo: 1
Ingrese el rango de trabajo del access point 1: 300
Ingrese el rango de trabajo de la estacion  1: 230
Ingrese el rango de trabajo de la estacion  2: 230
Ingrese el rango de trabajo de la estacion  3: 230
Ingrese el rango de trabajo de la estacion  4: 230
*** Adding controller
*** Add switches/APs
*** Add hosts/stations
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Add links
*** Starting network
*** Starting controllers
*** Starting switches/APs
*** Post configure nodes
*** Starting CLI:
mininet-wifi>
```

Figura IV.17: Condiciones iniciales experimento con estándar IEEE 802.11g canal 1. Fuente: Autor.

Para este caso, se realizó además, una prueba para verificar retardos y pérdida de paquetes en este experimento, lo cual se muestra en la Figura IV.18.

```
*** Results: 0% dropped (12/12 received)
mininet-wifi> pingallfull
*** Ping: testing ping reachability
sta4 -> sta1 sta3 sta2
sta1 -> sta4 sta3 sta2
sta3 -> sta4 sta1 sta2
sta2 -> sta4 sta1 sta3
*** Results:
 sta4->sta1: 1/1, rtt min/avg/max/mdev 2.413/2.413/2.413/0.000 ms
 sta4->sta3: 1/1, rtt min/avg/max/mdev 3.561/3.561/3.561/0.000 ms
 sta4->sta2: 1/1, rtt min/avg/max/mdev 3.488/3.488/3.488/0.000 ms
 sta1->sta4: 1/1, rtt min/avg/max/mdev 2.444/2.444/2.444/0.000 ms
 sta1->sta3: 1/1, rtt min/avg/max/mdev 1.129/1.129/1.129/0.000 ms
 sta1->sta2: 1/1, rtt min/avg/max/mdev 0.904/0.904/0.904/0.000 ms
 sta3->sta4: 1/1, rtt min/avg/max/mdev 1.774/1.774/1.774/0.000 ms
 sta3->sta1: 1/1, rtt min/avg/max/mdev 0.882/0.882/0.882/0.000 ms
 sta3->sta2: 1/1, rtt min/avg/max/mdev 0.824/0.824/0.824/0.000 ms
 sta2->sta4: 1/1, rtt min/avg/max/mdev 1.556/1.556/1.556/0.000 ms
 sta2->sta1: 1/1, rtt min/avg/max/mdev 0.672/0.672/0.672/0.000 ms
 sta2->sta3: 1/1, rtt min/avg/max/mdev 0.761/0.761/0.761/0.000 ms
mininet-wifi>
```

Figura IV.18: Prueba de pérdida de paquetes para condiciones iniciales con estándar IEEE 802.11g canal 1. Fuente: Autor.

De la Figura IV.18 se puede decir dos cosas, primero que no hay pérdida de paquetes, y segundo, se pueden obtener los resultados de los retardos obtenidos para cada conexión entre estaciones al realizar ping. Específicamente los valores de retardo obtenidos son el valor mínimo, el valor promedio y el valor máximo respectivamente.

Después de fijar las condiciones iniciales de la red, el siguiente paso es verificar la velocidad de transmisión máxima que esta maneja, ya que esta no siempre coincide con el máximo teórico antes visto en la Tabla 4.1, para esto, lo primero es hacer un test de ancho

de banda entre estaciones, para este caso se escogieron las estaciones sta1 y sta2 (aunque con cualquier estación funciona igual), y después se revisa el estado del enlace para la estación 1, todo esto se puede observar en la Figura IV.19.

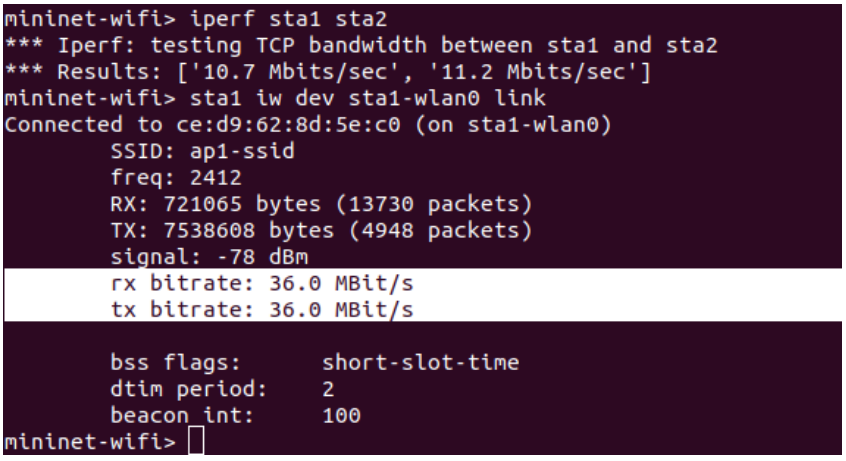


Figura IV.19: Test de ancho de banda y estado del enlace para condiciones iniciales. Fuente: Autor.

Después de verificar el estado del enlace en la estación 1, se pudo observar que para este caso no se llega a la velocidad máxima de transmisión teórica dictada por el estándar, ya que solo se alcanzan velocidades de 36 Mbps cuando se transmite y cuando se recibe, esto se puede observar en la Figura IV.19, así que, para este experimento, esta tasa será la considerada como máxima permitida, a lo cual los cambios a realizar, consistirán en, básicamente, disminuir dicha velocidad de transmisión. Además, de brindar información acerca de la tasa de transmisión, es posible observar el ancho de banda que se utiliza en el enlace, el cual para este caso es 10.7 Mbps en la recepción y 11.2 Mbps en la transmisión. Es bastante importante conocer el ancho de banda utilizado y la velocidad de transmisión ya que estos dos valores tienen una estrecha relación con el tipo de modulación utilizada tanto en el transmisor como en el receptor del enlace, lo cual para este proyecto, es uno de los objetivos.

Continuando con el experimento de cambio en la velocidad de transmisión en el enlace de la estación 1, como se sabe que el estándar IEEE 802.11g trabaja en la ventana de 2.4 GHz, el comando para cambiar la velocidad de transmisión en el punto de acceso es el siguiente:

```
ap1 iw dev ap1-wlan1 set bitrates legacy-2.4 11
```

En el anterior comando, se propuso como máxima tasa de transmisión 11 Mbps, además, en el caso de que se estuviera manejando un estándar que trabajara en la ventana de 5 GHz, solo se cambiaría el 2.4 por un 5 en el comando. En la Figura IV.20 se muestra como efectivamente se cambio la velocidad de transmisión en el receptor del enlace de la estación 1. Cabe aclarar que, las velocidades que se pueden configurar en Mininet son reducidas y seguramente tienen que ver con especificaciones de los estándares WiFi o de los elementos emulados.

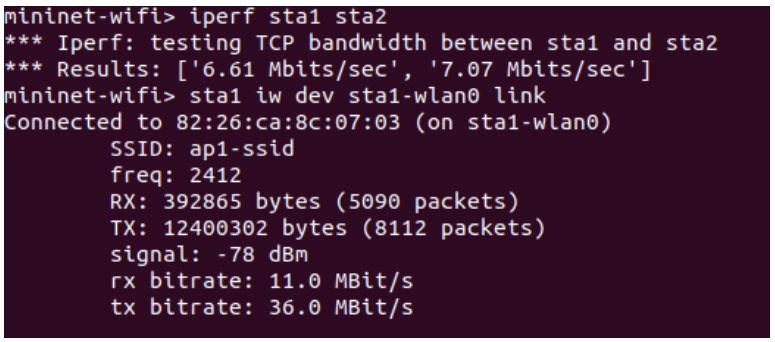


Figura IV.20: Cambio de velocidad de transmisión en el punto de acceso. Fuente: Autor.

Continuando el experimento, el comando requerido para cambiar la velocidad de transmisión en la estación 1 es el siguiente:

sta1 iw dev sta1-wlan0 set bitrates legacy-2.4 2

Al igual que se hizo con el punto de acceso, se procede a verificar el cambio del ancho de banda en el enlace y el estado en el mismo, para comprobar la velocidad máxima permitida en el transmisor, todo esto se puede observar en la Figura IV.21.

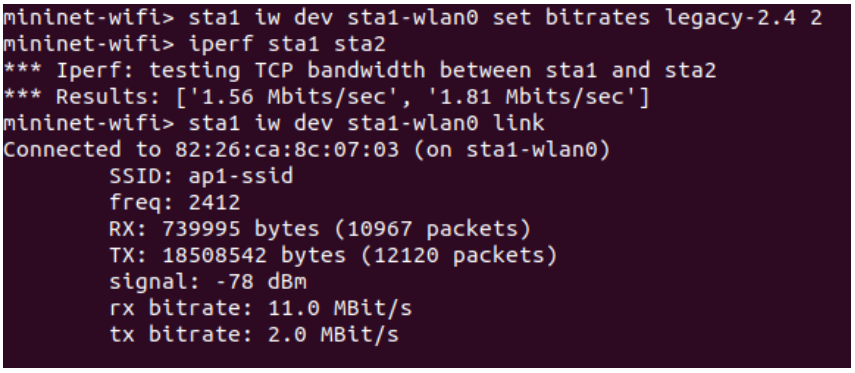


Figura IV.21: Cambio de velocidad de transmisión en la estación 1. Fuente: Autor.

Efectivamente, Para ambos casos, fue posible cambiar la velocidad máxima de transmisión, por lo que lo último que resta en este experimento es el evaluar si existió algún cambio significativo en los retardos al enviar paquetes y la tasa de pérdida de paquetes, como se muestra en la Figura IV.22.

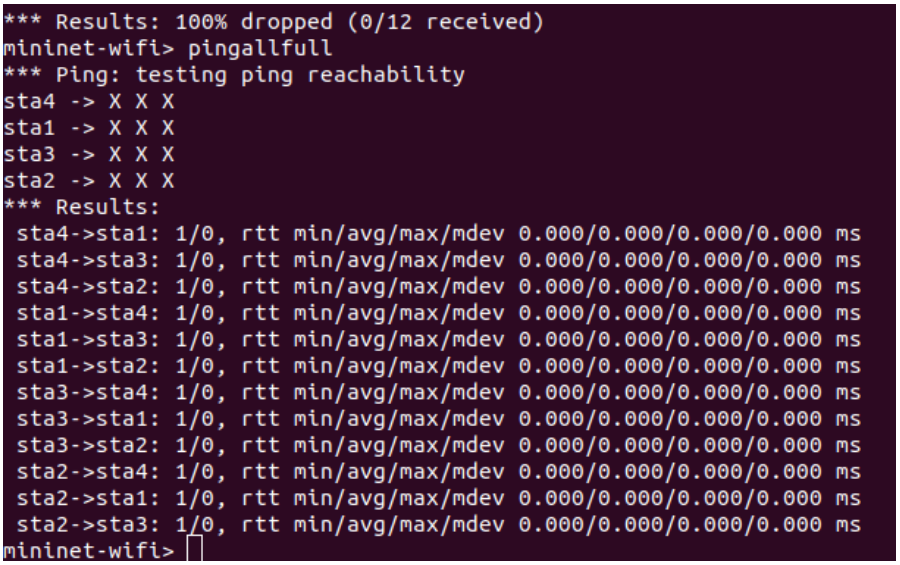


Figura IV.22: Prueba de pérdida de paquetes luego de experimentación con velocidad de transmisión. Fuente: Autor.

Para este caso, hubo un gran cambio tanto para el retardo como para la tasa de pérdidas, ya que se tiene que la tasa de pérdidas es del 100 % y no es posible comunicar las estaciones, esto puede deberse a variar razones, la primera es que al disminuir la tasa original de manera tan significativa, se cambia el tipo de modulación utilizada por lo cual, los demás dispositivos no son capaces de comunicarse debido a su configuración inicial, la segunda razón podría ser, que los dispositivos al funcionar a tasas de transmisión distintas no tengan compatibilidad.

Para otros estándares WiFi, el proceso de experimentación es exactamente el mismo, por lo que no se incluirá en el documento para evitar redundancia.

3. Frecuencia de trabajo

La frecuencia de trabajo en conexiones inalámbricas WiFi depende de la banda en la que se encuentre el enlace, 2.4 GHz o 5 GHz (Banda libre), además del canal que ocupe dentro de la banda, cada canal tiene un ancho de banda de 20 MHz, ya que nos encontramos trabajando bajo los estándares IEEE que se encuentran pre-establecidos dentro de Mininet-WiFi cómo se puede apreciar en la tabla 4.1.

Teniendo en cuenta que para el desarrollo del trabajo nos encontramos limitados a las limitaciones del software, se restringió la operabilidad a los estándares IEEE 802.11g, 802.11b, 802.11a, 802.11n, teniendo en cuenta que cada estándar funciona con modelos distintos de modulación y codificación, protocolos de seguridad, canales de operación y bandas de frecuencia, hace que al cambiar el modo de operación, se realice de manera directa un cambio en los aspectos mencionados anteriormente, en la tabla 4.2 se puede ver el tipo de modulación usado para cada estándar.

Estándar	Método de codificación/Modulación
IEEE 802.11g	DSSS y OFDM
IEEE 802.11b	CSMA/CA
IEEE 802.11a	OFDM
IEEE 802.11n	MIMO SDM

Tabla 4.2: Métodos de codificación/modulación implementados en estándares WiFi [21].

En la Figura IV.23 se puede ver de manera condensada y ordenada cronológicamente la transformación del estándar IEEE 802.11x, de acuerdo a la información mencionada anteriormente.

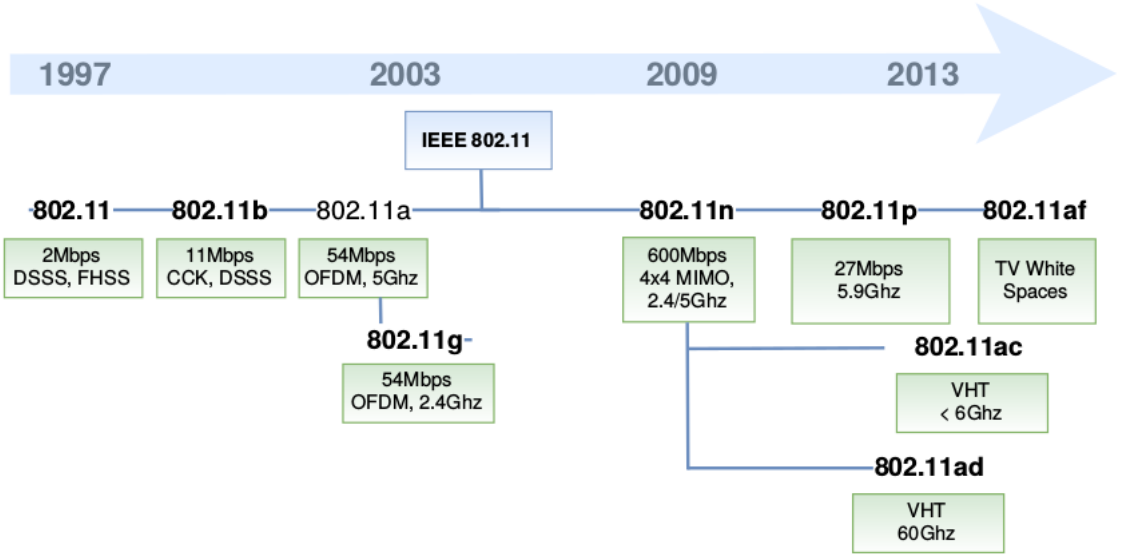


Figura IV.23: Evolución del estándar IEEE 802.11 [21].

En la red que se plantea en este trabajo al iniciar la emulación se selecciona el modo de operación y el canal de trabajo, esto se realiza a través de una ventana de diálogo que permite al operario seleccionar estos parámetros, en una red física es posible realizar cambios en el modo de operación mientras la red está funcionando, sin embargo, en el software Mininet-WiFi este cambio solo es posible hacerlo cuando se declara la topología de la red, lo que nos impide hacer cambios en frecuencia durante la emulación.

La selección del modo de operación y el canal de funcionamiento se realiza iniciando la emulación, es necesario repetir este procedimiento un número de veces igual al número de puntos de acceso (AP) que tenga la red, en la Figura IV.24 se puede ver una red con un único punto de acceso, el cual se configuró en modo 'g' y canal 3, esto quiere decir que está trabajando a una frecuencia central de 2.422 MHz.

