ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROTOTIPO DE UNA SDN UTILIZANDO HERRAMIENTAS OPEN-SOURCE

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

JUAN FRANCISCO GUANO VISCARRA
juan.guano@epn.edu.ec

DIRECTOR: ING. CHRISTIAN JOSÉ TIPANTUÑA TENELEMA, MSc.
christian.tipantuna@epn.edu.ec

Quito, mayo 2017

DECLARACIÓN

Yo, Juan Francisco Guano Viscarra, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Juan Francisco Guano Viscarra

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Juan Francisco Guano Viscarra, bajo mi supervisión.

Ing. Christian Tipantuña, MSc.

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Agradezco a Dios, por brindarme la vida y la sabiduría para haber conseguido este logro que tanto he perseguido.

A mi madre, que tanto apoyo y amor incondicional me ha brindado a lo largo de mi vida.

A mi familia, que siempre creyó que iba alcanzar una meta más en mi vida.

A mi director de trabajo de titulación, MSc. Christian Tipantuña por su esfuerzo, paciencia, dedicación y guía a lo largo de la realización del presente trabajo.

A mis amigos, que de alguna u otra manera colaboraron y me brindaron su apoyo.

DEDICATORIA

A mi madre y familia, que siempre creyeron en mí.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

ÍNDICE DE COMANDOS

RESUMEN

En el presente trabajo se realiza la implementación de un prototipo de una SDN (*Software Defined Network*) utilizando una solución basada en hardware y software libre.

En el capítulo uno se describen los conceptos más importantes referentes a las SDN, su funcionamiento y las ventajas que tienen sobre las redes tradicionales. Además, se describen las características técnicas principales del protocolo OpenFlow, del controlador Ryu, del módulo Open vSwitch y de la computadora de placa reducida Raspberry Pi.

En el capítulo dos se presenta el diseño de la SDN a implementar. Se exponen los procesos de instalación y configuración de los sistemas operativos Raspbian y OpenWrt además del módulo Open vSwitch, todos estos elementos necesarios para el funcionamiento del prototipo. Se documenta el diseño de una aplicación básica de firewall utilizando el controlador Ryu y se desarrolla una interfaz gráfica para la configuración de parámetros en la aplicación. Se realiza la simulación del prototipo en el software Mininet con su correspondiente extensión Mininet-WiFi y posteriormente se realiza la implementación del prototipo físico.

En el capítulo tres se exponen las pruebas de conectividad y se realiza la verificación del intercambio de mensajes OpenFlow entre los dispositivos empleando el software Wireshark. Se presenta una comparación del perfomance (*throughput* y *jitter*) con los resultados obtenidos en la simulación, con el paquete Mininet-WiFi, y con el prototipo físico. Adicionalmente se verifica el funcionamiento de la aplicación de firewall implementada sobre la SDN. Al final del capítulo se presenta una tabla de costos referenciales de la implementación del prototipo.

En el capítulo cuatro se presentan las conclusiones y recomendaciones obtenidas a lo largo de la realización del prototipo.

PRESENTACIÓN

A nivel académico e investigativo, al estudiar las SDN, es recomendable realizar las distintas pruebas en prototipos para luego adaptar los cambios y configuraciones a sistemas de red que se encuentran en producción. La mayoría de prototipos, sin embargo, requieren de una alta inversión económica para su implementación; por ello en el presente trabajo de titulación se propone la implementación de un prototipo de bajo costo utilizando una solución basada en hardware y software libre. Adicionalmente, se pretende que el prototipo propuesto sea compatible con dispositivos inalámbricos (*access points*), es decir que brinde cierto grado de movilidad a las estaciones.

La presente implementación propone incentivar el estudio y experimentación con SDN y tanto la información recopilada como el prototipo obtenido pueden ser utilizados en prácticas de laboratorio a nivel de pregrado y posgrado, y en capacitación para empresas o instituciones que pretendan adoptar esta tecnología en sus redes de telecomunicaciones.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

En este capítulo se presenta una revisión de los fundamentos básicos de las Redes Definidas por Software (SDN del inglés *Software Defined Networks*), sus elementos y ventajas sobre las redes tradicionales. Además, se describen brevemente las características y campos del protocolo OpenFlow, las herramientas de hardware y software a ser utilizados en el prototipo. Adicionalmente se realiza una revisión de las características principales del software de simulación Mininet.

1.1 REDES DEFINIDAS POR SOFTWARE [1]

La introducción de nuevas tecnologías de red, ha generado consigo nuevas formas de administrar las redes debido a que cada vez estas incrementan su tamaño y complejidad, además de la demanda de mayor rapidez en los servicios e incremento en la cantidad de recursos, demandando así mayor agilidad y eficiencia en las redes, dentro de este contexto emerge el paradigma de las Redes Definidas por Software (SDN).

Las SDN tienen como principal objetivo el unificar la administración de la red y utilizando software, controlar la conectividad y el flujo de datos, posibilitando así la inclusión de reglas específicas para la manipulación de flujos de datos. En una SDN se plantea una arquitectura de red la cual separa el plano de datos del plano de control, este último plano tiene la característica de ser programable; esta propiedad posibilita realizar una abstracción de las características de los dispositivos de red mediante el uso de aplicaciones y servicios, permitiendo el manejo de la red como si se tratase de una entidad lógica o virtual.

1.1.1 ARQUITECTURA BÁSICA DE UNA SDN

La arquitectura básica de una SDN, la cual se puede apreciar en la Figura 1.1, se compone de tres capas: aplicación, control e infraestructura, con sus respectivas interfaces entre ellas. En la parte inferior se ubica el plano de datos, este contiene los dispositivos de red, los mismos que brindan sus características y funcionalidades al plano de control.

El plano de datos requiere abarcar las características de control y gestión necesarias para efectuar las acciones provenientes desde el controlador. En la parte del centro se encuentra el plano de control, compuesto principalmente por el controlador SDN.

El controlador realiza el control de bajo nivel sobre los dispositivos de la red e interpreta las órdenes del plano de aplicación, respondiendo a los requerimientos de las aplicaciones de acuerdo a los recursos de red que posee y determina las políticas convenientes para cumplir las demandas requeridas.

En la parte superior se ubica la capa aplicación, la misma que hace uso de las interfaces API[1], para enviar sus requerimientos de la red hacia el plano de control [1].

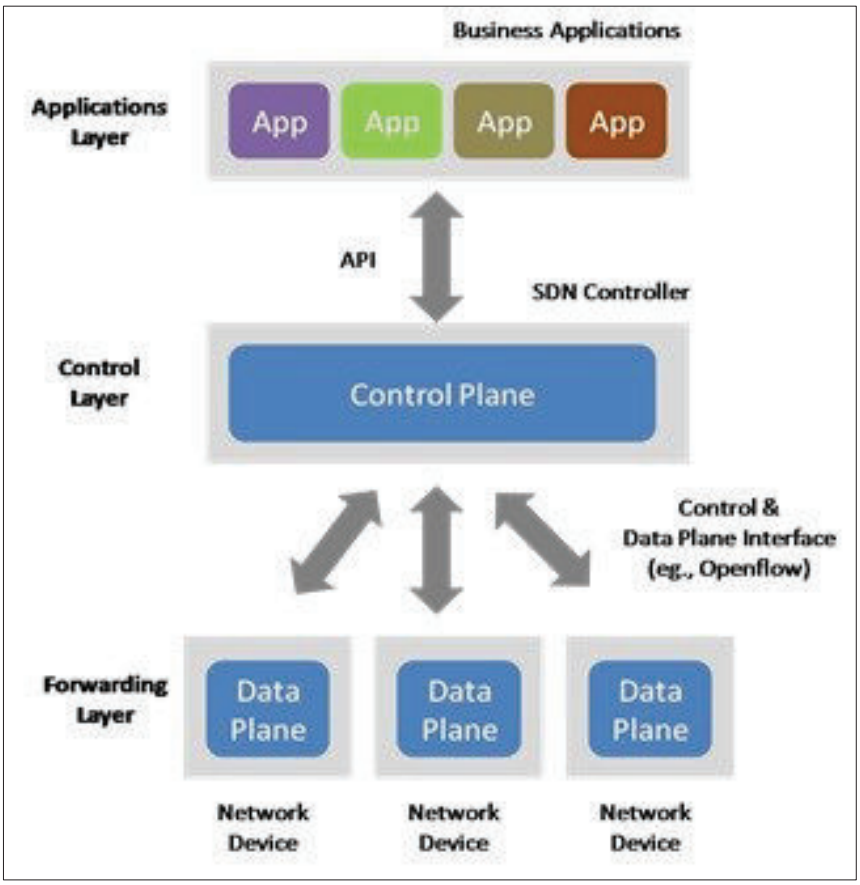

Figura 1.1. Arquitectura básica de una SDN [2]

A continuación, se describen los elementos básicos de una SDN:

1.1.1.1 Plano de Datos

El plano de datos o *forwarding* comprende los dispositivos de red y dependiendo de las reglas establecidas en las tablas de flujo[2] realiza el manejo de los paquetes entrantes y los procesa. La información es enviada hacia el controlador, para que

[1] API (*Aplication Programming Interface*): Formato de mensajes, el cual es utilizado, para la comunicación de una aplicación con el sistema operativo u otro programa de control.

[2] Tabla de flujo: Componente el cual contiene entradas con reglas para realizar el reenvió de paquetes en un *switch* para SDN.

efectúe las funciones de procesamiento o el análisis de datos, posteriormente la información ingresa y egresa del plano de datos a través de puertos sean estos físicos y/o virtuales. El plano de datos debe realizar el proceso de consulta al controlador para realizar el reenvió, no puede hacerlo por sí solo, sin embargo, en casos que se requiera un procesamiento más eficiente, como, por ejemplo, enfrentar alguna falla en la red[3], el controlador podrá almacenar ordenes preestablecidas para que el plano de datos los utilice y actué autónomamente.

1.1.1.2 Plano de Control

El plano de control ejerce la función principal de control en una SDN, está compuesto por el software controlador, el cual es el encargado de manejar los dispositivos de red y organizar el traspaso de información. El software controlador realiza el control del plano de datos y recepta de la capa aplicación los elementos para determinar el tipo de control que se va a ejecutar.

Dentro de la amplia variedad de controladores se encuentran, Ryu [3] , Floodlight [4], OpenDaylight [5], ONOS [6], entre otros. El controlador puede ser un solo componente lógico o un conjunto de varios controladores que componen una sola entidad. Un controlador puede contener varios módulos que se unen para efectuar funciones de red, como por ejemplo: determinación de la topología de red, recopilación de estadísticas de red, entre otros.

1.1.1.3 Plano de Aplicación

En este plano se localizan las aplicaciones, ya que dependiendo del caso pueden ser una o más, a dichas aplicaciones se les puede otorgar funciones que se requieran del controlador o un conjunto de ellos, a manera de reglas y políticas.

Las interfaces API intercambian mensajes entre el controlador y el plano de aplicación para consultar, informar el estado de la SDN y modificar el comportamiento de la red. Un ejemplo de estas acciones es la posibilidad de crear o modificar la conectividad dentro de la red o realizar el filtrado del tráfico de datos.