```
  sys.version_info < (3,) and warnings.warn(pre + "*" * 60 + msg + "*" * 60)
Ingrese el modo de Trabajo del access point a,b,g,n: g
Ingrese el canal de Trabajo del access point: 3
Si desea cambiar el modo o el canal del access point debe reiniciar la simulación.
Ingrese el rango de trabajo del access point 1: 200
Ingrese el rango de trabajo de la estacion  1: 100
Ingrese el rango de trabajo de la estacion  2: 100
Ingrese el rango de trabajo de la estacion  3: 100
Ingrese el rango de trabajo de la estacion  4: 100
```

Figura IV.24: Configuración modo de operación y canal. Fuente: Autor.

Al realizar una prueba de la conexión de cualquier estación base con el punto de acceso como en la Figura IV.25, se puede apreciar el estado del link, en esta prueba se ve como la frecuencia de trabajo es la esperada.

```
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:04:00 (on sta1-wlan0)
        SSID: ap1-ssid
        freq: 2422
        RX: 8627 bytes (198 packets)
        TX: 94 bytes (2 packets)
        signal: -78 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
```

Figura IV.25: Estado del enlace estación Sta1 y Punto de acceso AP1. Fuente: Autor.

4. Scripts de Python para modificar los parámetros

Se realizaron archivos de extensión python para aprovechar la funcionalidad de Mininet-wifi de ejecutar comandos desde fuentes externas, se realizó un archivo diferente para cada uno de los parámetros a modificar, además de ser una manera de generalizar cada procedimiento, se busca que la persona encargada de manipular la red, lo pueda hacer de una manera amigable, todos estos archivos de extensión .py (python) se encuentran en la sección de anexos.

5. Escenario 1: Múltiples puntos de acceso

En esta parte del documento, se mostrará todo lo relacionado con las pruebas realizadas en un escenario con múltiples puntos de acceso y estaciones, donde se manejarán diferentes recursos para cada punto de acceso, lo cual es uno de los objetivos a tratar, ya que se requieren redes un poco más complejas que las manejadas anteriormente para demostrar la gran utilidad de utilizar un controlador de capa cruzada. El escenario a manejar se muestra en la Figura IV.26, además, el código de python utilizado se encuentra en la sección de anexos con el nombre de Escenario1E.py.

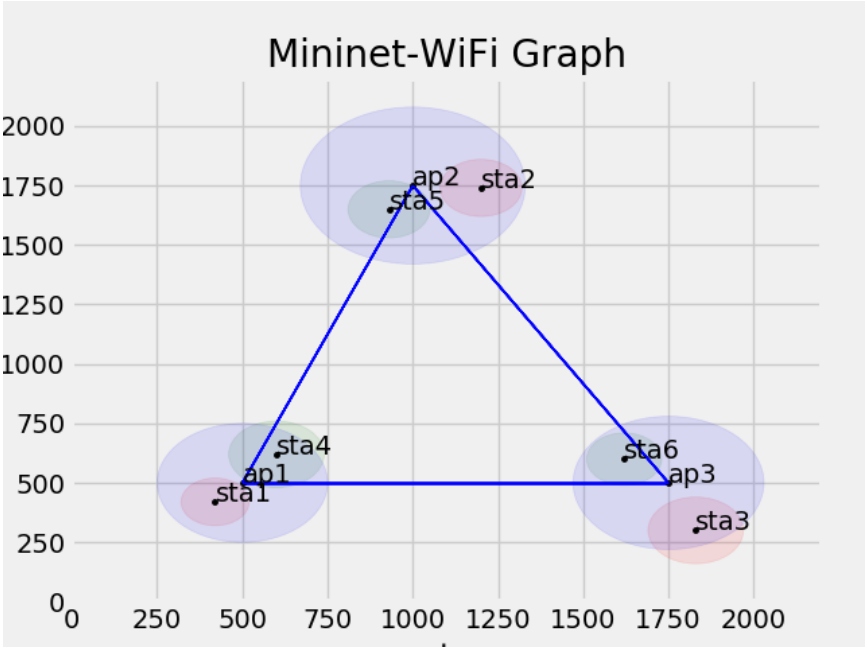


Figura IV.26: Escenario 1: Múltiples puntos de acceso. Fuente: Autor.

Para este escenario se utilizaron 6 estaciones con 3 puntos de acceso para dar complejidad a la red, además, las condiciones iniciales utilizadas se muestran en la Figura IV.27.

```
julian@julian-ThinkPad-E460:~/Escritorio/PythonTesis$ sudo python Escenario1E.py
Ingrese el modo de Trabajo de ap1 a,b,g,n: g
Ingrese el canal de Trabajo de ap1: 1
Ingrese el modo de Trabajo de ap2 a,b,g,n: g
Ingrese el canal de Trabajo de ap2: 6
Ingrese el modo de Trabajo de ap3 a,b,g,n: a
Ingrese el canal de Trabajo ap3: 36
Ingrese el rango de trabajo del access point 1: 250
Ingrese el rango de trabajo del access point 2: 330
Ingrese el rango de trabajo del access point 3: 280
Ingrese el rango de trabajo de la estacion  1: 100
Ingrese el rango de trabajo de la estacion  2: 120
Ingrese el rango de trabajo de la estacion  3: 140
Ingrese el rango de trabajo de la estacion  4: 140
Ingrese el rango de trabajo de la estacion  5: 120
Ingrese el rango de trabajo de la estacion  6: 110
```

Figura IV.27: Condiciones iniciales escenario 1. Fuente: Autor.

Ya con los parámetros iniciales en rango, estándar y canal utilizados, se procede a realizar pruebas de conexión para identificar una tasa inicial de pérdida de paquetes, esto se muestra en la Figura IV.28, donde se puede ver claramente que existe conexión entre todas las estaciones y que para esta prueba el porcentaje de pérdida de paquetes es del 0 %.

```
*** Results: 0% dropped (30/30 received)
mininet-wifi> pingallfull
*** Ping: testing ping reachability
sta1 -> sta3 sta2 sta4 sta5 sta6
sta3 -> sta1 sta2 sta4 sta5 sta6
sta2 -> sta1 sta3 sta4 sta5 sta6
sta4 -> sta1 sta3 sta2 sta5 sta6
sta5 -> sta1 sta3 sta2 sta4 sta6
sta6 -> sta1 sta3 sta2 sta4 sta5
```

Figura IV.28: Prueba de conexión escenario 1. Fuente: Autor.

Además de obtener información acerca de la tasa de pérdida de paquetes, esta prueba también ofrece información acerca de los retardos en el envío de paquetes entre estaciones, como se había mencionado anteriormente, y esto se muestra en la Figura IV.29 en donde, cada estación envía paquetes a la otras y se toman resultados en términos de milésimas de segundo o ms, ofreciendo retardos mínimos, promedio y máximos respectivamente, esto se profundizará más adelante realizando pruebas comparativas entre estaciones asociadas a diferentes puntos de acceso.

```
*** Results:
 sta1->sta3: 1/1, rtt min/avg/max/mdev 3.358/3.358/3.358/0.000 ms
 sta1->sta2: 1/1, rtt min/avg/max/mdev 3.577/3.577/3.577/0.000 ms
 sta1->sta4: 1/1, rtt min/avg/max/mdev 4.463/4.463/4.463/0.000 ms
 sta1->sta5: 1/1, rtt min/avg/max/mdev 2.349/2.349/2.349/0.000 ms
 sta1->sta6: 1/1, rtt min/avg/max/mdev 0.982/0.982/0.982/0.000 ms
 sta3->sta1: 1/1, rtt min/avg/max/mdev 1.214/1.214/1.214/0.000 ms
 sta3->sta2: 1/1, rtt min/avg/max/mdev 3.536/3.536/3.536/0.000 ms
 sta3->sta4: 1/1, rtt min/avg/max/mdev 3.379/3.379/3.379/0.000 ms
 sta3->sta5: 1/1, rtt min/avg/max/mdev 8.937/8.937/8.937/0.000 ms
 sta3->sta6: 1/1, rtt min/avg/max/mdev 0.830/0.830/0.830/0.000 ms
 sta2->sta1: 1/1, rtt min/avg/max/mdev 3.120/3.120/3.120/0.000 ms
 sta2->sta3: 1/1, rtt min/avg/max/mdev 1.893/1.893/1.893/0.000 ms
 sta2->sta4: 1/1, rtt min/avg/max/mdev 1.920/1.920/1.920/0.000 ms
 sta2->sta5: 1/1, rtt min/avg/max/mdev 2.683/2.683/2.683/0.000 ms
 sta2->sta6: 1/1, rtt min/avg/max/mdev 2.093/2.093/2.093/0.000 ms
 sta4->sta1: 1/1, rtt min/avg/max/mdev 0.797/0.797/0.797/0.000 ms
 sta4->sta3: 1/1, rtt min/avg/max/mdev 2.574/2.574/2.574/0.000 ms
 sta4->sta2: 1/1, rtt min/avg/max/mdev 1.711/1.711/1.711/0.000 ms
 sta4->sta5: 1/1, rtt min/avg/max/mdev 0.875/0.875/0.875/0.000 ms
 sta4->sta6: 1/1, rtt min/avg/max/mdev 1.813/1.813/1.813/0.000 ms
 sta5->sta1: 1/1, rtt min/avg/max/mdev 1.579/1.579/1.579/0.000 ms
 sta5->sta3: 1/1, rtt min/avg/max/mdev 1.759/1.759/1.759/0.000 ms
 sta5->sta2: 1/1, rtt min/avg/max/mdev 3.357/3.357/3.357/0.000 ms
 sta5->sta4: 1/1, rtt min/avg/max/mdev 3.099/3.099/3.099/0.000 ms
 sta5->sta6: 1/1, rtt min/avg/max/mdev 2.252/2.252/2.252/0.000 ms
 sta6->sta1: 1/1, rtt min/avg/max/mdev 1.291/1.291/1.291/0.000 ms
 sta6->sta3: 1/1, rtt min/avg/max/mdev 0.736/0.736/0.736/0.000 ms
 sta6->sta2: 1/1, rtt min/avg/max/mdev 2.017/2.017/2.017/0.000 ms
 sta6->sta4: 1/1, rtt min/avg/max/mdev 0.852/0.852/0.852/0.000 ms
 sta6->sta5: 1/1, rtt min/avg/max/mdev 1.606/1.606/1.606/0.000 ms
mininet-wifi>
```

Figura IV.29: Prueba de retardos escenario 1. Fuente: Autor.

Con las pruebas básicas de conexión realizadas, el siguiente paso en este escenario, es comprobar que las estaciones de los diferentes puntos de acceso trabajan con diferentes recursos tanto en estándar WiFi, velocidad de transmisión y frecuencia, ya que el rango de cada elemento fue definido anteriormente y para la ganancia de la antena y potencia es bastante sencillo realizar dichos cambios de ser necesarios. El primer enlace a comprobar será entre las estaciones del punto de acceso 1, como se muestra en la Figura IV.30, en la cual se toma de referencia la estación 1.

```
mininet-wifi> iperf sta1 sta4
*** Iperf: testing TCP bandwidth between sta1 and sta4
*** Results: ['6.33 Mbits/sec', '6.78 Mbits/sec']
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 46:0d:b7:af:d1:96 (on sta1-wlan0)
        SSID: ap1-ssid
        freq: 2412
        RX: 231270 bytes (3656 packets)
        TX: 4586360 bytes (3035 packets)
        signal: -79 dBm
        rx bitrate: 24.0 MBit/s
        tx bitrate: 36.0 MBit/s
```

Figura IV.30: Estado del enlace estación 1 escenario 1. Fuente: Autor.

El segundo enlace a comprobar será el de las estaciones que pertenecen al punto de acceso 2, esto se muestra en la Figura IV.31, donde se usa como referencia la estación 2.

```
mininet-wifi> iperf sta2 sta5
*** Iperf: testing TCP bandwidth between sta2 and sta5
*** Results: ['8.31 Mbits/sec', '8.71 Mbits/sec']
mininet-wifi> sta2 iw dev sta2-wlan0 link
Connected to ae:3a:82:70:f3:df (on sta2-wlan0)
        SSID: ap2-ssid
        freq: 2437
        RX: 282991 bytes (4588 packets)
        TX: 5842150 bytes (3855 packets)
        signal: -84 dBm
        rx bitrate: 18.0 MBit/s
        tx bitrate: 18.0 MBit/s
```

Figura IV.31: Estado del enlace estación 2 escenario 1. Fuente: Autor.

Por último, al igual que con los puntos de acceso 1 y 2, se realiza el mismo procedimiento con el punto de acceso 3, y se toma como referencia la estación 3 como se puede ver en la Figura IV.32.

```
mininet-wifi> iperf sta3 sta6
*** Iperf: testing TCP bandwidth between sta3 and sta6
*** Results: ['5.40 Mbits/sec', '5.99 Mbits/sec']
mininet-wifi> sta3 iw dev sta3-wlan0 link
Connected to 9e:0a:18:d8:bf:63 (on sta3-wlan0)
        SSID: ap3-ssid
        freq: 5180
        RX: 279839 bytes (5297 packets)
        TX: 4166068 bytes (2763 packets)
        signal: -86 dBm
        rx bitrate: 12.0 MBit/s
        tx bitrate: 12.0 MBit/s
```

Figura IV.32: Estado del enlace estación 3 escenario 1. Fuente: Autor.

Para simplificar los resultados obtenidos y mostrar todas las estaciones y como se manejan distintos recursos en cada una, en la tabla 4.3 se muestran los valores relevantes de lo obtenido anteriormente.

Estación	Frecuencia de Trabajo (GHz)	Velocidad de transmisión (Mbps)	Nivel de señal (dBm)
1	2.412	36	-79
2	2.437	18	-84
3	5.18	12	-86
4	2.412	24	-83
5	2.437	36	-77
6	5.18	24	-83

Tabla 4.3: Resultados obtenidos con condiciones iniciales escenario 1. Fuente: Autor.

De la tabla 4.3, se puede observar como cada estación trabaja con distintos recursos con respecto a las demás ya desde el inicio de su operación, esto hace que la red sea mucho mas dinámica, configurable y con un comportamiento adaptativo a diferencia de las redes convencionales.

El siguiente paso es realizar cambios en términos de potencia y velocidad de transmisión en las estaciones e identificar este efecto en métricas como lo son el retardo en el envío de paquetes y la tasa de pérdida de paquetes, para ello se utilizarán 3 estaciones, una por cada punto de acceso, y se comunicarán con estaciones de diferentes puntos de acceso para las diferentes pruebas. Las estaciones utilizadas para esta prueba serán la estación 4 del punto de acceso 1, la estación 2 del punto de acceso 2 y finalmente la estación 6 del punto de acceso 3. Para la estación 4 se variará únicamente la ganancia de la antena, para la estación 4 se disminuirá la velocidad de transmisión y para la estación 6 se variará tanto la ganancia de la antena como la velocidad de transmisión de tal forma que los nuevos valores quedarán expresados según la tabla 4.4.

Estación	Valores Iniciales			Nuevos Valores		
	Ganancia	Tasa de transmisión (Mbps)	Nivel de señal (dBm)	Ganancia	Tasa de transmisión (Mbps)	Nivel de señal (dBm)
2	5	18	-84	5	11	-84
4	5	24	-83	10	24	-78
6	5	24	-83	15	6	-73

Tabla 4.4: Nuevos valores para pruebas de pérdida de paquetes y retardos entre estaciones con múltiples puntos de acceso. Fuente: Autor.

La siguiente fase en este proceso de pruebas, luego de haber cambiado en las estaciones 2, 4, y 6 respectivamente parámetros en términos de potencia y tasa o velocidad de transmisión, es realizar pruebas con tráfico generado desde Mininet. Para esta prueba se comunicarán las estaciones 4 con la 3, la estación 2 con la estación 1 y finalmente la estación 6 con la estación 5 y se revisarán las métricas de tasa de pérdida de paquetes y retardos entre estaciones como se había mencionado anteriormente.

Los resultados que se verificarán a continuación serán los de los índices de pérdida de paquetes para cada caso, para el primer caso, los resultados se muestran en la Figura IV.33, obtenidos directamente de la terminal de esta estación.

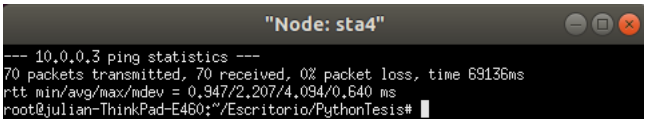


Figura IV.33: Tasa de pérdida de paquetes entre la estación 4 y la estación 3. Fuente: Autor.

En la Figura IV.34 se pueden apreciar los resultados del segundo caso obtenidos en la terminal de la estación sta2.

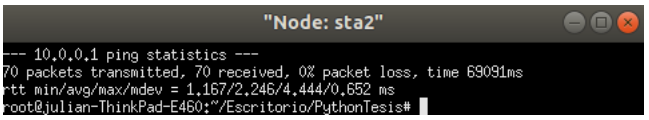


Figura IV.34: Tasa de pérdida de paquetes entre la estación 2 y la estación 1. Fuente: Autor.

Para el último caso, se muestran los resultados obtenidos en la terminal de la estación sta6, como se puede apreciar en la Figura IV.35.

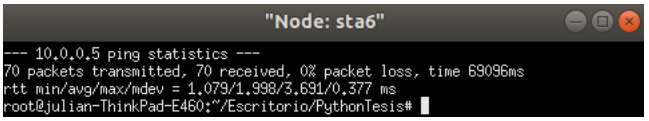


Figura IV.35: Tasa de pérdida de paquetes entre la estación 6 y la estación 5. Fuente: Autor.

Como se puede observar de los resultados obtenidos, en cada prueba se enviaron 70 paquetes y se obtuvieron índices de pérdida de paquetes del 0 %, lo que indica que todo funciona bien y que todavía la red es bastante simple por lo que es muy difícil tener pérdida de paquetes. Para la métrica de retardo, se utilizó la información brindada por la prueba anterior, y esta se organizó en la Figura IV.36.

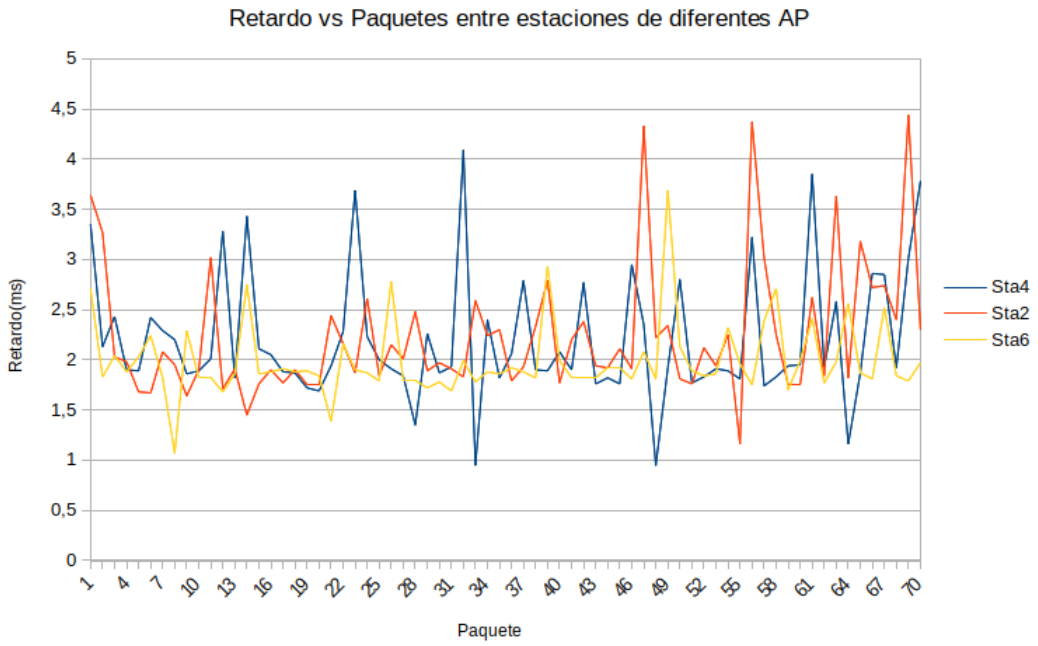


Figura IV.36: Retardo vs paquetes con cambios en términos de potencia y tasa de transmisión. Fuente: Autor.

En la Figura IV.36 hay varias cosas interesantes a resaltar, lo primero sería que el retardo está en términos de 2 a 4 milisegundos, lo cual es bueno debido a que para tecnologías de 5G se esperan retardos similares, lo segundo a resaltar es que la calidad en la potencia de la señal y la tasa de transmisión parecen tener una relación inversa con el retardo, es decir, que entre más bajo sea el tráfico en el enlace y la calidad de la señal parece provocar que hayan retardos más altos, como sucede con las estaciones 2 y 4 respectivamente, mientras que la estación 6 se mantiene dentro de un intervalo más pequeño y no presenta cambios instantáneos de valores elevados. En la Tabla 4.5 se muestran los valores de mayor relevancia en términos de retardo para cada estación a lo largo de la prueba.

Estación	**Retardo (ms)**		
	Mínimo	**Máximo**	**Promedio**
2	1.16	4.44	2.45
4	0.95	4.09	2.207
6	1.079	3.69	1.998

Tabla 4.5: Valores relevantes en la prueba de retardos entre estaciones. Fuente: Autor.

De la Tabla 4.5 podemos concluir que al utilizar controladores de capa cruzada se pueden lograr, hasta este punto, en emulaciones, retardos del orden de 1 ms lo cual es un gran avance en contraste a las redes convencionales de la actualidad.

6. Escenario 2: Handover

Para este escenario, con el objetivo de demostrar la funcionalidad y capacidad que podría tener un controlador de capa cruzada, se utilizó el concepto handover como base. El handover es bastante común en las telecomunicaciones móviles, este se define como la transición de una señal de usuario específica desde una estación móvil hacia otra geográficamente adyacente sin la desconexión de la llamada [22].

En el escenario 2 se mostrarán estaciones con movimiento que cambiarán su asociación a un punto de acceso dependiendo de varios factores, el primero será nivel de potencia (con el modelo mayor señal primero), y el segundo el balance de carga (con el modelo menor carga primero), lastimosamente no fue posible realizar pruebas con el segundo factor ya que presenta muchos errores dentro de Mininet-WiFi. El archivo utilizado para este escenario fue Handover.py de la sección de anexos.

En la Figura IV.37 se muestra la ubicación espacial de los puntos de acceso y estaciones utilizadas.

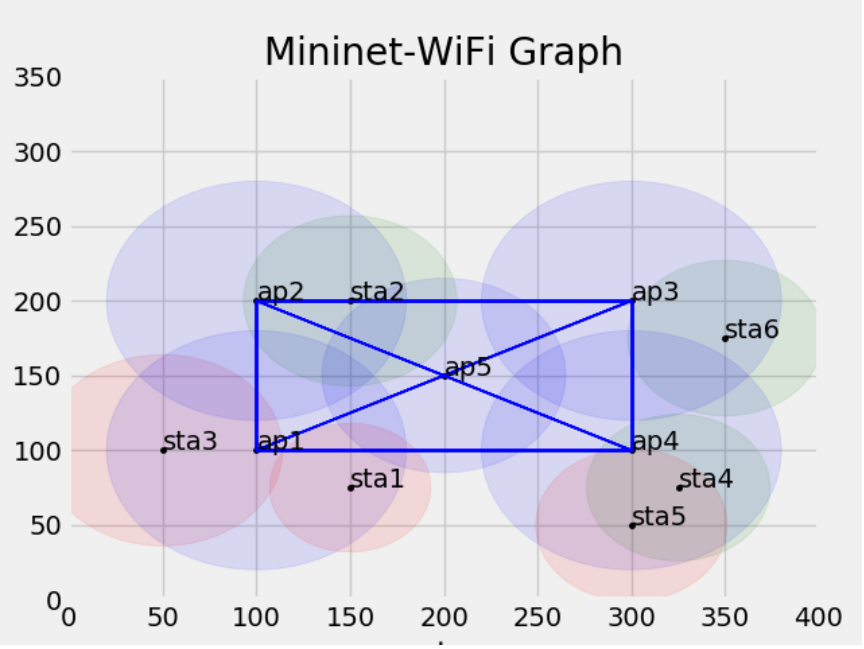


Figura IV.37: Escenario 2 ubicaciones iniciales. Fuente: Autor.

Como se mencionó anteriormente, las estaciones tendrán movimiento (para este caso en linea recta) a medida que transcurre el tiempo, hasta un punto donde se detiene la simulación, las ubicaciones finales de las estaciones móviles y los puntos de acceso se muestran en la Figura IV.38.

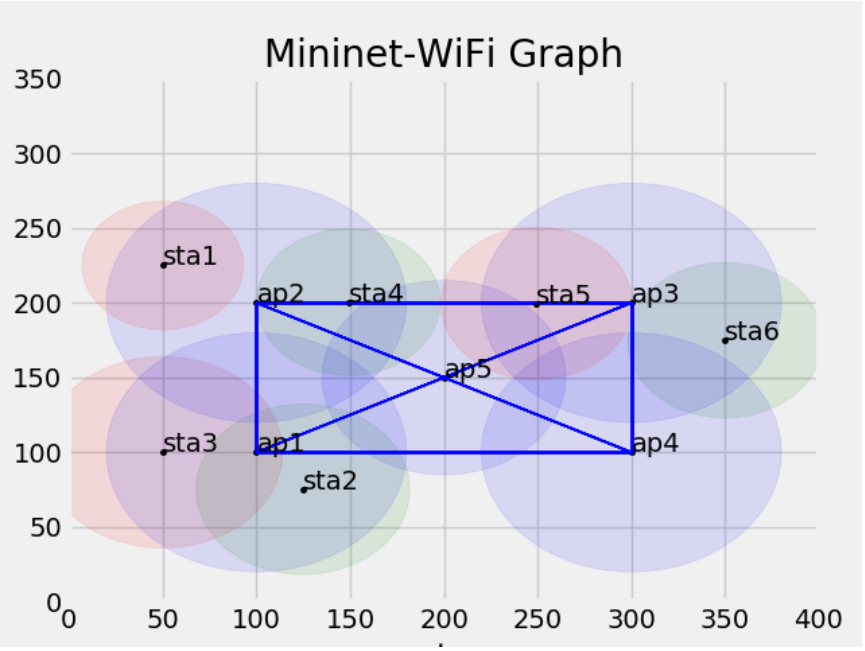


Figura IV.38: Escenario 2 ubicaciones finales. Fuente: Autor.

Las métricas para este escenario serán las mismas que las usadas anteriormente, sólo que se añadirá en la última parte de la experimentación tomas de distancias y niveles de señal para intentar descubrir la relación de estas con el handover de las estaciones móviles. Antes de profundizar en las pruebas, en la tabla 4.6 se muestran las distancias recorridas, el tiempo utilizado y la velocidad de cada estación con movimiento en el escenario handover, esto con el objetivo de averiguar qué efectos produce la velocidad de movimiento de las estaciones en las métricas.

Estación	Distancia(m)	Tiempo(s)	Velocidad(m/s)
1	180.28	10	18.03
2	127.48	12	10.62
4	215.06	22	9.82
5	158.11	22	7.19

Tabla 4.6: Velocidades de las estaciones móviles. Fuente: Autor.

6.1. Pruebas con tasa de pérdida de paquetes y retardos

Para estas pruebas se analizaron 3 diferentes situaciones que se pueden dar en una red con estaciones móviles: para la primera situación se usaron 2 estaciones las cuales van a diferente velocidad y están ubicadas espacialmente en diferentes puntos, estas estaciones son sta1 y sta2. Para la segunda situación, se escogió una estación estática con una en movimiento, estas son sta6 y sta5 respectivamente. Para la última situación posible en este escenario, se tiene una situación similar a la anterior, sólo que para este caso la estación en movimiento realiza handover 2 veces, es decir, cambia su asociación con un punto de acceso 2 veces, estas estaciones serán sta3 y sta4.

Los resultados obtenidos para las métricas utilizadas con las diferentes situaciones anteriormente mencionadas se muestran en la Tabla 4.7.

	Situación 1	**Situación 2**	**Situación 3**
#Paquetes enviados	50	50	50
#Paquetes recibidos	39	46	40
#Errores	9	3	10
Tasa de perdida de paquetes(%)	22	8	20
Retardo mínimo (ms)	1.494	0.684	0.873
Retardo promedio (ms)	82.28	91.331	28.8
Retardo máximo (ms)	2062.48	2053.89	1034.53

Tabla 4.7: Resultados comparativos para 3 diferentes situaciones de handover. Fuente: Autor.

Como se puede observar en la tabla 4.7, los resultados dejan de ser ideales, en comparación con el escenario 1 donde no había pérdida de paquetes y los retardos eran bastante bajos. En este escenario ya se cuentan con tasas de pérdida superiores al 20 % y retardos del orden de 2 segundos, lo cual es bastante malo ya que se deben siempre mantener retardos del orden de los milisegundos para no generar problemas en la comunicación entre estaciones y la calidad de servicio (QoS). De la tabla también se puede inferir que la velocidad de movimiento de las estaciones tiene una relación proporcional directa con el retardo, además, también se tiene una relación directa de la tasa de pérdida de paquetes con el número de traspasos que se produzcan en la comunicación.

6.2. Pruebas con la distancia y nivel de señal

En este punto se realizaron pruebas con el objetivo de obtener qué tipo de relación puede existir entre la distancia a la que se encuentra una estación al punto de acceso más cercano, su nivel de señal y finalmente cómo afecta todo esto en el handover de la estación. En estos escenarios todos los puntos de acceso poseen las mismas características en términos de ganancia y potencia de transmisión con el objetivo de tener claridad en los resultados. Para estas pruebas se utilizaron las situaciones 2 y 3 mencionadas anteriormente, esto debido a la facilidad de manejo de distancias al tener solo una estación en movimiento.

Para el cálculo de la distancia a los puntos de acceso asociados a lo largo de la emulación para la estación sta5 de la situación 2 se utilizó el archivo Distancia.py que se encuentra en la sección de anexos. En las pruebas con la situación 2 se tomó una muestra con 17 datos de la distancia a la que se encuentra el punto de acceso más cercano, cuál es el punto de acceso asociado y el nivel de señal que se posee.

En las Figuras IV.39 y IV.40 respectivamente, se muestran ejemplos de cómo se toman resultados posibles para la estación sta5 para los dos puntos de acceso a los que se asocia a lo largo de la emulación.

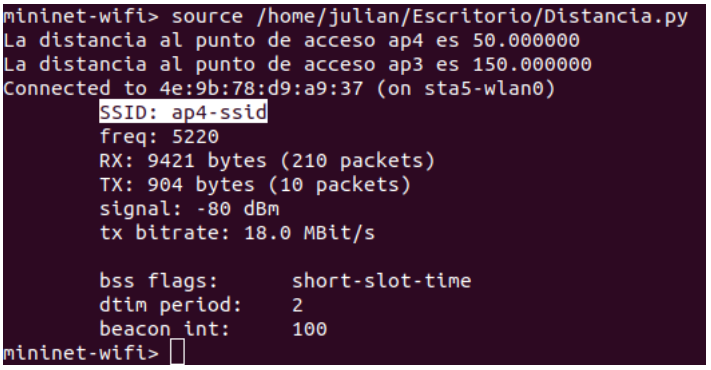


Figura IV.39: Sta5 asociada con ap4. Fuente: Autor.

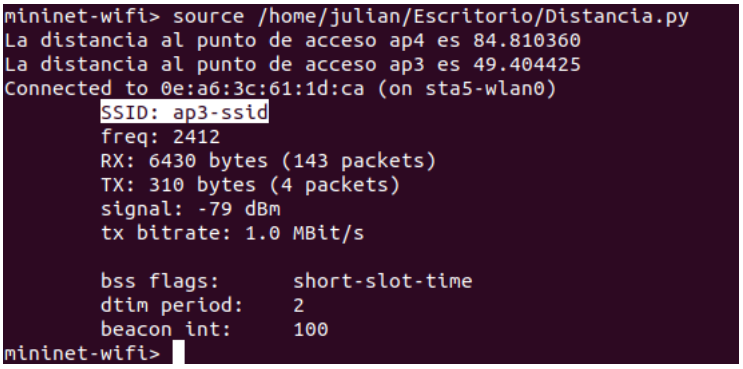


Figura IV.40: Sta5 asociada con ap3. Fuente: Autor.

Como se puede evidenciar de las Figuras IV.39 y IV.40, existe claramente un cambio en el punto de acceso asociado a la estación sta5 a lo largo de la emulación, además, se tiene conocimiento de la distancia a la que se encuentra la estación en cualquier momento, los valores obtenidos en esta prueba se muestran en la Tabla 4.8.

Dato	Nivel de señal(dBm)	Distancia ap4(m)	Distancia ap3(m)	AP asociado
1	-80	50	150	ap4
2	-76	41.27	141.19	ap4
3	-71	32.24	131.79	ap4
4	-66	26.83	125.79	ap4
5	-61	21.03	118.53	ap4
6	-55	16.59	110.07	ap4
7	-59	20.59	94.87	ap4
8	-69	30.42	83.98	ap4
9	-74	38.01	77.07	ap4
10	-78	45.97	70.55	ap4
11	-81	51.39	66.46	ap4
12	-83	58.25	61.72	ap4
13	—-	65.18	57.49	—-
14	-81	74.97	52.65	ap3
15	-79	84.81	49.40	ap3
16	-78	91.16	48.32	ap3
17	-79	111.72	50.60	ap3

Tabla 4.8: Datos tomados del recorrido de sta5 a lo largo de la emulación. Fuente: Autor.

Con los resultados mostrados en la Tabla 4.8, se pueden concluir varias cosas, lo primero es que el nivel de la señal tiene una relación estrecha con la distancia del punto de acceso a la que se encuentra asociada la estación móvil, lo segunda conclusión da en que las antenas utilizadas tienen una sensibilidad que ronda los -85 a -90 dBm y luego se pierde la conexión con el punto de acceso, y por último, que el algoritmo de señal fuerte primero no asegura que no haya pérdida de paquetes por lo que en trabajos futuros se puede encaminar en el desarrollo de algoritmos que minimicen la pérdida de paquetes al tener periodos de desconexión menores o nulos.

Para el cálculo de la distancia a los puntos de acceso asociados a lo largo de la emulación para la estación sta4 de la situación 3 se utilizó el archivo Distancia2.py que se encuentra en la sección de anexos. En las pruebas con la situación 3 se tomó una muestra con 24 datos de la distancia a la que se encuentra el punto de acceso más cercano, cuál es el punto de acceso asociado y el nivel de señal que se posee.

En las Figuras IV.41, IV.42 y IV.43 respectivamente, se muestran ejemplos de cómo se toman resultados posibles para la estación sta4 para los tres puntos de acceso a los que se asocia a lo largo de la emulación.

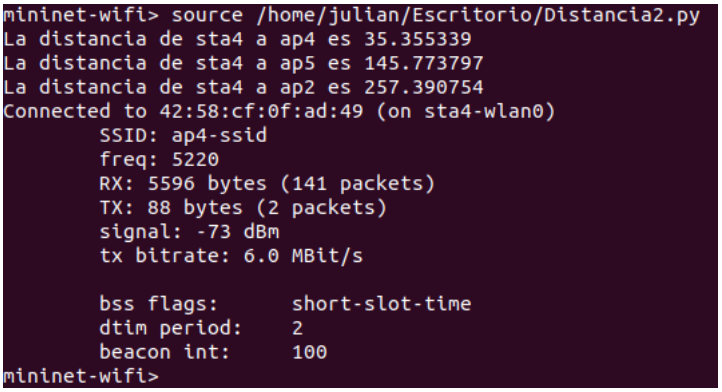


Figura IV.41: Sta4 asociada con ap4. Fuente: Autor.

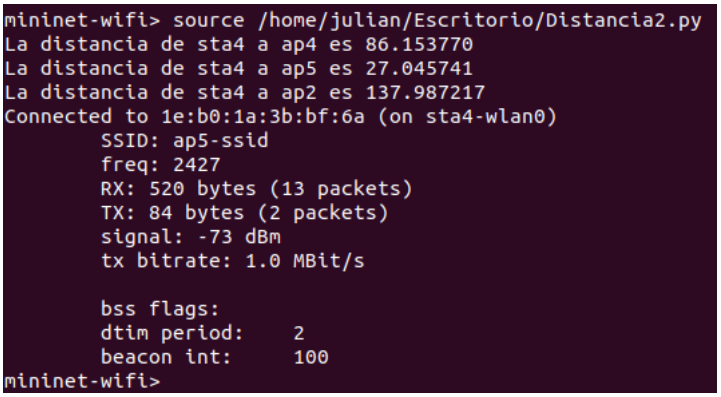


Figura IV.42: Sta4 asociada con ap5. Fuente: Autor.

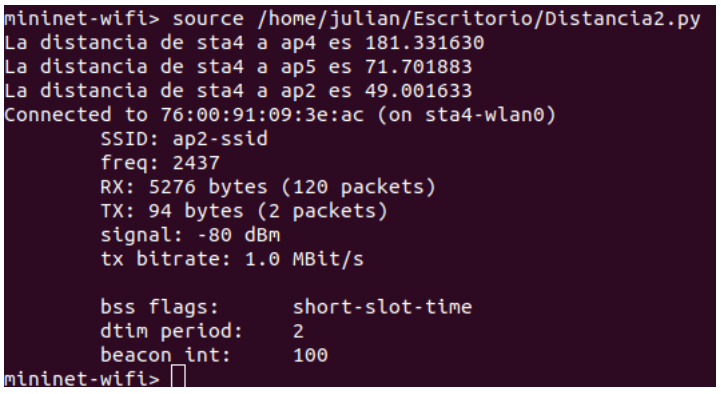


Figura IV.43: Sta4 asociada con ap2. Fuente: Autor.

Como se observa de las Figuras IV.41 , IV.42 y IV.43, para esta situación en la estación sta4 existen dos traspasos ya que se cambia el punto de acceso asociado 2 veces. Los resultados obtenidos para esta situación se muestran en la Tabla 4.9.

Dato	Nivel de señal(dBm)	Distancia ap4(m)	Distancia ap5(m)	Distancia ap2(m)	AP asociado
1	-73	35.36	145.77	257.39	ap4
2	-57	16.48	126.62	238.31	ap4
3	-29	5.56	108.46	220.22	ap4
4	-53	15.9	96.07	207.86	ap4
5	-64	24.30	87.51	199.31	ap4
6	-70	32.84	78.98	190.78	ap4
7	-75	40.48	71.43	183.21	ap4
8	-80	50.06	62.05	173.77	ap4
9	-83	57.74	54.61	166.22	ap4
10	—-	67.35	45.48	156.82	—-
11	-76	80.82	33.04	143.71	ap5
12	-70	92.37	23.41	132.53	ap5
13	-61	104.89	16.16	120.50	ap5
14	-60	114.52	15.94	111.33	ap5
15	-65	123.18	20.16	103.14	ap5
16	-71	131.85	26.64	95.04	ap5
17	-75	140.52	34.10	87.06	ap5
18	-80	148.23	41.14	80.08	ap5
19	-85	159.79	52.06	69.90	ap5
20	-89	170.39	62.27	61.01	ap5
21	—-	180.03	71.66	53.47	—-
22	-77	196.41	87.74	42.63	ap2
23	-74	206.05	97.24	38.16	ap2
24	-73	215.67	106.77	35.79	ap2

Tabla 4.9: Toma de datos para el recorrido de sta4 a lo largo de la emulación. Fuente: Autor.

De manera análoga a los resultados obtenidos de la Tabla 4.8, en la Tabla 4.9 se muestra como el nivel de señal tiene una relación directa con la distancia y existen periodos de desconexión mientras se implementa el algoritmo de handover. Por último, al tener condiciones tan **ideales** en las antenas transmisoras, se puede observar como se puede utilizar también la distancia a los puntos de acceso como un método para escoger el punto de acceso al cual asociar la estación móvil, esto para trabajos futuros.

7. Escenario 3: Ad-Hoc

Se consideró poner a prueba el funcionamiento del controlador de capa cruzada usando tecnología *Ad-Hoc*, teniendo en cuenta que éste es un tipo de red descentralizada en el cual no se ven involucrados puntos de acceso (AP), en cambio, cada nodo participa en el encaminamiento de los paquetes.

En las redes Ad-Hoc los nodos funcionan como clientes y enrutadores mesh, este tipo de red posee redundancia en cuanto a los caminos disponibles para llegar a destino, esto es gracias a que un nodo tiene contacto con varios nodos y en caso de que uno falle puede llegar a destino a través de otro.

Para comprobar el funcionamiento de la tecnología Ad-Hoc se planteó una red en forma de malla con nueve estaciones móviles, todos funcionando en modo Ad-Hoc ubicados

espacialmente como se muestra en la Figura IV.44, el protocolo de enrutamiento elegido es OLSR (Optimized Link State Routing), un protocolo proactivo, esto quiere decir que la información de la topología de red se intercambia de manera periódica, esto facilita las redes que tienen movilidad, el código usado para esta topología se encuentra en la sección de anexos como **Red_Ad-Hoc.py**.

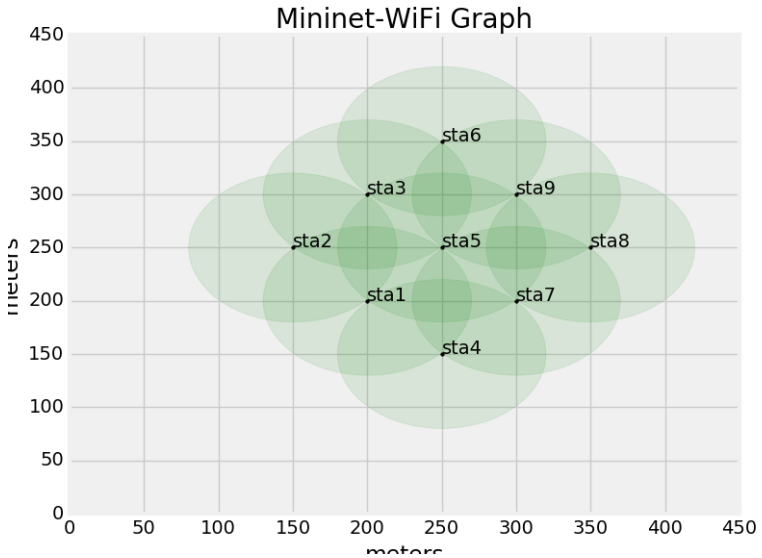


Figura IV.44: Topología de la red Ad-Hoc. Fuente:Autor

En la Figura IV.45 se puede apreciar la inicialización de las estaciones móviles bajo el protocolo OLSRD, para comprobar el funcionamiento de la red, se tomó como referencia la estación móvil 2 (sta2) ya que es una de las más retiradas y se envió una solicitud ICMP al resto de estaciones móviles, se realizaron 7 peticiones y se tomaron los valores máximos mínimos y promedio en cuanto a tiempo de respuesta, el resultado se encuentra evidenciado en la tabla 4.10.