1.1.2 ESTÁNDARES Y NORMAS DE LAS SDN

A continuación, se revisará un estándar y una norma, los cuales actualmente son los más difundidos para las SDN. Cabe mencionar que el estándar surgio en conjunto con las SDN y la norma se desarrolló posteriormente.

---

[3] Falla en la red: Falla parcial o total de uno o varios componentes de una red ya sea por mal funcionamiento, falla natural o causada por la mano humana.

1.1.2.1 Open Networking Foundation [7]

La ONF[4] propuso en el año 2012 una normativa para las SDN, a continuación, se presenta una lista donde se explican los componentes que conforman la arquitectura de una SDN según la ONF. Estos componentes se pueden apreciar en la Figura 1.2.

- ***Drivers* y Agentes de interfaces:** Cada interfaz se implementa mediante un par driver-agente, el agente representa al "sur", inferior, de cara hacia la infraestructura y el *driver* representa el lado que mira hacia el "norte", arriba, de cara hacia la aplicación.
- **Interfaces de borde norte (NBI - *NorthBound Interfaces*):** Interfaces entre aplicaciones SDN y controladores, ofrecen vistas abstractas de la red y proporcionan el comportamiento de la red y de sus requerimientos. Actualmente no existe una API estándar, pero los más utilizados son REST y Java.
- **Aplicación SDN (SDN *Application*):** Programas que de forma directa, explicita y utilizando programación comunican los requerimientos y el comportamiento deseado de la red al controlador SDN a través de una NBI (*Northbound Interface*).

  Además, pueden poseer una visión abstracta de la red con el fin de tomar de decisiones internas. Una aplicación SDN consta de una aplicación lógica y uno o más *drivers* NBI.
- **Interfaz de control al plano de datos (CDPI - *Control to Data-Plane Interface*):** Interfaz definida entre un controlador y una ruta de datos que proporciona al menos lo siguiente: control pragmático de las operaciones de reenvío, capacidad de anuncio, informes estadísticos y notificaciones de eventos. La CDPI se implementa de manera abierta independientemente del fabricante o proveedor y a esta interfaz también se la conoce con el nombre de SBI (*SouthBound Interface*).
- **Ruta de Datos (*Datapath*):** dispositivo de red lógico que expone visibilidad y control sobre sus capacidades de reenvío y procesamiento de datos. La representación lógica puede abarcar todos o un subconjunto de los recursos

[4] ONF (*Open Networking Foundation*): Organización sin fines de lucro, creada para promover y estandarizar las SDN, el protocolo OpenFlow y tecnologías relacionadas.

físicos. Una ruta de datos está formada por un agente CDPI, un conjunto de uno más motores de reenvío de tráfico y de cero o más funciones de procesamiento de tráfico. Una o más rutas de datos pueden estar contenidas en un solo elemento de red, una combinación física integrada de los recursos de comunicaciones gestionada como una sola unidad. También pueden estar definidas a través de múltiples elementos de red físicos.

- **Controlador:** entidad lógica centralizada cuya función es la de traducir los requerimientos enviados por la capa aplicación SDN a través de las rutas de datos (*datapaths*) y de otorgar a las aplicaciones SDN una visión abstracta de la red (pudiendo incluir estadísticas y eventos). El controlador consta de uno o más agentes NBI, el centro de control lógico, y el driver CDPI.
- **Administración y Gestión:** El plano de administración o gestión efectúa tareas estáticas que se manejan mejor fuera de la aplicación, de los planos de control y de datos; por ejemplo, se puede tener la gestión de la relación comercial entre proveedor y cliente, la asignación de recursos a los clientes, la configuración del equipo físico, accesibilidad y coordinación de credenciales entre las entidades lógicas y físicas. Cada entidad comercial tiene su propia entidad de administración.

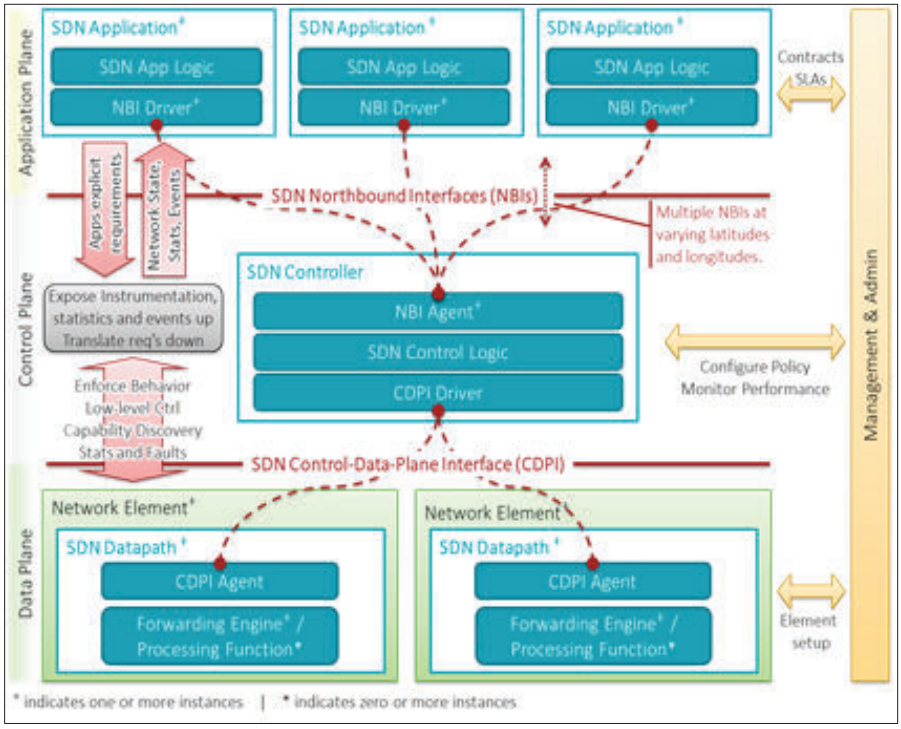

Figura 1.2. Arquitectura de una SDN según la ONF [7]

1.1.2.2 RFC 7426 – SDN: Layers and Architecture Terminology [8]

El RFC[5] 7426 fue lanzado en enero de 2015, por parte de la IETF[6], a partir de un análisis de los estándares y prácticas existentes sobre SDN. El RFC contiene principalmente lo siguiente:

- Describe la arquitectura de las SDN.
- Define capas, planos, interfaces y abstracciones.
- Enlista un gran número de documentos de referencia que se utilizaron de base para la consolidación del RFC.
- El principal objetivo de dicho documento, es el de definir la comunicación entre las entidades que componen las secciones "norte" (*northbound*) y "sur" (*southbound*) presentes en la arquitectura de una SDN.

A continuación, se presenta una lista donde se explican los componentes que conforman la arquitectura de una SDN según el RFC 7426. Estos componentes se pueden apreciar en la Figura 1.3.

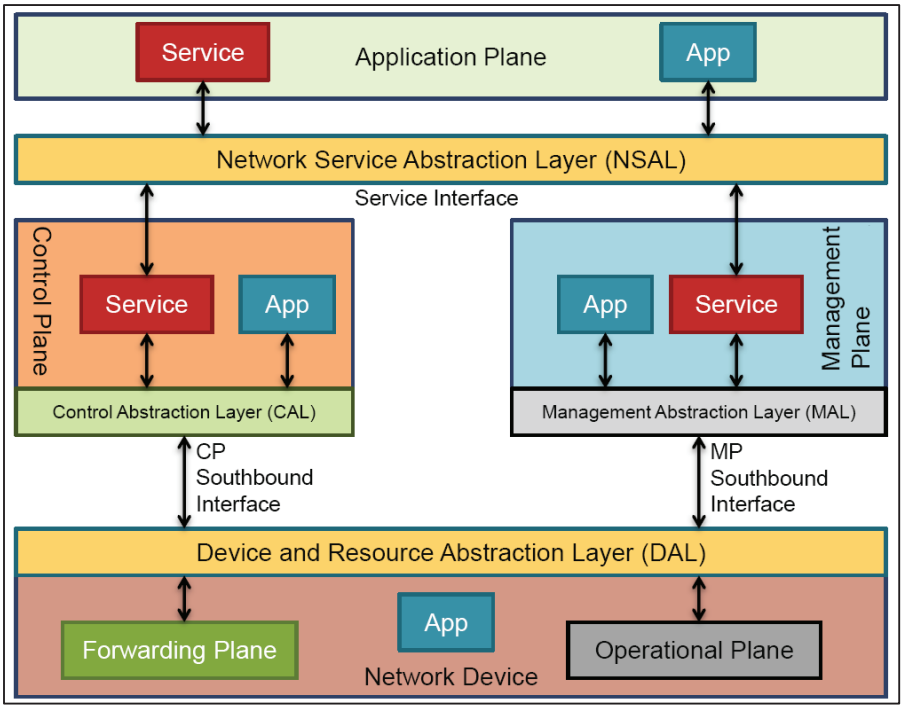

Figura 1.3. Arquitectura de una SDN según el RFC 7426 [9]

[5] RFC (*Request for Comments*): Serie de publicaciones que describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como protocolos, procedimientos, etc. y comentarios e ideas sobre estos.
[6] IETF (*Internet Engineering Task Force*): Organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad.

- **Dispositivo de red:** Dispositivo que realiza una o más operaciones de red relacionadas a la manipulación y reenvío de paquetes. En este modelo de referencia no se hace distinción del tipo de dispositivo de red, ya que puede ser físico o virtual.
- **Interfaz:** Punto de interacción entre dos entidades. En el caso de que dos entidades estén localizadas en diferentes ubicaciones la interfaz se implementa a través de un protocolo de red. Si dos entidades están colocadas en una misma ubicación, la interfaz puede ser implementada mediante un API, una comunicación entre procesos (IPC[7]), o un protocolo de red.
- **Aplicación (App):** En el contexto de las SDN es una pieza de software que utiliza servicios para realizar una determinada función. La operación de una aplicación puede ser parametrizada, como por ejemplo, pasar ciertos argumentos al momento de llamar a la aplicación. Una aplicación es una pieza de software del tipo *standalone*, sin embargo una aplicación no puede ofrecer interfaces a otras aplicaciones o servicios.
- **Servicio:** Pieza de software que realiza una o más funciones y provee una o más APIs a aplicaciones u otros servicios de las mismas capas o de diferentes capas, para ser utilizadas como funciones y retornar uno o más resultados. Los servicios pueden ser combinados con otros servicios para crear un nuevo servicio.
- **Plano de reenvío (FP - *Forwarding Plane*):** Colección de recursos de todos los dispositivos de red responsables del reenvío de tráfico.
- **Plano operacional (OP - *Operation Plane*):** Colección de recursos responsables del manejo de la administración de toda operación de los dispositivos de red individuales.
- **Plano de control (CP - *Control Plane*):** Colección de funciones con la responsabilidad de realizar el control de uno o más dispositivos de red, el plano de control indica a los dispositivos de red el cómo procesar y reenviar paquetes.