```
*** Creating nodes
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Creating links
Starting olsrd in sta1-wlan0...
Starting olsrd in sta2-wlan0...
Starting olsrd in sta3-wlan0...
Starting olsrd in sta4-wlan0...
Starting olsrd in sta5-wlan0...
Starting olsrd in sta6-wlan0...
Starting olsrd in sta7-wlan0...
Starting olsrd in sta8-wlan0...
Starting olsrd in sta9-wlan0...
*** Starting network
sta1 sta2 sta3 sta4 sta5 sta6 sta7 sta8 sta9 *** Running CLI
*** Starting CLI:
mininet-wifi>
```

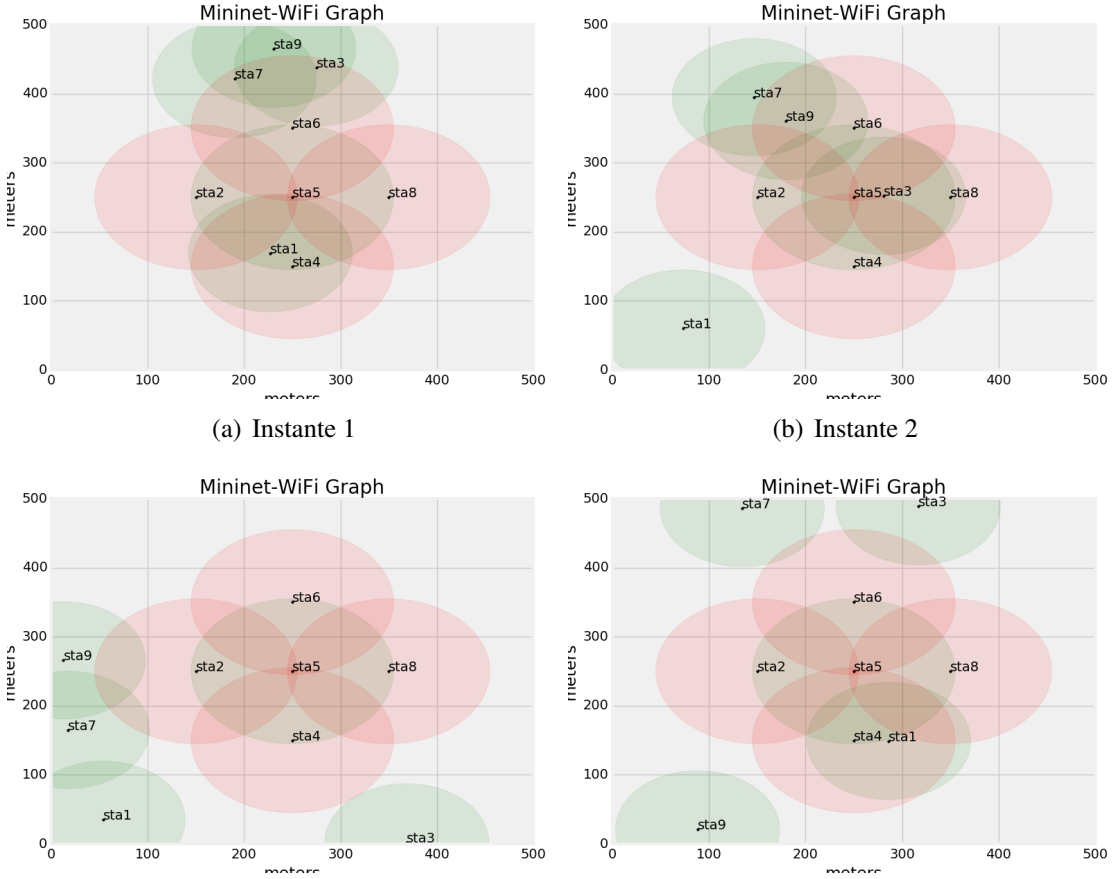
Figura IV.45: Inicialización protocolo OLSRD para las estaciones base. Fuente: Autor.

Como se muestra en la Tabla 4.10, la estación 2 presenta conexión con todas las otras estaciones, no se presentó pérdida de paquetes durante esta prueba, esto gracias a la continua conexión de los nodos y la falta de movilidad de los mismos.

Estación destino	**Tiempo min** $[ms]$	**Tiempo máx** $[ms]$	**Tiempo Avg** $[ms]$	**% de pérdida de paquetes**
sta1	0.500	1.128	0.218	0 %
sta3	0.823	9.309	2.119	0 %
sta4	0.748	8.345	1.992	0 %
sta5	0.707	6.702	1.671	0 %
sta6	0.630	6.912	2.116	0 %
sta7	0.832	1.628	1.080	0 %
sta8	0.493	0.842	0.704	0 %
sta9	0.858	1.154	0.944	0 %

Tabla 4.10: Resultados prueba ping estación móvil sta2. Fuente: Autor.

Usando como base los resultados de los escenarios planteados anteriormente de movilidad y handover, se modificó la topología de la Figura IV.44, como se aprecia en la Figura IV.46, se establecieron 5 estaciones sin movimiento: sta2, sta4, sta5, sta6 y sta8 (color rojo en la Figura IV.46), estas están dispuestas de tal manera que siempre se van a poder comunicar entre ellas, y las estaciones móviles: sta1,sta3,sta7 y sta9 (color verde en la figura IV.46) se moverán de manera aleatoria dentro del plano.



(a) Instante 1

(b) Instante 2

(c) Instante 3

(d) Instante 4

Figura IV.46: Red Ad-Hoc con movilidad y handover. Fuente: Autor.

En esta red el traspaso de comunicación o handover es realizado bajo el criterio de señal más fuerte (ssf) por sus siglas en inglés, para corroborar el funcionamiento del traspaso y el protocolo de enrutamiento proactivo se envían paquetes entre dos estaciones y se captura la ruta que toma cada uno, las estaciones base tienen asignada la dirección ip 10.0.0.x/8, en donde x depende del número de estación.

En el primer caso se envió un paquete de 60 bytes de la estación móvil sta7 (10.0.0.7) a la estación sta4 (10.0.0.4) como se puede ver en la Figura IV.47, en la primera solicitud, el paquete llegó a su destino después de ser enrutado por las estaciones sta9 (10.0.0.9), sta5 (10.0.0.5) y sta6 (10.0.0.6), el tiempo total del envío del paquete fue de 111.599 ms; en la segunda solicitud el paquete fue enrutado a través de las estaciones sta2 (10.0.0.2) y sta5 (10.0.0.5), en este caso el tiempo de envío fue de 49.244 ms; para la tercera solicitud no se pudo establecer la conexión a través de ningún nodo, el resumen de esta prueba se encuentra en la tabla 4.11.

```
mininet-wifi> sta7 traceroute sta4
traceroute to 10.0.0.4 (10.0.0.4), 30 hops max, 60 byte packets
 1  10.0.0.9 (10.0.0.9)  5.244 ms  20.006 ms  21.317 ms
 2  10.0.0.6 (10.0.0.6)  31.339 ms  50.265 ms  50.797 ms
 3  10.0.0.5 (10.0.0.5)  63.273 ms  63.790 ms  64.383 ms
 4  10.0.0.4 (10.0.0.4)  76.868 ms  77.134 ms  111.599 ms
mininet-wifi> sta7 traceroute sta4
traceroute to 10.0.0.4 (10.0.0.4), 30 hops max, 60 byte packets
 1  10.0.0.2 (10.0.0.2)  5.358 ms  19.482 ms  20.764 ms
 2  10.0.0.5 (10.0.0.5)  34.226 ms  34.715 ms  36.040 ms
 3  10.0.0.4 (10.0.0.4)  46.383 ms  47.245 ms  49.244 ms
mininet-wifi> sta7 traceroute sta4
traceroute to 10.0.0.4 (10.0.0.4), 30 hops max, 60 byte packets
 1  10.0.0.7 (10.0.0.7)  3055.554 ms !H  3055.469 ms !H  3055.455 ms !
```

Figura IV.47: Prueba comunicación sta7-sta4. Fuente: Autor.

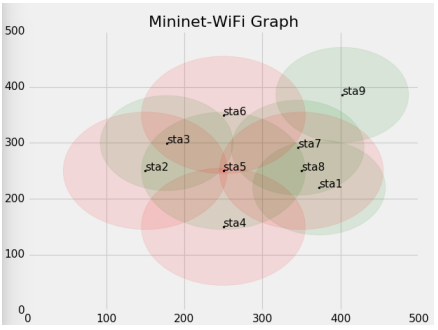
Fuente	**Ruta**	**Destino**	**Tiempo de envío** $[ms]$
sta7	sta7-sta9-sta6-sta5-sta4	sta4	111.599
sta7	sta7-sta5-sta2-sta4	sta4	49.244
sta7	sta7-inalcanzable	sta4	3055.455

Tabla 4.11: Resumen prueba comunicación sta7-sta4. Fuente: Autor.

En el segundo caso se envió un paquete de las mismas características desde una estación móvil sta7 (10.0.0.7) a otra estación móvil sta3 (10.0.0.3) como se puede apreciar en la Figura IV.48 (a), en este caso, en los dos primeros intentos de comunicación no se pudo realizar la conexión, sin embargo, en el tercer intento el paquete fue enrutado a través de las estaciones sta2 (10.0.0.2), sta5 (10.0.0.5) y sta4 (10.0.0.4), el tiempo de envío fue de 70.634 ms, en la última prueba el paquete fue enrutado a través de la estación sta3 (10.0.0.3) con un tiempo de envío de 66.379 ms, en la Figura IV.48(b) se puede apreciar la ubicación espacial de las estaciones al momento de realizar la última prueba.

```
mininet-wifi> sta7 traceroute sta3
traceroute to 10.0.0.3 (10.0.0.3), 30 hops max, 60 byte packets
 1  10.0.0.7 (10.0.0.7)  3080.914 ms !H  3080.829 ms !H  3080.816 ms !
H
mininet-wifi> sta7 traceroute sta3
traceroute to 10.0.0.3 (10.0.0.3), 30 hops max, 60 byte packets
 1  10.0.0.7 (10.0.0.7)  3066.511 ms !H  3066.451 ms !H  3066.436 ms !
H
mininet-wifi> sta7 traceroute sta3
traceroute to 10.0.0.3 (10.0.0.3), 30 hops max, 60 byte packets
 1  10.0.0.2 (10.0.0.2)  1.467 ms  26.464 ms  26.758 ms
 2  10.0.0.5 (10.0.0.5)  34.219 ms  36.870 ms  38.061 ms
 3  10.0.0.4 (10.0.0.4)  45.471 ms  46.188 ms  47.022 ms
 4  10.0.0.3 (10.0.0.3)  68.663 ms  69.186 ms  70.634 ms
mininet-wifi> sta7 traceroute sta3
traceroute to 10.0.0.3 (10.0.0.3), 30 hops max, 60 byte packets
 1  10.0.0.5 (10.0.0.5)  3.876 ms  38.002 ms  39.323 ms
 2  10.0.0.3 (10.0.0.3)  46.881 ms  65.295 ms  66.379 ms
```

(a) Ruta paquete sta7-sta3



(b) ubicación estaciones última conexión

Figura IV.48: Envío de paquete de estación móvil a estación móvil. Fuente: Autor

En esta sección no se analizaron las pérdidas en paquetes ocasionadas por el traspaso o handover ni los tiempos de respuesta de las peticiones ICMP ya que se estudió en la sección de escenario handover.

8. Escenario 4: Red mixta SDN y Ad-Hoc

Como último escenario se planteó combinar los resultados de la red descentralizada con estaciones base de tecnología Ad-Hoc y puntos de acceso de tipo OpenVSwitch funcionando con el controlador de capa cruzada usado en los anteriores escenarios, esto con el fin de garantizar la conexión de las estaciones móviles el mayor tiempo posible.

Esta red funciona bajo el principio de que si una estación móvil pierde conexión con su punto de acceso otra estación sea capaz de prestarle servicios, para la implementación de esta red se planteó una topología similar a la del escenario 2, 5 puntos de acceso conectados entre si y 7 estaciones base, como se puede ver en la figura IV.49, el código usado para esta topología se encuentra en la sección de anexos como **RedMixta.py**.

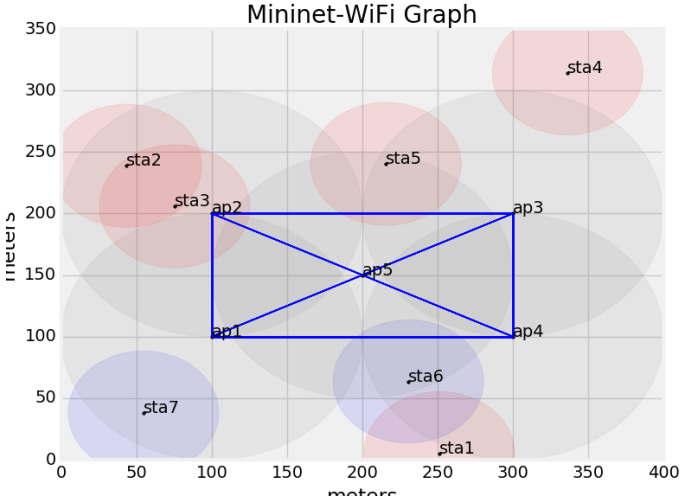


Figura IV.49: Topología red mixta. Fuente: Autor.

Ya que las estaciones base están en movimiento de manera aleatoria deben ser capaces de conectarse a cualquier punto de acceso, además de esto, para garantizar la calidad de servicio, se usó handover bajo el criterio de señal más fuerte (ssf o strongest signal first por sus siglas en inglés), esto quiere decir que la estación base se conectará al punto de acceso del cual reciba una mayor potencia en la señal.

Para lograr el correcto funcionamiento de las estaciones móviles, fue necesario habilitar dos tarjetas de red WiFi, una configurada en modo Ad-Hoc para poder garantizar la conexión entre estaciones, y la otra tarjeta fue configurada para conectarse a los puntos de acceso, al usar el comando **iwconfig** se pueden verificar las conexiones que posee cada estación móvil, en la figura IV.50 se puede apreciar que la estación base sta1 está conectada con una tarjeta de red al punto de acceso Ap1 (ssid_1) a través de la interfaz sta1-wlan1, además se puede ver la información del enlace en el momento de realizar el test.