[7] IPC (*Inter-Process Communication*): Mecanismo que permite a los procesos comunicarse y sincronizarse entre sí, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la red subyacente a una entidad. Estos procesos se realizan en conjunto con todos las entidades que participan en la comunicación entre los planos de control y datos.

El plano de control interactúa en mayor parte con el plano de reenvío y en menor parte con el plano operacional.

Plano de administración (MP - *Management Plane*): Colección de funciones con la responsabilidad del monitoreo, configuración, y mantenimiento de uno o más dispositivos de red o partes de dispositivos de red. El plano de red se relaciona en mayor parte con el plano operacional y en menor parte con el plano de reenvío.

- **Plano de aplicación:** Colección de aplicaciones y servicios que programan el comportamiento de la red.
- **Capa de abstracción de dispositivo y recurso (DAL - *Device and resource Abstraction Layer*):** Capa que realiza una abstracción de los recursos de los dispositivos, basada en uno o más modelos. La capa provee un punto de referencia común para el plano de reenvío y el plano operacional del dispositivo.
- **Capa de abstracción de control (CAL - *Control Abstraction Layer*):** Capa que realiza una abstracción al plano de control. Provee acceso a la interfaz de plano de control de borde sur (CPSI[8]).
- **Capa de abstracción de administración (MAL - *Management Abstraction Layer*):** Capa que realiza una abstracción del plano de administración. Provee acceso a la interfaz de plano de administración de borde sur (MPSI[9]).
- **Capa de abstracción de servicios de red (NSAL - *Network Services Abstraction Layer*):** Capa que realiza abstracciones de servicios que pueden ser utilizadas por otras aplicaciones y servicios.

1.1.3 VENTAJAS DE UNA SDN SOBRE REDES TRADICIONALES [10]

En los últimos años son varias las tecnologías de red que han surgido, entre las que se pueden mencionar a: 5G[10], *Cloud Computing*[11], entre otras; las cuales demandan gran cantidad de recursos para manejar la información, dicho manejo principalmente se lo efectúa en los centros de datos y además representan nuevos

[8] CPSI (*Control-Plane Southbound Interface*): Interfaz localizada entre plano de control y la DAL.
[9] MPSI (*Management-Plane Southbound Interface*): Interfaz localizada entre plano de administración y la DAL.
[10] 5G: Quinta generación de tecnologías de telefonía móvil, se basa en comunicaciones de redes heterogéneas y un aumento en la movilidad de los usuarios.
[11] *Cloud Computing:* Tecnología de red que posibilita el almacenamiento de archivos e información en el Internet.

retos y necesidades en las redes de datos [10]. Por lo que la implementación de las SDN representa varias ventajas sobre las redes tradicionales, las cuales se presentan a continuación.

1.1.3.1 Automatización

La automatización permite a las redes actuar siguiendo los cambios de los servidores. También llamada "agilidad" porque representa la habilidad para inicializar y desactivar redes según se requiera. Esto ocurre rápidamente, eficientemente y con una intervención humana mínima.

1.1.3.2 Escalabilidad

Con la llegada de los centros de datos y ambientes en la nube, el número de estaciones de trabajo que se conectan a una red ha crecido exponencialmente. Las limitaciones del tamaño de las tablas de direcciones MAC y número de VLANs han traído consigo inconvenientes en la instalación y desarrollo de redes. El gran número de dispositivos físicos presentes en *data centers* también posee un problema de control de *broadcast*. El uso de túneles y redes virtuales pueden contener un número razonable de dispositivos en un dominio de *broadcast*.

1.1.3.3 Virtualización de red

La idea general de virtualización es la de crear una abstracción de alto nivel que se ejecute arriba de la entidad física que se esté utilizando. El crecimiento de la virtualización en servidores ha creado demanda para virtualización de red. Esto implica el tener una abstracción virtual de una red ejecutándose arriba de la red física. Al utilizar virtualización, el administrador de red debe ser capaz de manejar una red en cualquier parte y en cualquier momento, dependiendo o independientemente de la expansión o contracción de la red existente. En consecuencia, el nivel de virtualización de red requerida, para seguir el paso de la expansión e innovación en *data centers*, no es posible de ser alcanzado con las redes actuales.

1.1.3.4 Seguridad [11]

Una de las ventajas de las SDN que más atrae a los administradores de TI es la seguridad centralizada.

Con la gran cantidad de hosts enviando información a través de una red de datos, cada vez es más difícil aplicar *firewalls* y políticas de filtrado de contenido, y cuando

se agrega complejidades a una red de datos, como por ejemplo manejo de seguridad en dispositivos BYOD[12], el problema de seguridad se agrava.

El controlador SDN proporciona un punto central de control para distribuir información de seguridad y políticas de manera óptima en toda la empresa. Centralizar el control de seguridad en una entidad, como al utilizar un controlador SDN, tiene la desventaja de crear un punto central de ataque, pero las SDN se pueden utilizarse eficazmente para gestionar la seguridad en toda la empresa si se implementan de forma segura y correcta.

1.2 DISPOSITIVOS DE RED EN LAS SDN

En la presente sección se hará un repaso del funcionamiento de un *switch* tradicional y del funcionamiento del protocolo OpenFlow, el cual es uno de los más utilizados en dispositivos de red en las SDN.

1.2.1 FUNCIONAMIENTO Y ARQUITECTURA BÁSICA DE UN SWITCH TRADICIONAL

Una parte importante de la red son los *switches* a continuación se presenta de manera breve el funcionamiento y arquitectura básica de un *switch* tradicional [12]. En la Figura 1.4, se puede observar un *switch* tradicional no habilitado para SDN, que contiene los siguientes componentes:

- ***Transceivers* (TRX):** Son puertos transmitiendo o recibiendo señales de comunicación sobre medios de trasmisión, como: fibra óptica, cobre, radio frecuencia, entre otros.
- **Circuito integrado de aplicación específica (ASIC - *Application Specific Integrated Circuit*):** Utilizado para los paquetes de datos que ingresan o salen. Un ASIC esta optimizado para ejecutar un número limitado de tareas, se puede llegar a ejecutar tares a una alta velocidad de hasta 40Gbit/s por puerto.
- **Tablas de capa 2 y capa 3:** Bloques principales donde actúan los ASICs.

Para satisfacer las demandas de servidores virtualizados y redes de área local virtuales (VLANs), se han añadido nuevas características para una mejor protección de los hosts o de las máquinas virtuales: control de acceso, calidad de servicio (QoS), agrupación de puertos, entre otras.

---

[12] BYOD (*Bring your own device*): Política empresarial que consiste en que los empleados lleven sus propios dispositivos a su lugar de trabajo para tener acceso a recursos de red de la empresa.

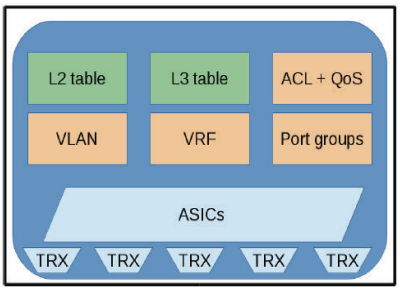

Figura 1.4. Esquema básico de un *switch* tradicional [12]

En la Figura 1.5, se puede apreciar la combinación de los componentes de un *switch* tradicional con los planos de datos, control y administración. Los componentes se combinan de la siguiente manera:

- Los *transceivers* (TRX) y los ASICs forman el plano de control.
- Un CPU de propósito general contiene tanto el plano de control como el plano de administración.
- El plan de control maneja funciones de enrutamiento y también es responsable de calcular las reglas de reenvío.
- El plano de administración es utilizado para configurar y modificar los parámetros de red del *switch*.

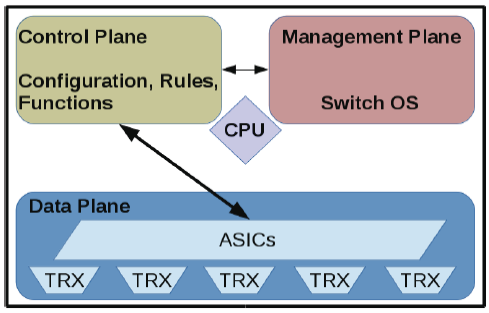

Figura 1.5. Planos de datos, control y administración en un *switch* [12]

En la Figura 1.6, se puede observar el proceso que realiza un nuevo paquete cuando llega al *switch*, los pasos que sigue son los siguientes:

1. Un nuevo paquete llega a un puerto receptor y es almacenado en un buffer.
2. Cuando no existe ninguna regla de emparejamiento para el paquete en el plano de datos, el plano de control decide qué hacer con el paquete.
3. El plano de control recibe el paquete desde el plano de datos y ejecuta funciones de enrutamiento.
4. Posteriormente, el plano de control almacena en las tablas de reenvío, la acción calculada a ejecutar como por ejemplo, él envió del paquete hacia

algún puerto de salida. Las tablas de reenvío son almacenadas en una memoria de contenido direccionable, para un emparejamiento rápido y eficiente.

5. En este punto, el plano de datos puede aplicar las reglas almacenadas en la memoria ternaria de contenido direccionable (TCAM).
6. El plano de datos reenvía el paquete al puerto de salida.
7. El puerto de salida transmite el paquete sobre el medio.

La petición hacia el plano de control toma tiempo, ya que la acción necesita ser calculada y el CPU del plano de control se ejecuta a una baja velocidad.

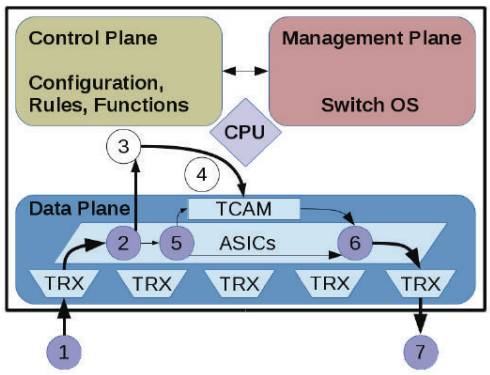

Figura 1.6. Pasos que sigue un paquete al llegar al *switch* [12]

En la Figura 1.7 se puede apreciar el proceso que realiza un paquete que tiene la misma fuente y destino, para el reenvío se utiliza reglas ya existentes, siempre y cuando no hayan expirado, los pasos que sigue el paquete son los siguientes:

1. El paquete llega.
2. Este es emparejado con las reglas almacenadas en la TCAM.
3. Luego el paquete es reenviado a un puerto de salida.
4. El puerto de salida transmite el paquete sobre el medio.