```
mininet-wifi> sta1 iwconfig
sta1-wlan0  IEEE 802.11  ESSID:"ssid_1"
          Mode:Managed  Frequency:2.432 GHz  Access Point: 02:00:00:00:0C:00
          Bit Rate:6 Mb/s   Tx-Power=1 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=32/70  Signal level=-78 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:2   Missed beacon:0

lo        no wireless extensions.

sta1-wlan1  IEEE 802.11  ESSID:"adhocNet"
          Mode:Ad-Hoc  Frequency:2.432 GHz  Cell: 02:CA:FF:EE:BA:01
          Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

Figura IV.50: Configuración tarjetas de red. Fuente: Autor.

Se realizó esta misma prueba a dos estaciones diferentes, en la figura IV.51(a) se puede apreciar que la estación móvil sta5 al momento de realizar la prueba tiene activa la interfaz sta5-wlan1 configurada en modo Ad-hoc, sin embargo la interfaz sta5-wlan0 no tiene ningún punto de acceso asociado, por otro lado se encuentra la estación sta7 (como se puede ver en la figura IV.51(b)), esta estación móvil está configurada bajo la premisa de sólo poseer una tarjeta de red, destinada a conectarse a la red mediante un punto de acceso (no puede hacer uso de los servicios de la red Ad-Hoc), en este caso la estación sta7 está conectada al punto de acceso ap2 (ssid_2), también se puede apreciar en la imagen que al tener una calidad de enlace más baja que la mostrada para la estación sta1 en la figura IV.50 debido a un nivel de potencia menor en la señal, la tasa de transmisión se ve penalizada.

```
mininet-wifi> sta5 iwconfig
sta5-wlan1  IEEE 802.11  ESSID:"adhocNet"
          Mode:Ad-Hoc  Frequency:2.432 GHz  Cell: 02:CA:FF:EE:BA:01
          Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

sta5-wlan0  IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

lo        no wireless extensions.
```

(a) Conexiones de red sta5

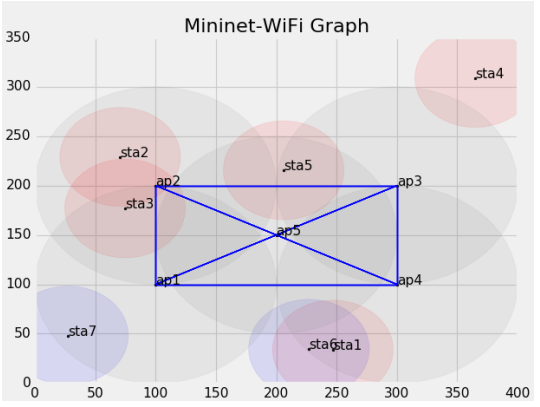
```
mininet-wifi> sta7 iwconfig
sta7-wlan0  IEEE 802.11  ESSID:"ssid_2"
          Mode:Managed  Frequency:2.432 GHz  Access Point: 02:00:00:00:0D:00
          Bit Rate:1 Mb/s   Tx-Power=1 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=21/70  Signal level=-89 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

lo        no wireless extensions.
```

(b) Conexiones de red sta7

Figura IV.51: Estado de conexión de red sta5 y sta7. Fuente: Autor.

Se realizó una prueba de conexión a través de solicitudes ICMP de una estación móvil (sta6) a una de las estaciones más retiradas en ese momento pero dentro del rango de cobertura de un punto de acceso (ap2), los resultados se pueden apreciar en la figura IV.52, en donde en la figura IV.52(a) se puede ver la ubicación espacial de todos los nodos de la red y en la figura IV.52(b) el resultado del ping entre estaciones sta6 y sta2.



(a) Ubicación de nodos de la red

```
mininet-wifi> sta6 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=3.72 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=7.57 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=5.76 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=2.98 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=3.84 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=3.28 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=4.34 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=3.21 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=4.09 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=3.74 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=9.86 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=3.54 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=3.93 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=3.80 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=5.03 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=4.20 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=4.34 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=3.38 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=3.12 ms
^C
--- 10.0.0.2 ping statistics ---
19 packets transmitted, 19 received, 0% packet loss, time 18022ms
rtt min/avg/max/mdev = 2.984/4.410/9.869/1.658 ms
mininet-wifi>
```

(b) Prueba ICMP sta6-sta2

Figura IV.52: Prueba de conexión entre estaciones. Fuente: Autor.

Para verificar el correcto funcionamiento de la red en modo Ad-Hoc se estableció fija la ubicación de 3 estaciones, sta1, sta3 y sta6. Se ubicaron de tal manera que sta3 estuviera en el rango de un punto de acceso (ap3) pero sta3 y sta6 no, como se puede ver en la

figura IV.53, en este caso las estaciones sta3 y sta1 poseen dos tarjetas de red inalámbrica y una de ellas está configurada en modo Ad-Hoc, por ende sta3 podrá brindarle servicio de conexión, mientras que sta6 no posee esta característica y aunque se encuentra en el rango de la estación sta3 no le es posible conectarse a la red.

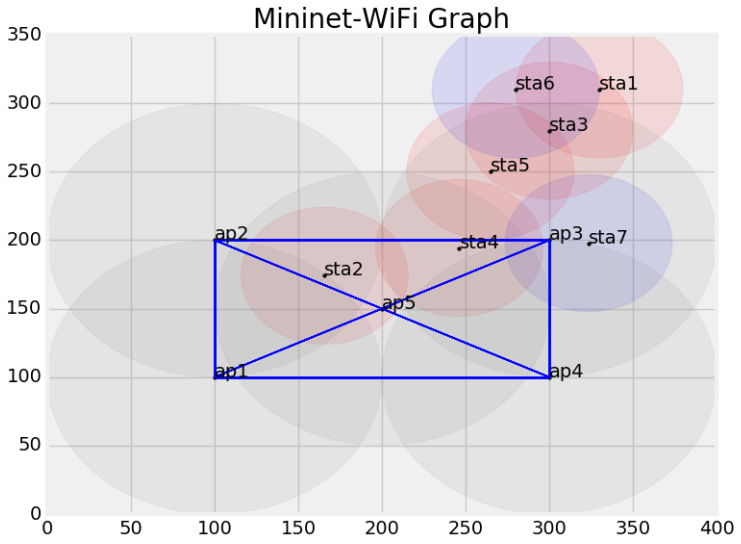


Figura IV.53: Topología de red mixta para prueba de conexión. Fuente: Autor.

Se comprobaron las conexiones de red de las estaciones sta1 y sta6 usando el comando **iwconfig**, el resultado se puede apreciar en la figura IV.54, en este caso se puede ver que efectivamente la estación sta1 posee dos tarjetas de red y una de ellas está configurada para hacer parte de una red Ad-Hoc (figura IV.54(a)), sin embargo, no está asociada a ningún punto de acceso, por otro lado, en la figura IV.54(b) se puede ver que la estación sta6 sólo posee una tarjeta de red y esta no se encuentra asociada a ningún punto de acceso.

```
mininet-wifi> sta1 iwconfig
lo        no wireless extensions.

sta1-wlan1  IEEE 802.11  ESSID:"adhocNet"
          Mode:Ad-Hoc  Frequency:2.432 GHz  Cell: 02:CA:FF:EE:BA:01
          Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

sta1-wlan0  IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

(a) Tarjetas de red sta1

```
mininet-wifi> sta6 iwconfig
lo        no wireless extensions.

sta6-wlan0  IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

(b) Tarjeta de red sta6

Figura IV.54: Estado de tarjetas de red estaciones sta1 y sta6. Fuente: Autor.

Para verificar el funcionamiento de esta red, se envió un mensaje ICMP desde las estaciones sta6 y sta1 hacia la estación sta3 que es la más cercana y la que les podría brindar servicios, en la Figura IV.55 se encuentra el resultado de esta prueba, se puede ver en la Figura IV.55(b) que hay conexión entre la estación sta3 y sta1, sin embargo, como se ve en la Figura IV.55(b) no es posible conectarse a la estación sta6, se obtuvo el resultado esperado ya que como se mencionó anteriormente las estaciones que no dispongan de una tarjeta de red adicional no podrán funcionar en modo Ad-Hoc.

```
mininet-wifi> sta1 ping -c 4 sta3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.485 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.576 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.436 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.904 ms

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.436/0.600/0.904/0.183 ms
```

(a) ping entre sta1 y sta3

```
mininet-wifi> sta6 ping -c 4 sta3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.6 icmp_seq=1 Destination Host Unreachable
From 10.0.0.6 icmp_seq=2 Destination Host Unreachable
From 10.0.0.6 icmp_seq=3 Destination Host Unreachable
From 10.0.0.6 icmp_seq=4 Destination Host Unreachable

--- 10.0.0.3 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3065ms
pipe 4
```

(b) ping entre sta6 y sta3

Figura IV.55: Prueba de mensajes ICMP a estación base dentro de la red SDN. Fuente: Autor.

En la Figura IV.56 se puede ver el resultado de hacer varias solicitudes ICMP desde la estación sta1 que se encuentra fuera de rango de los puntos de acceso de la red SDN, la primer prueba se realiza desde la estación sta1 a la estación sta5 que se encuentra asociada al nodo ap3 (como se puede ver en la figura IV.57 (a)), en este caso se logró respuesta en las 3 solicitudes realizadas, en el segundo caso se realizó el ping a la estación sta2, esta estación se encuentra asociada al punto de acceso ap5 como se puede ver en la figura IV.57 (b), en este caso al realizar la primer solicitud hubo un porcentaje de pérdida de paquetes del 66 % (dos paquetes) sin embargo, al realizar la prueba de nuevo todas las solicitudes fueron exitosas, esto debido a que en la primera solicitud se establecieron las tablas de enrutamiento en el controlador.

```
mininet-wifi> sta1 ping -c 3  sta5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=0.531 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.589 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.694 ms

--- 10.0.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.531/0.604/0.694/0.073 ms
mininet-wifi> sta1 ping -c 3  sta2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=219 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 1 received, 66% packet loss, time 2025ms
rtt min/avg/max/mdev = 219.815/219.815/219.815/0.000 ms
mininet-wifi> sta1 ping -c 3  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=0.524 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=10.2 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.602 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.524/3.788/10.239/4.561 ms
```

Figura IV.56: Solicitudes ICMP (ping) con estaciones asociadas a diferentes puntos de acceso. Fuente: Autor.

Además en la Figura IV.57 se puede ver el estado de enlace la estación móvil y su respectivo punto de acceso, se puede ver que la calidad del link no es la óptima, sin embargo, se podría mejorar cambiando los parámetros de las estaciones base como se hizo en este trabajo en capítulos anteriores.

```
mininet-wifi> sta5 iwconfig
sta5-wlan1  IEEE 802.11  ESSID:"adhocNet"
          Mode:Ad-Hoc  Frequency:2.432 GHz  Cell: 02:CA:FF:EE:BA:01
          Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

lo        no wireless extensions.

sta5-wlan0  IEEE 802.11  ESSID:"ssid_3"
          Mode:Managed  Frequency:2.432 GHz  Access Point: 02:00:00:00:0E:00
          Bit Rate:1 Mb/s   Tx-Power=1 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=27/70  Signal level=-83 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

(a) Conexiones de red sta5

```
mininet-wifi> sta2 iwconfig
sta2-wlan1  IEEE 802.11  ESSID:"adhocNet"
          Mode:Ad-Hoc  Frequency:2.432 GHz  Cell: 02:CA:FF:EE:BA:01
          Tx-Power=15 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

lo        no wireless extensions.

sta2-wlan0  IEEE 802.11  ESSID:"ssid_5"
          Mode:Managed  Frequency:2.432 GHz  Access Point: 02:00:00:00:10:00
          Bit Rate:1 Mb/s   Tx-Power=1 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=33/70  Signal level=-77 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:3   Missed beacon:0
```

(b) Conexiones de red sta2

Figura IV.57: Puntos de acceso asociados a las estaciones sta5 y sta2. Fuente: Autor.

Capítulo V

Conclusiones

- Como alternativa para las comunicaciones de 5G, la implementación de un controlador SDN/SDR puede ser una alternativa bastante beneficiosa debido a la manera en que ofrece el manejo de recursos de forma dinámica.
- Los principales beneficios que podrá ofrecer en la parte física el controlador SDN/SDR en comparación con tecnologías existentes serán la disminución de costos y del área requerida para la implementación de un sistema de telecomunicaciones móviles.
- Aunque lo realizado fue una emulación de lo que sería la implementación de un controlador SDN/SDR, se puede evidenciar que la implementación de este controlador es factible, y seguramente en unos años posiblemente será una tecnología ampliamente utilizada en las telecomunicaciones.
- En cuanto a indicadores, se puede decir que en las situaciones donde las estaciones son estáticas, estos superan a los que se obtienen normalmente en las tecnologías existentes, además, dichas cifras se adecuan a las proyecciones que se tienen para lo que serán las redes 5G, como se puede observar en [23].
- Aunque todavía queda bastante para lo que será la implementación total de un controlador SDN/SDR, con este trabajo se dan unas nociones de su comportamiento en diferentes situaciones, para así poder determinar cuáles serían las ventajas y las desventajas de su implementación en redes 5G.
- El controlador ofrece buenos resultados en cuanto a la métrica de retardos, obteniendo en el mejor de los casos retardos del orden de 0.6ms como se puede observar en la tabla 4.7, pero muestra resultados deficientes en la tasa de pérdida de paquetes, en especial al momento de hacer handover, donde se obtuvieron pérdidas de hasta el 22 % como se muestra en la tabla 4.7, por lo que será necesario para trabajos futuros idear técnicas que mejoren esta métrica.
- Se logró combinar la red SDN basada en el controlador de capa cruzada con estaciones base con tecnología Ad-Hoc, sin embargo al funcionar bajo diferentes protocolos de enrutamiento se pueden presentar incompatibilidades, además se aumenta el tiempo de retardo total y el flujo de paquetes al OLSR ser un protocolo reactivo,

sin embargo, se garantiza una mayor área de cobertura, como trabajo futuro se sugiere probar el funcionamiento de esta red mixta con un protocolo de enrutamiento en los host Ad-Hoc distinto.

- Se logró integrar la tecnología SDN y SDR en una emulación de red inalámbrica funcionando bajo el protocolo IEEE 802.11, sin embargo, los conceptos tratados en este trabajo pueden ser aplicados a distintas redes que requieran conexión inalámbrica cómo la telefonía móvil.

Anexos

1. Red2WE.py

```
import sys
from mininet.node import OVSSwitch,OVSKernelSwitch,RemoteController,OVSController
from mininet.log import setLogLevel, info
from mininet.topolib import TreeTopo
from mn_wifi.net import Mininet_wifi
from mn_wifi.node import Station, OVSKernelAP
from mn_wifi.cli import CLI
from mn_wifi.link import wmediumd
from mn_wifi.wmediumdConnector import interference
from subprocess import call
from mininet.link import TCLink, Intf
from mn_wifi.replaying import ReplayingBandwidth
from mn_wifi.wmediumdConnector import interference


modo=raw_input('Ingrese el modo de Trabajo de a,b,g,n: ')
canal=int(raw_input('Ingrese el canal de Trabajo: '))
#print('Si desea cambiar el modo o el canal del access point debe reiniciar la simulación.')