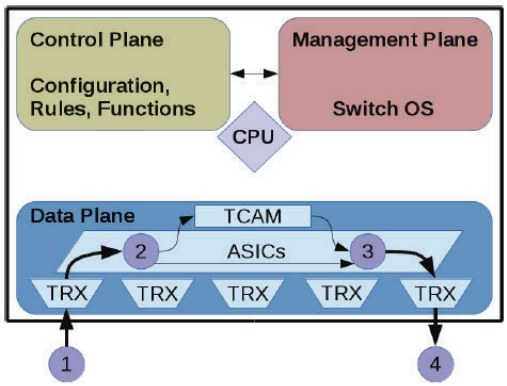

Figura 1.7. Pasos de un paquete con una regla de reenvío conocida [12]

En la Figura 1.8, se puede observar un esquema básico de un *switch* SDN junto a su controlador, donde a diferencia de un *switch* tradicional, ver Figura 1.8, existe una separación entre el *switch* y el controlador, las principales características de un *switch* SDN son:

- El *switch* posee una capa de abstracción mínima, con pocas funciones de control y administración, las cuales se necesitan para la operación de arranque en frío y configuración del *switch*.
- Las decisiones se toman en el plano de control, el cual ahora forma parte del controlador, el cual ofrece un plano de administración central.

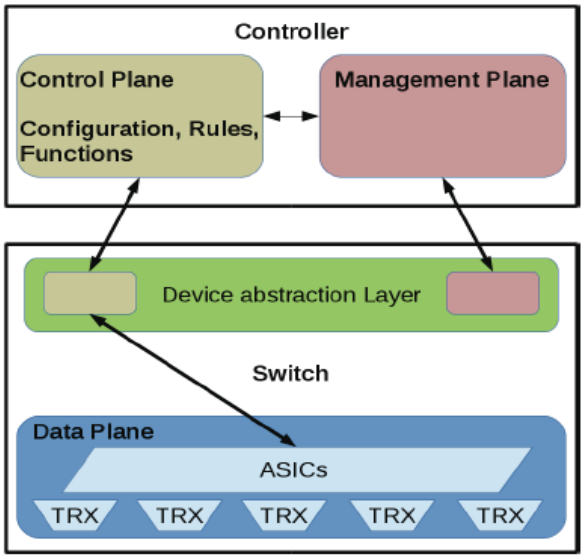

Figura 1.8. Esquema básico de un *switch* SDN y su controlador [12]

1.2.2 OPENFLOW [13]

OpenFlow surgió en el año 2008, a partir de ideas de investigadores de la Universidad de Stanford, los cuales buscaban nuevas formas de experimentar en las redes tradicionales actuales. Este protocolo propuso el concepto de un software de control que esté separado de la red, estandarizando la forma de comunicación entre el controlador con los dispositivos de red, permitiendo así programar tablas de flujo por las aplicaciones de software y especificando como ejercer el control de la red desde el controlador.

1.2.2.1 Dispositivos OpenFlow [13]

Un dispositivo de interconectividad, como por ejemplo *switch*, *access point* (AP), entre otros, para que sea considerado compatible con el protocolo OpenFlow, debe contener los siguientes componentes:

1.2.2.1.1 Tabla de flujos

La tabla de flujos es la parte esencial de los dispositivos, y en ella se localizan un conjunto de entradas las cuales brindan instrucciones de reenvío las cuales indican las acciones que se realizan con determinado flujo de tráfico. Entiéndase por flujo al conjunto de paquetes que tengan las mismas características, como, por ejemplo, compartir la dirección IP de origen e IP destino. Las entradas de flujo han ido incrementando los campos que lo componen, conforme han sido lanzadas sus versiones. Las entradas de flujo pueden estar compuestas por los campos dados a continuación:

- **Campos de correspondencia:** Especifican un cierto flujo a través del establecimiento de un conjunto de campos que se comparan con los paquetes recibidos en los dispositivos de interconectividad.
- **Prioridad:** Se establece un orden de preferencia en las entradas de una tabla de flujos.
- **Contadores:** Indican estadísticas sobre el flujo correspondiente a una entrada dada, como por ejemplo la cantidad de paquetes o *bytes* que se identifiquen.
- **Instrucciones:** Muestran una acción o las acciones que se ejecutarán con un flujo dado.
- **Temporizadores:** Estos elementos permiten la remoción de una entrada dada, de manera automática después de haber transcurrido un cierto lapso de tiempo.
- ***Cookie*:** Cada controlador añade cierta información llamada *cookie*, la cual funciona para identificar cada entrada de flujo, esta información se irá acumulando conforme se actualicen las tablas de flujos.

1.2.2.1.2 Protocolo OpenFlow

Este protocolo propone una forma abierta y estándar en la comunicación entre el controlador y los dispositivos de interconectividad, brindándole a este último la posibilidad de realizar las funciones: inserción, remoción, modificación y búsqueda en las entradas de la tabla de flujos a través del canal seguro. En la Figura 1.9 se puede apreciar un dispositivo OpenFlow, así como los componentes que lo conforman.

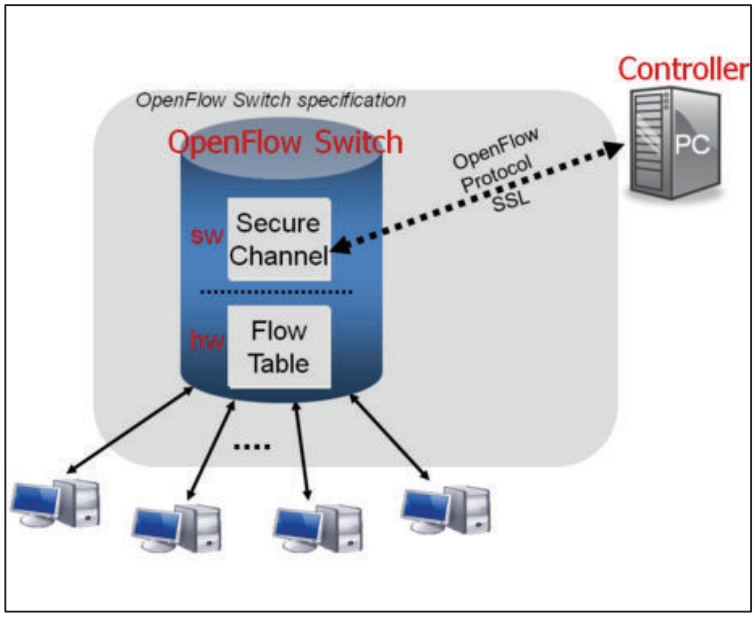

Figura 1.9. Dispositivo de red OpenFlow [13]

1.2.2.1.3 Canal seguro

Realiza la función de conectar el dispositivo con el controlador, permitiendo la trasferencia de comandos y paquetes entre ambos entes esto se lo realiza mediante el protocolo OpenFlow. Los protocolos para trasportar la información son TCP o SSL[13] en el caso de conexiones en ambientes inseguros.

1.2.2.2 Versiones del protocolo OpenFlow

La primera implementación de referencia del protocolo fue lanzada en noviembre de 2007, con la versión experimental 0.1.0, a partir de la colaboración de estudiantes e investigadores universitarios principalmente de las universidades de Stanford y Berkeley [14]. Posteriormente se continuo con las versiones experimentales: 0.2.0, 0.8.0, 0.81, 0.8.2, 0.8.9, lanzadas en 2008 y 0.9 en el 2009. La versión 1.0.0 [15], siendo la primera versión en ser ampliamente adoptada, fue lanzada en diciembre de 2009, esta versión contiene los siguientes campos de correspondencia (*match fields*), tal como se muestran en la Figura 1.10:

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
32 bits	48 bits	48 bits	16 bits	13 bits	3 bits	32 bits	32 bits	8 bits	6 bits	16 bits	16 bits

Figura 1.10. Campos de correspondencia del protocolo OpenFlow v. 1.0.0 [15]

[13] SSL (*Secure Socket Layer*): Protocolo para el establecimiento de una conexión segura mediante un canal cifrado entre dos extremos de una red de datos.

En la Tabla 1.1, se puede encontrar un resumen de los campos de correspondencia más relevantes que se han añadido en las versiones posteriores a la versión 1.0.0.

Tabla 1.1. Resumen de campos de correspondencia OpenFlow [16]

Versión del protocolo OpenFlow	Campos de Correspondencia
v 1.1.0	Campo metadata, campos para el protocolo SCTP[14], campos para el protocolo ICMP[15], campo de etiqueta MPLS[16], campo para clase de tráfico MPLS EXP[17], entre otros.
v 1.2.0	Campos de correspondencia extendidos de OpenFlow, campos para el protocolo IPv6, campos para el protocolo ICMPv6, entre otros.
v 1.3.0	Campos para extensión de cabecera IPv6, entre otros.
v 1.4.0	No posee nuevos campos que de su versión predecesora.
v 1.5.0	Campos para TCP *flags*, campo de puerto de egreso por conjunto de acciones, entre otros.

En el momento que un paquete ingresa a un dispositivo de interconectividad, este sigue el siguiente proceso:

- Los campos de correspondencia de los paquetes se extraen y se usan para encontrar coincidencias.
- Se procede a efectuar la comparación de las reglas previamente establecidas para cada entrada en la tabla de flujos OpenFlow, teniendo en cuenta el orden descendente de prioridad. Un paquete, puede coincidir con una entrada particular de la tabla de flujos, con uno o varios de sus campos.
- En caso de que exista coincidencia, se realizarán las acciones sobre el paquete establecidas en esa entrada. En el caso que no se halle

[14] SCTP (*Stream Transmission Control Protocol*): Protocolo de capa transporte orientado a las conexiones, similar a TCP, pero proporciona una transferencia de datos orientada a mensajes, similar a UDP.
[15] ICMP (*Internet Control Message Protocol*): Protocolos de capa de red, su tarea es manejar la función de control del funcionamiento correcto de la red, es utilizado para enviar todo tipo de mensajes de bajo nivel, que conciernen las irregularidades detectadas durante las conexiones de red.
[16] MPLS (*Multiprotocol Label* Switch*ing*): Estándar IP de conmutación de paquetes.
[17] MPLS EXP: Campo específico de MPLS para brindar calidad de servicio.

coincidencia, los primeros 200 *bytes* del paquete se enviarán al controlador para que estos sean procesados.

A partir de la versión 1.1.0 del protocolo OpenFlow [17] se introdujo el concepto de tabla de grupo. La tabla de grupo está diseñada para ejecutar operaciones comunes entre varios flujos, brindando la posibilidad de agrupar acciones que correspondan a un conjunto de flujos.

Los campos de correspondencia contienen nuevos campos adicionales, entre los que se tienen:

- **Metadatos**: Usado para transferir información entre las tablas cuando un paquete los atraviesa.
- **MPLS:** Etiqueta utilizada para brindar soporte al protocolo MPLS.
- **MPLS EXP:** Etiqueta para clase de tráfico, utilizada para brindar soporte al protocolo MPLS.

En el momento que un paquete ingresa al dispositivo, este se envía y se busca en la primera tabla. Si se encuentra una coincidencia, el paquete es procesado y si coincide con otra tabla, se envía hacia ella. El proceso termina una vez no existan más coincidencias con más tablas. Adicionalmente, en la versión 1.1.0 se gestionan instrucciones en lugar de acciones. Anteriormente una acción se asociaba a cada entrada de la tabla de flujos, el paquete se reenviaba o descartaba, o se procesaba normalmente como si se tratase de un dispositivo de interconectividad normal.

En la versión 1.1.0 las instrucciones son más complejas, porque pueden modificar el paquete, actualizar acciones agrupadas o actualizar el campo de metadatos.

La versión 1.2.0 [18] añadió el soporte a direcciones IPv6, además de la posibilidad de que los dispositivos de interconectividad puedan conectarse a múltiples controladores, los cuales también pueden comunicarse también entre ellos.

La versión 1.3.0 [19] añade la característica de controlar la cantidad de paquetes mediante medidores de flujo. También se incluyen conexiones auxiliares entre el controlador y los dispositivos de red. Además de la incorporación de *cookies*.

La versión 1.4.0 [20] incluye una nueva función para monitorización de flujos.

Finalmente la versión 1.5.0 [21] posee una nueva característica llamada tabla de egreso, la cual consiste en el permitir la búsqueda de coincidencias de paquetes basada en el puerto de salida, característica que no se implementa en versiones anteriores.

1.3 TECNOLOGÍAS Y COMPONENTES EMPLEADAS EN EL PROTOTIPO

El presente prototipo para su desarrollo propone la utilización de tecnologías y componentes de hardware y software del tipo *open-source*. A continuación, se presenta las tecnologías empleadas en el prototipo.

1.3.1 COMPONENTES DE HARDWARE

A continuación, en la presente sección se presentan las herramientas de hardware utilizadas en el prototipo.

1.3.1.1 Raspberry Pi [22]

Una de las herramientas de hardware libre ampliamente utilizadas en los últimos años, es la plataforma Raspberry Pi, la cual es un computador de placa reducida que se compone de puertos de entrada y salida, que permiten la implementación de proyectos en diversos campos de la ingeniería a un bajo costo [22]. En la Tabla 1.2 se presenta las principales especificaciones de los distintos modelos de la plataforma.

Tabla 1.2. Especificaciones técnicas de Raspberry Pi por modelo [23]

Modelo	Raspberry Pi Modelo B	Raspberry Pi Modelo B+	Raspberry Pi 2 Modelo B	Raspberry Pi 3 Modelo B
System On a Chip (SoC)	BCM2835	BCM2835	BCM2836	BCM2837
CPU	ARM11 @ 700MHz	ARM11 @ 700MHz	Quad Cortex A7 @ 900MHz	Quad Cortex A53 @ 1,2GHz
GPU	250 MHz VideoCore IV	250 MHz VideoCore IV	250 MHz VideoCore IV	400 MHz VideoCore IV
RAM	512MB	512MB	1GB	1GB
Puertos USB	2	4	4	4
Almacenamiento	SD	microSD	microSD	microSD
GPIO[18]	26	40	40	40
Ethernet	10/100Mbps	10/100Mbps	10/100Mbps	10/100Mbps
Conectividad Inalámbrica	No incluida	No incluida	No incluida	Wi-Fi y Bluetooth 4.0

[18] GPIO (*General Purpose Input/Output*,): Pin de entrada o salida, presentes en chips, que se puede programar por el usuario en tiempo de ejecución.

La gran aceptación de la plataforma, tanto por investigadores y estudiantes de ingeniería, se debe a que la plataforma está compuesta por puertos de entrada y salida, como, por ejemplo: puerto HDMI, puertos USB, puerto Jack 35mm de audio, bus serial de I/O y puertos GPIO. A continuación, en la Figura 1.11 se presenta la estructura física del Raspberry Pi 3 Modelo B, con sus respectivos componentes. Debido a que la placa que soporta varias distribuciones basadas en LINUX, es posible utilizar una interfaz gráfica y diseñar aplicaciones a partir de una gran cantidad de lenguajes de programación.

La distribución libre más utilizada para esta plataforma es Raspbian, la cual es una versión ligera de Debian, optimizada para arquitecturas ARM[19] de 32 bits.

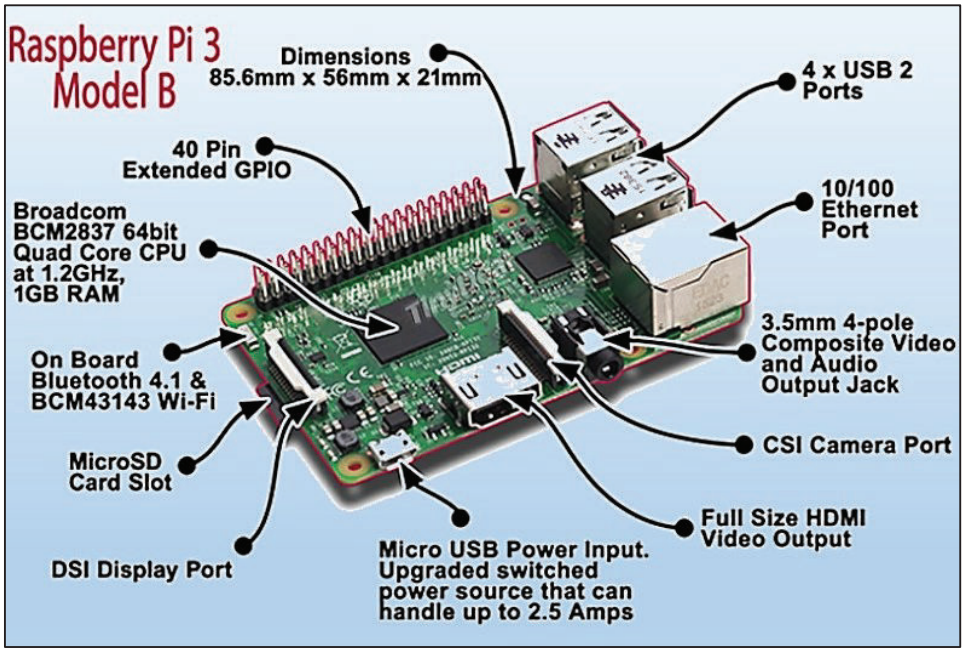

Figura 1.11. Arquitectura física de la placa Raspberry Pi 3 Modelo B [24]

1.3.2 COMPONENTES DE SOFTWARE

A continuación, en la presente sección se presentan las herramientas de software utilizadas en el prototipo.

1.3.2.1 Open vSwitch [25]

Basado en software, Open vSwitch (*Open Virtual Switch*) es un *switch* virtual que se encuentra en el plano de datos, es de código abierto y está bajo la licencia de Apache 2.0[20]. Open vSwitch habilita la automatización de las redes mediante la

[19] Arquitectura ARM: Procesadores de arquitectura de 32 y 64 bits, los cuales poseen bajo consumo de energía, son ampliamente utilizados en dispositivos móviles y portátiles, ya que no requiere refrigeración adicional.
[20] Apache 2.0: Licencia de software libre permisiva creada por la *Apache Software Foundation*.

extensión de su programabilidad, soportando la administración de interfaces y protocolos. Adicionalmente, puede ser instalado en diferentes sistemas operativos y plataformas de hardware.

La virtualización en conjunto con Open vSwitch es utilizada para el desarrollo de topologías virtuales, debido a la posibilidad de ejecutar múltiples conmutadores en un mismo *host*.

1.3.2.1.1 Componentes de Open vSwitch [12]

De manera breve se puede decir que Open vSwitch se constituye de múltiples componentes, en la Figura 1.12 se puede apreciar sus principales componentes, los cuales se describen a continuación:

- **`openvswitch.ko`:** Es el módulo de kernel de LINUX que permite la conmutación basada en flujos, mediante un `data path`[21] rápido localizado en el *kernelspace*[22].
- **`ovs-vswitchd`:** Un proceso en el *userspace*[23], el cual implementa la lógica del *switch* y realiza la abstracción del plano de control.
- **`ovsdb-server`:** Un servidor de base de datos ligero, el cual contiene la configuración del sistema.

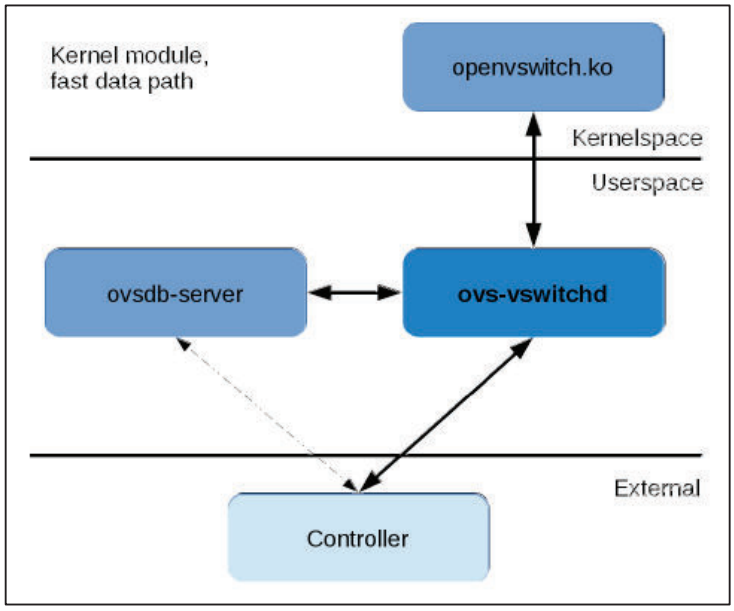

Figura 1.12. Componentes de Open vSwitch [12]

[21] El termino `data path` es utilizado para referirse al reenvió de paquetes en un dispositivo, como por ejemplo, un *switch*.
[22] *Kernelspace*: Proceso que puede ocupar los recursos de memoria del kernel del sistema operativo.
[23] *Userspace*: Proceso que ejecutan las aplicaciones de usuario utilizando sus propios recursos de memoria.

Un controlador externo puede comunicarse con `ovs-vswitchd` a través del protocolo OpenFlow, y también puede comunicarse directamente la base de datos `ovsdb-server`.

1.3.2.1.2 Funcionamiento básico de Open vSwitch [26]

En Open vSwitch, existen dos componentes principales para el reenvío de paquetes. El primero y más extenso, es `ovs-vswitchd`. El otro componente, es el módulo *`data path`* en kernel, este usualmente está programado especialmente para alto rendimiento del sistema operativo anfitrión.