rangoAp1=int (raw_input('Ingrese el rango de trabajo del access point 1: '))
rangoSta1=int (raw_input('Ingrese el rango de trabajo de la estacion  1: '))
rangoSta2=int (raw_input('Ingrese el rango de trabajo de la estacion  2: '))
rangoSta3=int (raw_input('Ingrese el rango de trabajo de la estacion  3: '))
rangoSta4=int (raw_input('Ingrese el rango de trabajo de la estacion  4: '))


def myNetwork():

    net = Mininet_wifi(topo=None,
                       build=False,
                       link=wmediumd,
                       wmediumd_mode=interference,
                       ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0 = net.addController(name='c0',
                           controller=RemoteController,
                           ip='127.0.0.1',
                           protocol='tcp',
                           port=6653)

    info( '*** Add switches/APs\n')


    ap1 = net.addAccessPoint('ap1', cls=OVSKernelAP, ssid='ap1-ssid',
                             channel=canal,mode=modo,position='500.0,500.0,0',
range=rangoAp1,protocols='OpenFlow13')

    info( '*** Add hosts/stations\n')
    sta4 = net.addStation('sta4', ip='10.0.0.4',channel=canal,mode=modo,
```

```
                            position='700.0,500.0,0',range=rangoSta4)
    sta1 = net.addStation('sta1', ip='10.0.0.1',channel=canal,mode=modo,
                            position='500.0,700.0,0',range=rangoSta1)
    sta3 = net.addStation('sta3', ip='10.0.0.3',channel=canal,mode=modo,
                            position='500.0,300.0,0',range=rangoSta3)
    sta2 = net.addStation('sta2', ip='10.0.0.2',channel=canal,mode=modo,
                            position='300.0,500.0,0',range=rangoSta2)

    info("*** Configuring Propagation Model\n")

    net.setPropagationModel(model="friis", sL = 2)

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info( '*** Add links\n')
    net.addLink(ap1,sta2)
    net.addLink(ap1,sta3)
    net.addLink(ap1,sta4)
    net.addLink( ap1, sta1)


    net.plotGraph(max_x=1000,max_y=1000)




    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches/APs\n')
    net.get('ap1').start([c0])

    info( '*** Post configure nodes\n')

    CLI(net)
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

2. Rango.py

```
px nodo1=raw_input('ingrese el nodo que desea modificar su rango: ')
px rango=raw_input('valor de rango nuevo: ')
py net.get(nodo1).setRange(rango)
```

3. GananciaAntena.py

```
px print('modificar la Ganancia de antena')
px nodo2= raw_input('Ingrese el nodo a modificar la Ganancia de antena: ')
px print('la ganancia de antena en el nodo '+nodo2+' es:')
py net.get(nodo2).wintfs[0].AntennaGain
px #print('información del enlace:')
px #nodo2 iw dev nodo2-wlan0 link
px pot= input('Ingrese la nueva ganancia en antena: ')
py net.get(nodo2).setAntennaGain(pot)
px #print('información del enlace:')
px #nodo2 iw dev nodo2-wlan0 link
```

4. PotenciaTx.py

```
px print('modificar potencia de transmisión')
px nodo2= raw_input('Ingrese el nodo a modificar la potencia de transmisión: ')
px print('la potencia de transmisión en el nodo '+nodo2+' es:')
py net.get(nodo2).wintfs[0].txpower
px #print('información del enlace:')
px #nodo2 iw dev nodo2-wlan0 link
px pot= input('Ingrese la nueva potencia de transmisión: ')
py net.get(nodo2).setTxPower(pot)
px #print('información del enlace:')
px #nodo2 iw dev nodo2-wlan0 link
```

5. Escenario1E.py

```
import sys
from mininet.node import OVSSwitch,OVSKernelSwitch,RemoteController,OVSController
from mininet.log import setLogLevel, info
from mininet.topolib import TreeTopo
from mn_wifi.net import Mininet_wifi
from mn_wifi.node import Station, OVSKernelAP
from mn_wifi.cli import CLI
from mn_wifi.link import wmediumd
from mn_wifi.wmediumdConnector import interference
from subprocess import call
from mininet.link import TCLink, Intf
from mn_wifi.replaying import ReplayingBandwidth
from mn_wifi.wmediumdConnector import interference


modo=raw_input('Ingrese el modo de Trabajo de ap1 a,b,g,n: ')
canal=int(raw_input('Ingrese el canal de Trabajo de ap1: '))
modo2=raw_input('Ingrese el modo de Trabajo de ap2 a,b,g,n: ')
canal2=int(raw_input('Ingrese el canal de Trabajo de ap2: '))
modo3=raw_input('Ingrese el modo de Trabajo de ap3 a,b,g,n: ')
canal3=int(raw_input('Ingrese el canal de Trabajo ap3: '))
```

```
print('Si desea cambiar el modo o el canal del access point debe reiniciar la simulación.')

rangoAp1=int (raw_input('Ingrese el rango de trabajo del access point 1: '))
rangoAp2=int (raw_input('Ingrese el rango de trabajo del access point 2: '))
rangoAp3=int (raw_input('Ingrese el rango de trabajo del access point 3: '))
rangoSta1=int (raw_input('Ingrese el rango de trabajo de la estacion  1: '))
rangoSta2=int (raw_input('Ingrese el rango de trabajo de la estacion  2: '))
rangoSta3=int (raw_input('Ingrese el rango de trabajo de la estacion  3: '))
rangoSta4=int (raw_input('Ingrese el rango de trabajo de la estacion  4: '))
rangoSta5=int (raw_input('Ingrese el rango de trabajo de la estacion  5: '))
rangoSta6=int (raw_input('Ingrese el rango de trabajo de la estacion  6: '))



def myNetwork():

    net = Mininet_wifi(topo=None,
                       build=False,
                       link=wmediumd,
                       wmediumd_mode=interference,
                       ipBase='10.0.0.0/8')



    info( '*** Adding controller\n' )
    c0 = net.addController(name='c0',
                           controller=RemoteController,
                           ip='127.0.0.1',
                           protocol='tcp',
                           port=6653)

    info( '*** Add switches/APs\n')
    ap1 = net.addAccessPoint('ap1', cls=OVSKernelAP, ssid='ap1-ssid',
                             channel=canal,mode=modo, position='500.0,500.0,0',
range=rangoAp1,protocols='OpenFlow13')


    ap2 = net.addAccessPoint('ap2', cls=OVSKernelAP, ssid='ap2-ssid',
                            channel=canal2,mode=modo2, position='1000.0,1750.0,0',
range=rangoAp2,protocols='OpenFlow13')


    ap3 = net.addAccessPoint('ap3', cls=OVSKernelAP, ssid='ap3-ssid',
                             channel=canal3,mode=modo3, position='1750.0,500.0,0',
range=rangoAp3,protocols='OpenFlow13')



    info( '*** Add hosts/stations\n')
    sta1 = net.addStation('sta1', ip='10.0.0.1',channel=canal,mode=modo,
                          position='420.0,420.0,0',range=rangoSta1)
    sta3 = net.addStation('sta3', ip='10.0.0.3',channel=canal3,mode=modo3,
                          position='1830.0,300.0,0',range=rangoSta3)
    sta2 = net.addStation('sta2', ip='10.0.0.2',channel=canal2,mode=modo2,
                          position='1200.0,1740.0,0',range=rangoSta2)

    sta4 = net.addStation('sta4', ip='10.0.0.4',channel=canal,mode=modo,
                          position='600.0,620.0,0',range=rangoSta4)
    sta5 = net.addStation('sta5', ip='10.0.0.5',channel=canal2,mode=modo2,
                          position='930.0,1650.0,0',range=rangoSta5)
    sta6 = net.addStation('sta6', ip='10.0.0.6',channel=canal3,mode=modo3,
                          position='1620.0,600.0,0',range=rangoSta6,circle_color='r')
```

```
    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info( '*** Add links\n')
    net.addLink(ap1, ap2)
    net.addLink(ap2, ap3)
    net.addLink(ap3, ap1)
    net.addLink(sta1, ap1)
    net.addLink(sta2, ap2)
    net.addLink(sta3, ap3)
    net.addLink(sta4, ap1)
    net.addLink(sta5, ap2)
    net.addLink(sta6, ap3)

    net.plotGraph(max_x=2200, max_y=2200)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()



    info( '*** Starting switches/APs\n')
    net.get('ap1').start([c0])
    net.get('ap2').start([c0])
    net.get('ap3').start([c0])

    info( '*** Post configure nodes\n')

    CLI(net)
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

6. Handover.py

```
'Handover'
import sys
import os

from mininet.node import OVSSwitch,OVSKernelSwitch,RemoteController,OVSController
from mininet.log import setLogLevel, info
from mininet.topolib import TreeTopo
from mn_wifi.net import Mininet_wifi
from mn_wifi.node import Station, OVSKernelAP
from mn_wifi.cli import CLI
from mn_wifi.link import wmediumd
from mn_wifi.replaying import ReplayingMobility
from mn_wifi.wmediumdConnector import interference
from subprocess import call
from mininet.link import TCLink, Intf
from mn_wifi.replaying import ReplayingBandwidth
from mn_wifi.wmediumdConnector import interference
from random import randint, uniform,random

x=randint(1,10)
y=randint(1,10)
```

```
def topology(args):
    rangoAp=int (raw_input('Ingrese el rango de trabajo de los access point : '))
    rangoSta=int (raw_input('Ingrese el rango de trabajo de las estaciones : '))
    tiempo=int (raw_input('Ingrese el tiempo de espera antes del movimiento de las
estaciones : '))

    "Create a network."
    net = Mininet_wifi(topo=None,
                       link=wmediumd,
                       build=False,
                       wmediumd_mode=interference,
                       ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0 = net.addController(name='c0',
                           controller=RemoteController,
                           ip='127.0.0.1',
                           protocol='tcp',
                           port=6653)

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



    #ta1_args, sta2_args = dict(), dict()


    sta1 = net.addStation('sta1',ip='10.0.0.1', range=(rangoSta -
x) ,mac='00:00:00:00:00:01', speed=1)
    sta2 = net.addStation('sta2',ip='10.0.0.2',range=(rangoSta + x) ,mac='00:00:00:00:00:02',
speed=1)
    sta3 = net.addStation('sta3',ip='10.0.0.3', range=(rangoSta +
2*x) ,mac='00:00:00:00:00:03', speed=1,position='50,100,0')
    sta4 = net.addStation('sta4',ip='10.0.0.4',range=(rangoSta - y),mac='00:00:00:00:00:04',
speed=1)
    sta5 = net.addStation('sta5',ip='10.0.0.5', range=(rangoSta +
y) ,mac='00:00:00:00:00:05', speed=1)
    sta6 = net.addStation('sta6',ip='10.0.0.6',range=(rangoSta +
2*y) ,mac='00:00:00:00:00:06', speed=1,position='350,175,0')
    ap1 = net.addAccessPoint('ap1', cls=OVSKernelAP,ssid='ap1-ssid',range=rangoAp, mode='a',
channel='36', position='100,100,0',protocols='OpenFlow13')
    ap2 = net.addAccessPoint('ap2',cls=OVSKernelAP, ssid='ap2-
ssid',range=rangoAp ,mode='g' ,channel='6', position='100,200,0',protocols='OpenFlow13')
    ap3 = net.addAccessPoint('ap3',cls=OVSKernelAP, ssid='ap3-
ssid',range=rangoAp ,mode='g' ,channel='1', position='300,200,0',protocols='OpenFlow13')
```

```
    ap4 = net.addAccessPoint('ap4',cls=OVSKernelAP, ssid='ap4-
ssid',range=rangoAp ,mode='a' ,channel='44', position='300,100,0',protocols='OpenFlow13')
    ap5 = net.addAccessPoint('ap5',cls=OVSKernelAP, ssid='ap5-ssid',range=rangoAp -
15 ,mode='b' ,channel='4', position='200,150,0',protocols='OpenFlow13')
    #c1 = net.addController('c1')

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

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info("*** Creating links\n")
    net.addLink(ap1, ap2)
    net.addLink(ap3, ap4)
    net.addLink(ap3, ap2)
    net.addLink(ap1, ap4)
    net.addLink(ap5, ap1)
    net.addLink(ap5, ap2)
    net.addLink(ap5, ap3)
    net.addLink(ap5, ap4)
    net.addLink(ap1, sta3)


    net.addLink(ap4, sta4)
    net.addLink(ap3, sta6)



    ####################
    info("*** \n")
    info(ap1.params)
    info("*** \n")
    info(ap2.params)
    info("*** \n")
    info(ap3.params)
    info("*** \n")
    info(ap4.params)
    info("*** \n")
    info(ap5.params)
    info("*** \n")

   # tiempo=25

   #Metodos para escoger el AP a asociar con la estacion


   #Least Loaded First (llf)
   #Strongest Signal First (ssf)

    #info(ap1.wintfs[1].signal)

    net.startMobility(time=0, ac_method='ssf')
    net.mobility(sta1, 'start', time=tiempo, position='150,75,0')
    net.mobility(sta2, 'start', time=tiempo+5, position='150,200,0')
    net.mobility(sta4, 'start', time=tiempo-10, position='325,75,0')
    net.mobility(sta5, 'start', time=tiempo-3, position='300,50,0')
    net.mobility(sta1, 'stop', time=tiempo+10, position='50,225,0')
    net.mobility(sta2, 'stop', time=tiempo+17, position='125,75,0')
    net.mobility(sta4, 'stop', time=tiempo+12, position='150,200,0')
    net.mobility(sta5, 'stop', time=tiempo+19, position='250,200,0')
    net.stopMobility(time=tiempo +30)



    net.plotGraph(max_x=400, max_y=350)
```

```
    info("*** Starting network\n")
    net.build()
    c0.start()
    ap1.start([c0])
    ap2.start([c0])
    ap3.start([c0])
    ap4.start([c0])
    ap5.start([c0])

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

    info("*** Stopping network\n")
    net.stop()


if __name__ == '__main__':
    setLogLevel('info')
    topology(sys.argv)
```

7. Distancia.py