En la Figura 1.13, se representa como los dos principales componentes trabajan juntos para el reenvío de paquetes. Primero el módulo *`data path`* en kernel recibe los paquetes, desde una NIC[24] física o una NIC virtual de una máquina virtual y se pueden producir dos casos:

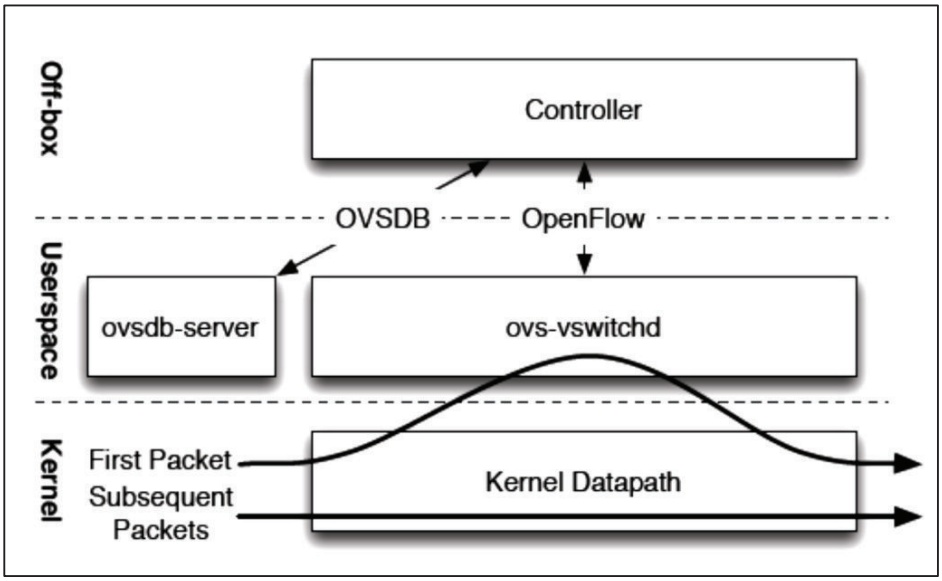

Figura 1.13. Funcionamiento básico de Open vSwitch [26]

- En el primer caso, `ovs-vswitchd` indica cómo manejar los paquetes al módulo *`data path`*, en este caso el módulo *`data path`* sigue las instrucciones, llamadas acciones, dadas por el `ovs-vswitchd`, el cual establece una lista de puertos físicos o túneles en las cuales trasmitir el paquete. Estas acciones también pueden especificar modificaciones o eliminación del paquete.

[24] NIC (*Network Interface Card):* Componente de hardware presente en un computador, el cual permite al computador conectarse a una red de datos, una NIC puede ser cableada o inalámbrica según al tipo de medio que utilice la red a la que se conecte.

- En el segundo caso, el *data path* no ha recibido instrucciones para manejar el paquete, en este caso el paquete es entregado al ovs-vswitchd. En el *userspace*, el ovs-vswitchd determina como el paquete debe ser manejado, luego el paquete es enviado de nuevo hacia el *data path* con el manejo deseado.

Usualmente ovs-vswitchd también indica al *data path* que almacene las acciones para casos en que se requiera manejar futuros paquetes de manera similar.

Open vSwitch, ha ido evolucionado considerablemente a través del tiempo en cuanto a cache de flujos[25]. La primera versión del *data path* empezó con un cache de micro flujos, el mismo que básicamente consistía en el cacheo por decisiones de reenvió en conexión de transporte. En últimas versiones, el *data path* tiene dos capas de almacenamiento: un cache de micro flujos y una capa secundaria llamado cache de mega flujos, la cual realiza un cacheo de decisiones de reenvío por acumulación de tráfico aparte de realizarlo en conexiones individuales.

Open vSwitch es usualmente utilizado como un *switch* SDN, y la principal manera de realizar un control sobre el reenvío de paquetes, es utilizando el protocolo OpenFlow. En Open vSwitch, el ovs-vswitchd recibe las tablas de flujo OpenFlow desde el controlador SDN, busca coincidencias entre cualquier paquete recibido desde el módulo *data path* y las tablas OpenFlow, reúne las acciones a aplicar y finalmente cachea el resultado del *data path* en el kernel. Esto permite que el módulo *data path* simplifique el proceso de reenvió de un paquete en un *switch* OpenFlow.

Desde el punto de vista del controlador, la separación entre componentes de usuario y kernel es una implementación invisible, pues cada paquete llega a una serie de tablas de flujo OpenFlow y el *switch* busca el flujo de más alta prioridad el cual tenga condiciones (coincidencias con los campos de correspondencia) por ser cumplidas para el paquete, luego el controlador ejecuta las acciones establecidas sobre OpenFlow.

---

[25] Cache de flujos: Acumulación o almacenamiento de flujos que se utiliza para optimizar y reducir el tiempo del proceso de reenvió de paquetes en *switches* OpenFlow. Este proceso automatiza varias tareas del proceso de reenvió de paquetes.

1.3.2.1.3 Aplicaciones de control de Open vSwitch [12]

Open vSwitch ofrece varias aplicaciones de control, las cuales brindan la posibilidad de configurar, administrar y monitorear el funcionamiento de un *switch* OpenFlow creado por el paquete de software Open vSwitch, en la Figura 1.14 se puede apreciar sus principales aplicaciones de control entre las cuales se tienen:

- **`ovs-dpctl`:** Una utilidad para el control y la conexión del *`data path`* (módulo de kernel `openvswitch.ko`)
- **`ovs-vsctl`:** Una utilidad principal utilizada para configurar el *switch* virtual `ovs-vswitchd`.
- **`ovs-ofctl`:** Una utilidad la cual puede ser utilizada para conectarse con el `ovs-vswitchd` a través de OpenFlow.
- **`ovsdb-tool`:** Una utilidad empleada para administrar la instancia de la base de datos `ovsdb-server`.
- **`ovs-appctl`:** Una utilidad para administrar los procesos demonio de Open-vSwitch.

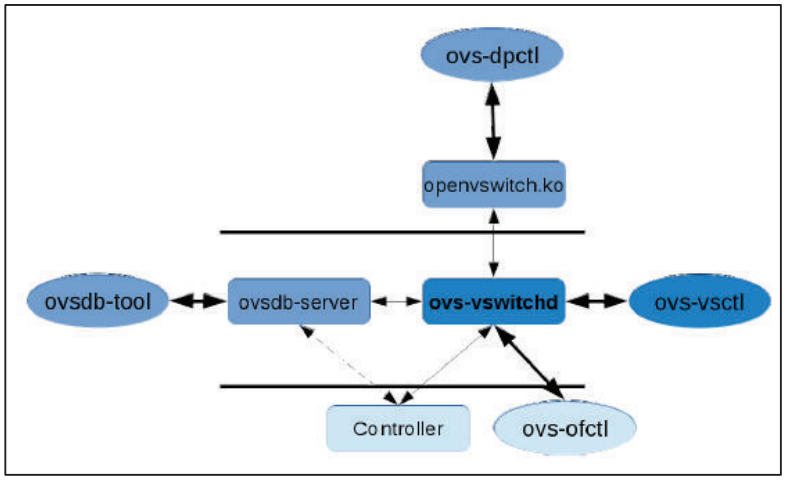

Figura 1.14. Aplicaciones de control de Open vSwitch [12]

1.3.2.2 Controlador basado en software Ryu [27]

Como se mencionó en la sección 1.1.1.2, existe una gran variedad de controladores SDN, en la presente sección se hablará sobre el controlador Ryu [27].

El controlador basado en software Ryu, se ejecuta sobre distribuciones LINUX y posee varios componentes de desarrollo para SDN. Ryu ofrece dichos componentes con una interfaz API bien definida que es muy útil para el desarrollo de nuevas aplicaciones de control sobre una SDN. Adicionalmente el controlador

permite la integración con otros componentes, como por ejemplo OpenStack[26], la cual es una plataforma de software para la administración de infraestructura SDN en la nube.

A continuación, se mencionan algunas de las características del controlador Ryu:

- Software de código abierto, disponible bajo la licencia Apache 2.0.
- Basado en el lenguaje de programación Python[27], conocido por su amplio soporte y fácil utilización.
- Contiene componentes que pueden ser reusados y ofrece la posibilidad de implementar nuevos componentes.
- Posee compatibilidad para ejecutarse sobre varios conmutadores OpenFlow, entre ellos Open vSwitch.

A continuación, en la Figura 1.15 se presentan de manera resumida los componentes del controlador Ryu [28].

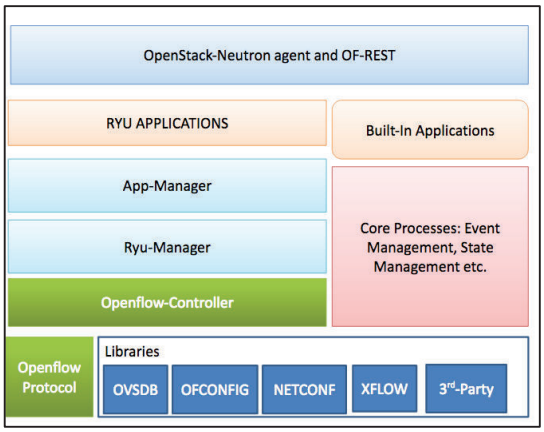

Figura 1.15. Resumen de componentes de Ryu [29]

1.3.2.2.1 Ejecutables

El componente bin/ryu-manager es el principal archivo ejecutable de Ryu y es necesario para ejecutar las aplicaciones de Ryu.

1.3.2.2.2 Componentes Base

ryu.base.app_manager es el componente que realiza el manejo principal de las aplicaciones de Ryu, encargándose de las siguientes funciones:

- Carga las aplicaciones de Ryu.
- Provee de contextos[28] a las aplicaciones de Ryu.

[26] OpenStack: Proyecto de computación en la nube para proporcionar una infraestructura como servicio (IaaS).
[27] Python: Lenguaje de programación de alto nivel, interpretado y multipropósito.
[28] Contextos: Los contextos son objetos Python que se pueden utilizar de forma compartida entre aplicaciones.

- Envía mensajes hacia las aplicaciones de Ryu.

1.3.2.2.3 Controlador OpenFlow

ryu.controller.controller es el componente principal del controlador OpenFlow, el cual maneja las conexiones entre conmutadores, además de generar y enviar los eventos OpenFlow hacia las aplicaciones de Ryu.

Los siguientes componentes con sus respectivas funciones son descritos a continuación:

- **ryu.controller.dpset:** Realiza el manejo de los conmutadores.
- **ryu.controller.ofp_event:** Contiene las definiciones de los eventos OpenFlow.
- **ryu.controller.ofp_handler:** Realiza el manejo básico de OpenFlow, incluyendo la función de negociación.

1.3.2.2.4 Codificador y Decodificador OpenFlow

A continuación, se presentan los dos componentes para cada versión de OpenFlow, desde la versión 1.0 hasta la 1.5.

- **ryu.ofproto.ofproto_v1_x:** Contiene las definiciones de OpenFlow 1.x. Unos ejemplos de estas son: el número de puerto TCP a utilizar y los valores correspondientes a los puertos para realizar el reenvió de flujos hacia el plano de datos.
- **ryu.ofproto.ofproto_v1_x_parser:** Son módulos escritos en Python en donde se puede encontrar el codificador y decodificador para OpenFlow 1.x. En este componente se encuentran las clases y funciones necesarias para la creación de los mensajes OpenFlow.

  Además, este componente es utilizado para poder reenviar, procesar, modificar e instalar flujos en el *switch* OpenFlow, entre otras acciones.