```
px import math

px a=float(sta5.position[0])
px b=float(sta5.position[1])
px c=float(ap4.position[0])
px d=float(ap4.position[1])
px e=float(ap3.position[0])
px f=float(ap3.position[1])


px distancia1=float(math.sqrt( (a-c)**2+(b-d)**2 ))
px distancia2=float(math.sqrt( (a-e)**2+(b-f)**2 ))

px print "La distancia al punto de acceso ap4 es %(distancia1)f" % {"distancia1": distancia1}
px print "La distancia al punto de acceso ap3 es %(distancia2)f" % {"distancia2": distancia2}

sta5 iw dev sta5-wlan0 link
```

8. Distancia2.py

```
px import math

px a=float(sta4.position[0])
px b=float(sta4.position[1])
px c=float(ap4.position[0])
px d=float(ap4.position[1])
px e=float(ap5.position[0])
px f=float(ap5.position[1])
px g=float(ap2.position[0])
px h=float(ap2.position[1])


px distancia1=float(math.sqrt( (a-c)**2+(b-d)**2 ))
px distancia2=float(math.sqrt( (a-e)**2+(b-f)**2 ))
px distancia3=float(math.sqrt( (a-g)**2+(b-h)**2 ))

px print "La distancia de sta4 a ap4 es %(distancia1)f" % {"distancia1": distancia1}
px print "La distancia de sta4 a ap5 es %(distancia2)f" % {"distancia2": distancia2}
px print "La distancia de sta4 a ap2 es %(distancia3)f" % {"distancia2": distancia3}

sta5 iw dev sta4-wlan0 link
```

9. Red_Ad-Hoc.py

```
# -*- coding: utf-8 -*-
"""
@author: nelsong
"""
import sys
import os

from mininet.log import setLogLevel, info
from mn_wifi.link import wmediumd, adhoc
from mininet.link import TCLink, Intf
from mn_wifi.cli import CLI
from mn_wifi.net import Mininet_wifi
from mn_wifi.wmediumdConnector import interference
def topology(args):

    "Create a network."
    net = Mininet_wifi(topo=None,
                       link=wmediumd,
                       build=False,
                       wmediumd_mode=interference,
                       ipBase='10.0.0.0/8')


    info("*** Creating nodes\n")
        #Configuracion estaciones moviles
    sta1 = net.addStation('sta1',ip='10.0.0.1', range=85 ,mac='00:00:00:00:00:01',speed=5)
    sta2 = net.addStation('sta2',ip='10.0.0.2',range=105 ,mac='00:00:00:00:00:02', position='150,250,0')
    sta3 = net.addStation('sta3',ip='10.0.0.3', range=85 ,mac='00:00:00:00:00:03',speed=5)
    sta4 = net.addStation('sta4',ip='10.0.0.4',range=105,mac='00:00:00:00:00:04', position='250,150,0')
    sta5 = net.addStation('sta5',ip='10.0.0.5', range=105 ,mac='00:00:00:00:00:05', position='250,250,0')
    sta6 = net.addStation('sta6',ip='10.0.0.6',range=105 ,mac='00:00:00:00:00:06',position='250,350,0')
    sta7 = net.addStation('sta7',ip='10.0.0.7',range=85 ,mac='00:00:00:00:00:07',speed=5)
    sta8 = net.addStation('sta8',ip='10.0.0.8', range=105 ,mac='00:00:00:00:00:11', position='350,250,0')
    sta9 = net.addStation('sta9',ip='10.0.0.9',range=85 ,mac='00:00:00:00:00:12',speed=5)

        #Configuracion Modelo de propagacion
    net.setPropagationModel(model="logDistance", exp=3.7)

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info("*** Creating links\n")
        #Configuracion enlaces Ad-Hoc
    net.addLink(sta1, cls=adhoc, intf='sta1-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta2, cls=adhoc, intf='sta2-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta3, cls=adhoc, intf='sta3-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta4, cls=adhoc, intf='sta4-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta5, cls=adhoc, intf='sta5-wlan0',
```

```
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta6, cls=adhoc, intf='sta6-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta7, cls=adhoc, intf='sta7-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta8, cls=adhoc, intf='sta8-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta9, cls=adhoc, intf='sta9-wlan0',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')

        #Configuraci n grafica posicion y rango
    net.plotGraph(min_x=0, max_x=500, min_y=0, max_y=500)
        #Configuracion modelo de movilidad
    net.setMobilityModel(time=10, model='RandomDirection', min_x=0, max_x=500, min_y=0, max_y=500,
                         min_v=0.5, max_v=2, seed=20)

    info("*** Starting network\n")
    net.build()
        #Configuracion grafica
    sta2.set_circle_color('r')
    sta4.set_circle_color('r')
    sta6.set_circle_color('r')
    sta8.set_circle_color('r')
    info("*** Running CLI\n")
    CLI(net)

    info("*** Stopping network\n")
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology(sys.argv)

# -*--FIN DE CODIGO---*-
```

10. RedMixta.py

```
# -*- coding: utf-8 -*-
"""
@author: nelsong
"""
import sys

from mininet.log import setLogLevel, info
from mn_wifi.cli import CLI
from mn_wifi.net import Mininet_wifi
from mn_wifi.link import adhoc, wmediumd
from mininet.node import OVSSwitch,OVSKernelSwitch,RemoteController,OVSController
from mn_wifi.wmediumdConnector import interference
from mn_wifi.node import Station, OVSKernelAP

def topology(args):

    "Create a network."
    net = Mininet_wifi(topo=None,
                       build=False,
                       link=wmediumd,
                       wmediumd_mode=interference,
                       ipBase='10.0.0.0/8',
                       ac_method='ssf')

    info( '*** Adding controller\n' )
            #Configuraci n controlador
    c0 = net.addController(name='c0',
                           controller=RemoteController,
                           ip='127.0.0.1',
                           protocol='tcp',
                           port=6653)

    info("*** Creating nodes\n")
            #configuraci n estaciones m viles
    sta1 = net.addStation('sta1',wlans=2,ip='10.0.0.1', range=50,mac='00:00:00:00:00:01', speed=4,position='330,310,0')
    sta2 = net.addStation('sta2',wlans=2,ip='10.0.0.2',range=50,mac='00:00:00:00:00:02', speed=5)
    sta3 = net.addStation('sta3',wlans=2,ip='10.0.0.3', range=50 ,mac='00:00:00:00:00:03', speed=4.5,position='300,280,0')
    sta4 = net.addStation('sta4',wlans=2,ip='10.0.0.4',range=50,mac='00:00:00:00:00:04', speed=3)
    sta5 = net.addStation('sta5',wlans=2,ip='10.0.0.5', range=50 ,mac='00:00:00:00:00:05', speed=3.5)
    sta6 = net.addStation('sta6',wlans=1,ip='10.0.0.6',range=50 ,mac='00:00:00:00:00:06', speed=4.5,position='280,310,0')
    sta7 = net.addStation('sta7',wlans=1,ip='10.0.0.7',range=50 ,mac='00:00:00:00:00:07', speed=5)

          #configuraci n puntos de acceso
    ap1 = net.addAccessPoint('ap1',cls=OVSKernelAP, ssid='ssid_1', mode='g', channel='5',
                             failMode="standalone",position='100,100,0',range=100,protocols='OpenFlow10')
    ap2 = net.addAccessPoint('ap2',cls=OVSKernelAP, ssid='ssid_2', mode='g', channel='5',
                             failMode="standalone",position='100,200,0',range=100,protocols='OpenFlow10')
    ap3 = net.addAccessPoint('ap3',cls=OVSKernelAP, ssid='ssid_3', mode='g', channel='5',
                             failMode="standalone",position='300,200,0',range=100,protocols='OpenFlow10')
    ap4 = net.addAccessPoint('ap4',cls=OVSKernelAP, ssid='ssid_4', mode='g', channel='5',
                             failMode="standalone",position='300,100,0',range=100,protocols='OpenFlow10')
```

```
    ap5 = net.addAccessPoint('ap5',cls=OVSKernelAP, ssid='ssid_5', mode='g', channel='5',
                             failMode="standalone",position='200,150,0',range=100,protocols='OpenFlow10')

        #Modelo de propagaci n
    net.setPropagationModel(model="logDistance", exp=3)

    info("*** Configuring wifi nodes\n")
    net.configureWifiNodes()

    info("*** Associating...\n")

        #Configuraci n enlaces Ad-Hoc
    net.addLink(sta1, cls=adhoc, intf='sta1-wlan1',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta2, cls=adhoc, intf='sta2-wlan1',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta3, cls=adhoc, intf='sta3-wlan1',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta4, cls=adhoc, intf='sta4-wlan1',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')
    net.addLink(sta5, cls=adhoc, intf='sta5-wlan1',
                ssid='adhocNet', mode='g', channel=5,
                ht_cap='HT40+', proto='olsr')

        #Configuraci n enlaces cableados
    net.addLink(ap1, ap2)
    net.addLink(ap3, ap4)
    net.addLink(ap3, ap2)
    net.addLink(ap1, ap4)
    net.addLink(ap5, ap1)
    net.addLink(ap5, ap2)
    net.addLink(ap5, ap3)
    net.addLink(ap5, ap4)
    net.addLink(ap1, sta6)
    net.addLink(ap3, sta7)

       #Configuraci n movilidad
    net.setMobilityModel(time=20, model='RandomDirection', min_x=50, max_x=350, min_y=50, max_y=250,
                         min_v=0.5, max_v=2, seed=10)
        #Gr fica nodos
    net.plotGraph(min_x=0, max_x=400, min_y=0, max_y=350)

       #Inicializaci n controlador y AP
    info("*** Starting network\n")
    net.build()
    c0.start()
    ap1.start([c0])
    ap2.start([c0])
    ap3.start([c0])
    ap4.start([c0])
    ap5.start([c0])

        #Asignaci n de direcciones IP
    info("*** Addressing...\n")
    sta1.setIP('10.0.0.1', intf="sta1-wlan1")
    sta2.setIP('10.0.0.2', intf="sta2-wlan1")
    sta3.setIP('10.0.0.3', intf="sta3-wlan1")
    sta4.setIP('10.0.0.4', intf="sta4-wlan1")
    sta5.setIP('10.0.0.5', intf="sta5-wlan1")

        #Configuraci n gr fica
    sta1.set_circle_color('r')
    sta2.set_circle_color('r')
    sta3.set_circle_color('r')
    sta4.set_circle_color('r')
    sta5.set_circle_color('r')
    sta6.set_circle_color('blue')
    sta7.set_circle_color('blue')
    ap1.set_circle_color('grey')
    ap2.set_circle_color('grey')
    ap3.set_circle_color('grey')
    ap4.set_circle_color('grey')
    ap5.set_circle_color('grey')
    info("*** Running CLI\n")
    CLI(net)

    info("*** Stopping network\n")
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology(sys.argv)

#--FIN DE CODIGO---------
```

Bibliografía

[1] V. Autores, "¿qué es sdn y como va a cambiar la visión de las redes? una introducción sencilla." urlhttps://www.error509.com/2019/02/que-es-sdn-y-como-va-a-cambiar-la-vision-de-las-redes-una-introduccion-sencilla/, 2018.

[2] L. CEVALLOS, "Implementación de redes definidas por software(sdn) sobre redes ieee 802.11 mediante mininet wifi." urlhttp://dspace.espoch.edu.ec/bitstream/123456789/9144/8/98T00197.pdf, 2018.

[3] R. Akeela and B. Dezfouli, "Software-defined radios: Architecture, state-of-the-art, and challenges," *Computer Communications*, vol. 128, pp. 106 – 125, 2018.

[4] V. Autores, "Centralizado, descentralizado y distribuido (p2p) ¿cuáles son las diferencias?." urlhttps://desarrolloactivo.com/articulos/centralized-decentralized-distributed-p2p/, 2018.

[5] A. Tar, "Bases de datos descentralizadas y distribuidas, explicadas." urlhttps://es.cointelegraph.com/explained/decentralized-and-distributed-databases-explained, 2017.

[6] V. Autores, "Red p2p." urlhttps://wikicharlie.cl/w/Red_P2P, 2019.

[7] A. Gatell, "Cómo funcina una red distribuida." url https://www.gatellasociados.com/como-funciona-una-red-distribuida/, 2018.

[8] V. Autores, "Redes." url https://www.mindomo.com/es/mindmap/redes-374eb0c892f446e69886dff3487953f5, 2019.

[9] TheLinuxFoundationProjects, "Descripción general de la plataforma." url https://www.opendaylight.org/what-we-do/odl-platform-overview, 2018.

[10] TheLinuxFoundationProjects, "Opendayligth beryllium." url https://www.opendaylight.org/what-we-do/current-release/beryllium, 2018.

[11] V. Autores, "¿qué es openflow?." urlhttps://searchdatacenter.techtarget.com/es/definicion/OpenFlow/, 2012.

[12] D. A. S. Carrera, "Redes definidas por software (sdn): Openflow." urlhttps://riunet.upv.es/bitstream/handle/10251/62801/SERRANO %20-%20Redes %20Definidas %20por %20Software %20(SDN): %20OpenFlow.pdf?sequence=3, 2015.

[13] Mininet.org, "Introducción a mininet." url https://github.com/mininet/mininet/wiki/Introduction-to-Mininetwhat, 2018.

[14] V. Autores, "Introduccion a mininet wi-fi." urlhttps://mininet-wifi.github.io/, 2018.

[15] H. Cho, C. Lai, T. K. Shih, and H. Chao, "Integration of sdr and sdn for 5g," *IEEE Access*, vol. 2, pp. 1196–1204, 2014.

[16] T. Feng, J. Bi, and K. Wang, "Allocation and scheduling of network resource for multiple control applications in sdn," *China Communications*, vol. 12, pp. 85–95, June 2015.

[17] T. Juhana and S. Girianto, "An sdr-based multistation fm broadcasting monitoring system," in *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pp. 1–4, Oct 2017.

[18] L. Ma, X. Wen, L. Wang, Z. Lu, and R. Knopp, "An sdn/nfv based framework for management and deployment of service based 5g core network," *China Communications*, vol. 15, pp. 86–98, Oct 2018.

[19] P. Shome, M. Yan, S. M. Najafabad, N. Mastronarde, and A. Sprintson, "Crossflow: A cross-layer architecture for sdr using sdn principles," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pp. 37–39, Nov 2015.

[20] S. Sun, M. Kadoch, L. Gong, and B. Rong, "Integrating network function virtualization with sdr and sdn for 4g/5g networks," *IEEE Network*, vol. 29, pp. 54–59, May 2015.

[21] N. World, "802.11: estándares de wi-fi y velocidades." urlhttps://www.networkworld.es/wifi/80211 -estandares-de-wifi-y-velocidades, 2018.

[22] PanamaHitek, "Handover en telecomunicaciones." urlhttp://panamahitek.com/handover-en-telecomunicaciones-parte/, 2014.

[23] CISCO, "El nuevo informe anual de internet de cisco pronostica que para 2023 más del 10 % de las conexiones móviles globales se ...." urlhttps://news-blogs.cisco.com/americas/es/2020/02/21/el-nuevo-informe-anual-de-internet-de-cisco-pronostica-que-para-2023-mas-del-10-de-las-conexiones-moviles-globales-seran-con-5g/, 2020.