1.3.2.2.5 Aplicaciones Ryu

Las aplicaciones que se instalan en conjunto con el controlador Ryu, pueden ser utilizadas como ejemplo para desarrolladores que empiezan a programar con Ryu o ser reutilizadas en nuevas aplicaciones en vista de que son archivos de texto plano escritos en Python. Entre las aplicaciones de ejemplo, que pueden ser modificadas, que vienen incorporadas en la instalación del software controlador Ryu se encuentran,

- **`ryu.app.simple_switch`:** Una implementación de un *switch* capa 2 que soporta OpenFlow 1.0.
- **`ryu.app.simple_switch_13`:** Una implementación de un *switch* capa 2 que soporta OpenFlow 1.3.
- **`ryu.app.rest`:** Un API para utilizar REST[29].

1.3.2.2.6 Librerías para el controlador Ryu

A continuación, se presentan varias librerías del controlador Ryu, que brindan grandes utilidades para el manejo de protocolos:

- **`ryu.lib.packet`:** Una implementación para codificar y decodificar paquetes de varios protocolos como por ejemplo TCP, UDP, IP, Ethernet, etc.
- **`ryu.lib.ovs`:** Una librería que contiene las clases y métodos necesarios para interactuar con el `ovsdb-server`.
- **`ryu.lib.of_config`:** Esta librería permite trabajar con Of-Config[30], ya que contienen las clases y métodos para utilizar este protocolo.
- **`ryu.lib.xflow`:** Una implementación para el manejo de los protocolos sFlow[31] y NetFlow[32].

1.3.2.2.7 Librerías de terceros

En esta sección se presenta librerías adicionales de Ryu, utilizadas en las definiciones de otras librerías mencionadas anteriormente, entre las cuales se tienen:

- **`ryu.contrib.ovs`:** Una librería que contiene clases y métodos necesarios para realizar las conexiones con Open vSwitch y que se utiliza para la implementación de la librería `ryu.lib.ovs`.
- **`ryu.contrib.oslo.config`:** Una implementación utilizada por las opciones en línea de comandos de ryu-manager y los archivos de configuración.

---

[29] REST (*Representational State Transfer*): Conjunto de convenciones en la interacción cliente-servidor sobre el protocolo HTTP.
[30] Of-Config: Protocolo especificado por la ONF usado para la configuración y control de los *switches*, en las redes definidas por software.
[31] sFlow: Protocolo que permite la recolección de información sobre el tráfico que se está manejando sobre la red.
[32] NetFlow: Protocolo de red desarrollado por Cisco Systems para recolectar información sobre tráfico IP, muy funcional en las SDN.

1.4 SOFTWARE DE SIMULACIÓN PARA SDN

En esta sección se presenta, el software emulador de SDN, Mininet [30], además de su extensión para redes inalámbricas Mininet-WiFi [31].

1.4.1 MININET

El software emulador de red Mininet se compone de una colección de hosts finales (los cuales ejecutan software LINUX estándar), conmutadores, enlaces y controladores ejecutándose sobre el kernel de LINUX. Mininet es utilizado para realizar pruebas rápidas con SDN y OpenFlow, y la interacción y manipulación de la red emulada se la realiza a través de una línea de comandos.

Con Mininet se pueden desarrollar y probar nuevas ideas antes de desplegarlas en un entorno real.

A continuación, se mencionan algunas características de Mininet:

- Basado en el lenguaje de programación Python.
- Envío de paquetes a través de interfaces reales Ethernet, con su velocidad de enlace y retardo.
- Es compatible con varios controladores SDN, como, por ejemplo, Ryu.

Dentro de las mejores características que posee este software se puede mencionar: la posibilidad de acceder a cada host individualmente, se puede acceder mediante una conexión remota, y cada host puede a su vez ejecutar individualmente las aplicaciones instaladas en distribuciones LINUX.

Mininet utiliza las funcionalidades que brinda la virtualización de un sistema operativo LINUX, permitiendo la creación de gran cantidad de nodos, ya sean estos *switches*, hosts o controladores en un solo computador.

1.4.2 MININET-WIFI

Como se mencionó en la sección 1.4.1, Mininet es muy útil para la emulación de redes que contengan dispositivos no inalámbricos, por eso y debido a la necesidad de añadir a Mininet la posibilidad de emular dispositivos inalámbricos como *access points y* estaciones, dentro de este contexto emerge Mininet-WiFi. Mininet-WiFi extiende la funcionalidad de Mininet añadiendo estaciones y *access points* virtualizados, utilizando los controladores de LINUX inalámbricos SoftMAC[33], la

[33] cfg80211: Componente de la arquitectura para redes inalámbricas de LINUX, que provee servicios de administración y configuración en el kernel, este componente viene instalado en las distribuciones de LINUX derivadas de Fedora y Debian, por lo cual el emulador es bastante compatible con la mayoría de estas distribuciones.

interfaz de configuración inalámbrica cfg80211[34] y el controlador para simulación inalámbrica mac80211_hwsim[35].

1.4.2.1 Arquitectura de Mininet-WiFi [32]

En la Figura 1.16 se puede apreciar la arquitectura de Mininet-WiFi, la cual se compone de los siguientes componentes:

- **MLME[36]:** Entidad localizada en el *kernel-space* para las estaciones y en el *user-space* actúa en conjunto con el demonio `hostapd`[37] para los AP.
- **Módulo mac80211_hwsim:** Se encuentra en el *kernel-space* y es el responsable de crear las interfaces virtuales WiFi necesarias para la operación de estaciones y *access points*.
- **iw, iwconfig y wpa_supplicant**: Son utilidades de LINUX, las dos primeras son utilizadas para configurar la interfaz y obtener información de las interfaces inalámbricas y la última es utilizada con `hostapd` para dar soporte WPA[38], entre otras tareas.
- **TC (*Traffic Control*):** Es una utilidad del tipo *user-space,* utilizada para configurar el organizador de paquetes del kernel de LINUX, el cual se encarga de controlar la velocidad, retardo, latencia y pérdida de paquetes, aplicando estos atributos en las interfaces virtuales de estaciones y AP, brindando un ambiente de simulación más próximo al real.
- Tanto estaciones como *access points* usan cfg80211 para comunicarse con el controlador del dispositivo inalámbrico, el API de configuración 802.11 provee comunicación entre las estaciones y mac80211[39].
- Esta arquitectura posee una comunicación directa entre el controlador del dispositivo WiFi a través de nl80211[40] que es utilizado para configurar el dispositivo cfg80211 y para la comunicación entre el *kernelspace* y el *userspace* [32].

---

[34] SoftMAC: Término utilizado para describir una NIC inalámbrica a la cual se pueda administrar via hardware.
[35] mac80211_hwsim: Módulo en kernel de LINUX utilizado para simular señales IEEE 802.11, muy útil en los casos que se requiere realizar pruebas en LINUX de una interfaz 802.11 sin utilizar una interfaz real.
[36] MLME (*Media Access Control Sublayer Management Entity*): Entidad de administración de la subcapa MAC de la arquitectura 802.11 que realiza funciones como autenticación, asociación, envío y recepción de *beacons,* entre otras.
[37] hostapd: Es un proceso *userspace* de LINUX utilizado para el manejo de AP y servidores de autenticación.
[38] WPA (*Wi-Fi Protected Access*): Sistema para proteger las redes inalámbricas (WiFi).
[39] mac80211: Componente de la arquitectura para redes inalámbricas de LINUX, que efectúa el manejo de las tareas asociadas con la MAC de dispositivos inalámbricos.
[40] nl80211: Componente de la arquitectura para redes inalámbricas de LINUX, que funciona como enlace entre el *userspace* y el *kernelspace*.

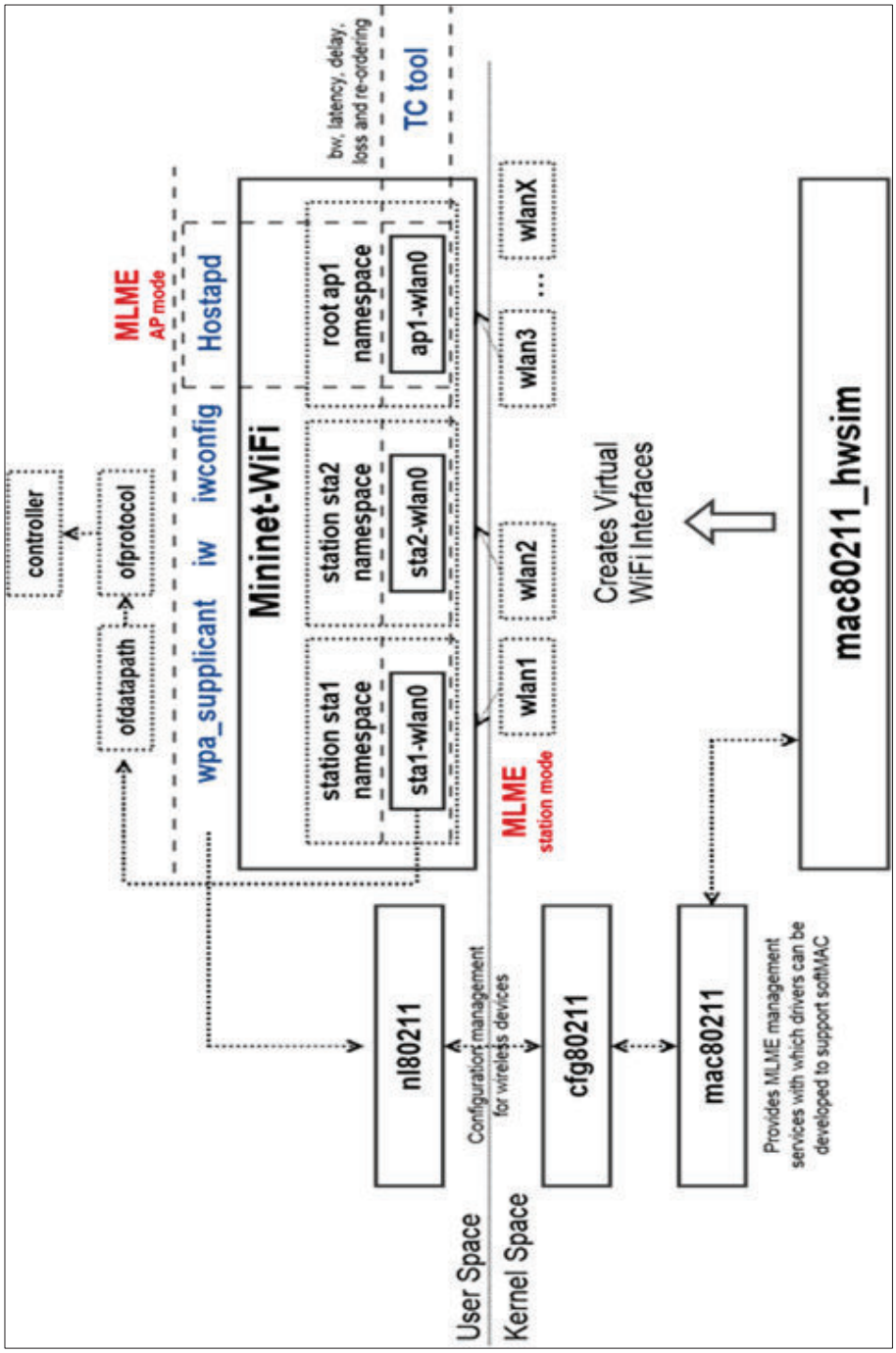

Figura 1.16. Arquitectura de Mininet WiFi [32]

CAPÍTULO 2: DISEÑO, SIMULACIÓN E IMPLEMENTACIÓN DEL PROTOTIPO

En este capítulo se presenta el diseño de la SDN a implementar. Se expone el proceso de instalación y la configuración de los sistemas operativos Raspbian y OpenWrt además del módulo Open vSwitch, todos estos elementos necesarios para el funcionamiento del prototipo. Se documenta el diseño de una pequeña aplicación de firewall utilizando el controlador Ryu y una interfaz gráfica de usuario para la configuración de los parámetros de la aplicación. Se presenta la implementación del prototipo tanto en simulación como en el prototipo físico.

2.1 ANÁLISIS DE REQUERIMIENTOS

En el prototipo a implementar se realizarán los procesos que se presentan a continuación en la Figura 2.1.

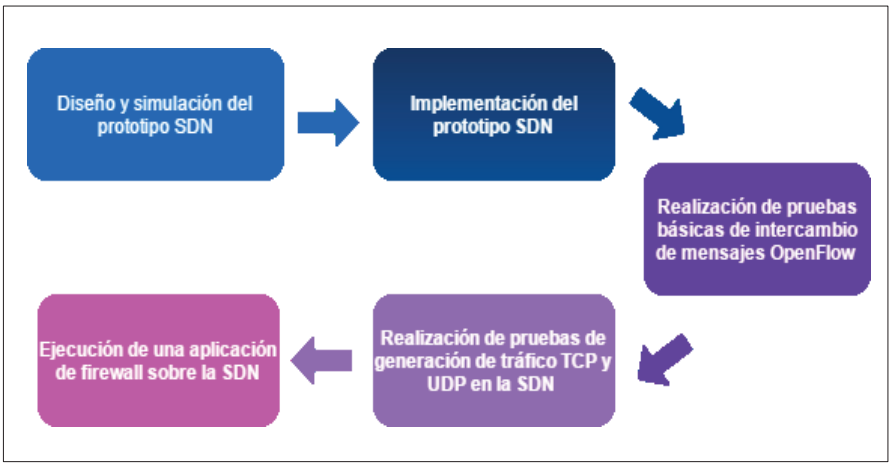

Figura 2.1. Procesos generales del prototipo

El prototipo de SDN a implementar, deberá cumplir los siguientes requerimientos:

a) Poseer dispositivos de interconectividad cableados (*switches*).
b) Disponer de dispositivos de interconectividad inalámbricos (*access points*).
c) Poseer un dispositivo controlador que realice el manejo de los dispositivos de interconectividad.
d) Todos los dispositivos (*switch*, AP y controlador) deberán ser de bajo costo y ser desarrollados sobre hardware y software libre.
e) La parte inalámbrica de la red, donde se ubican los AP, deberá tener los parámetros de una red de área local inalámbrica (WLAN) del tipo infraestructura.

f) Disponer de compatibilidad con el protocolo OpenFlow en todos los dispositivos de interconectividad de la red, además de compatibilidad en el dispositivo controlador.
g) Todos los dispositivos (*switch*, AP y controlador) deberán soportar el protocolo OpenFlow en su versión 1.3.0.
h) Enviar mensajes OpenFlow entre los dispositivos de interconectividad (*switch* y *access points*) y el dispositivo controlador para realizar la transmisión de datos en la red.
i) Ejecutar una aplicación de *firewall* en el dispositivo controlador.

2.2 ELEMENTOS DE LA SDN COMPUESTOS POR HARDWARE Y SOFTWARE

A continuación, se describirán los elementos compuestos por hardware y software que se utilizarán en la SDN, para satisfacer los requerimientos planteados.

2.2.1 SWITCH

Para el cumplimiento de los requerimientos: a), d), f) y h); el *switch* que se propone, está compuesto por una plataforma de hardware libre, como alternativa de bajo costo, además de adaptadores USB - Ethernet[41], para ampliar la capacidad de puertos de red, ya que la mayoría de plataformas de hardware libre cuenta con un solo puerto de este tipo.

Para el software, se propone utilizar una distribución basada en software libre, a la cual se le instalará un *switch* basado en software con el fin de agregarle compatibilidad con el protocolo OpenFlow en su versión 1.3.0.

En la Figura 2.2 se puede observar el diagrama de bloques del *switch* OpenFlow propuesto.

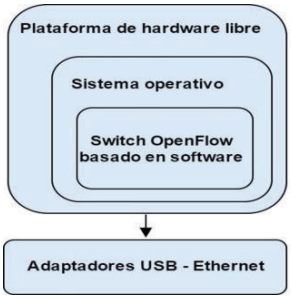

Figura 2.2. Diagrama de bloques de *switch* OpenFlow

[41] Adaptador USB-Ethernet: dispositivo que acondiciona un puerto USB en una tarjeta de red Ethernet.

2.2.2 ACCESS POINT

Para el cumplimiento de los requerimientos: b), d), f) y g); el *access point* que se propone, estará compuesto de manera similar al *switch* de la sección 2.2.1, con la diferencia de la necesidad de añadirle conectividad inalámbrica al dispositivo, mediante un adaptador USB-WiFi[42]. En la Figura 2.3, se puede apreciar el diagrama de bloques del *access point* OpenFlow propuesto.

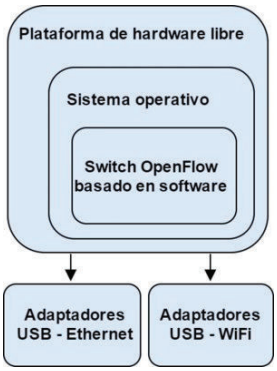

Figura 2.3. Diagrama de bloques de *access point OpenFlow*

2.2.3 CONTROLADOR

Para el cumplimiento de los requerimientos c), d), f) y g); el dispositivo controlador que se propone, estará conformado por una plataforma de hardware libre, una distribución basada en software libre y un controlador SDN basado en software, así el dispositivo podrá comunicarse mediante el protocolo OpenFlow con los demás dispositivos (*switch, access points*) y además permitirá aplicaciones de red sobre la SDN.

En la Figura 2.4, se puede observar el diagrama de bloques del dispositivo controlador OpenFlow propuesto.

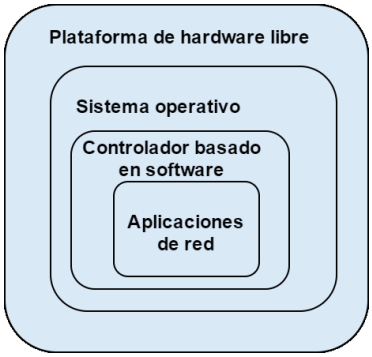

Figura 2.4. Diagrama de bloques del dispositivo controlador OpenFlow

[42] Adaptador USB-WiFi: Dispositivo que acondiciona un puerto USB en una tarjeta de red inalámbrica WiFi.

2.3 DISEÑO DEL PROTOTIPO DE LA SDN

En la Figura 2.5 se puede apreciar la topología de red básica del prototipo de SDN propuesto en el presente proyecto de titulación para cumplir con los requerimientos e) u i).

La red consta de un *switch,* dos *access points,* un controlador donde se tendrá la posibilidad de ejecutar aplicaciones de red sobre la SDN, como por ejemplo, una aplicación de *firewall*. Todos los dispositivos mencionados poseerán compatibilidad con el protocolo OpenFlow. Además, se contará con estaciones fijas e inalámbricas, según a que dispositivo de interconectividad se encuentren conectadas. La SDN en la parte inalámbrica de la red, donde se ubican los AP, tendrá los parámetros de una red de área local inalámbrica (WLAN) del tipo infraestructura.

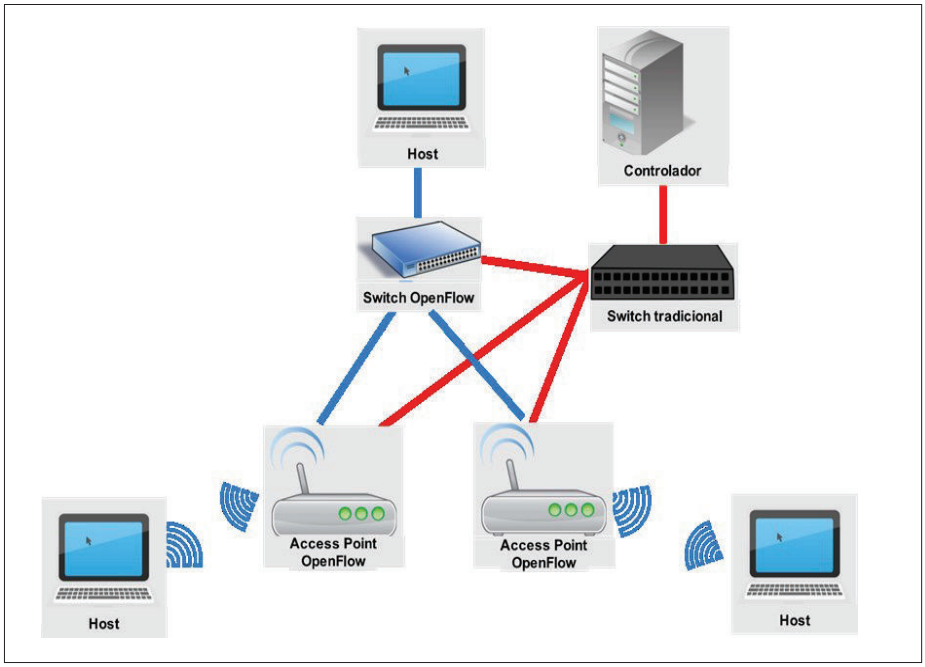

Figura 2.5. Esquema general del prototipo de SDN

2.3.1 PARÁMETROS INALÁMBRICOS DE LA SDN

2.3.1.1 Elección del canal inalámbrico de los AP de la SDN

Para la elección del canal inalámbrico que utilizará cada AP que formará parte de la SDN, se debe optar por canales los cuales sean los menos utilizados en el área donde se ubicarán los AP, con el fin de evitar la presencia de interferencia entre canales porque la misma disminuye el desempeño de la red inalámbrica. Se ha utilizado una herramienta de software que ayudará a realizar un escaneo de las redes WiFi cercanas y obtener la información de gran utilidad para la elección del

canal que utilizarán los AP en la SDN planteada. A continuación, en la Tabla 2.1 se presenta un resumen de la información obtenida por el software Wifi Analyzer[43] donde se pueden observar los canales inalámbricos más utilizados en las redes presentes en el área de donde se implementará la SDN.

Tabla 2.1. Canales más utilizados en el área

Número de canal utilizado	1	2	3	4	5	6	7	8	9	10	11	12	13
Total de número de veces utilizadas por un canal	6	0	0	0	3	3	1	0	1	1	11	0	0

Adicionalmente el software Wifi Analyzer proporciona una gráfica de los canales utilizados por las redes de la zona donde se implementará la red, como se muestra en la Figura 2.6

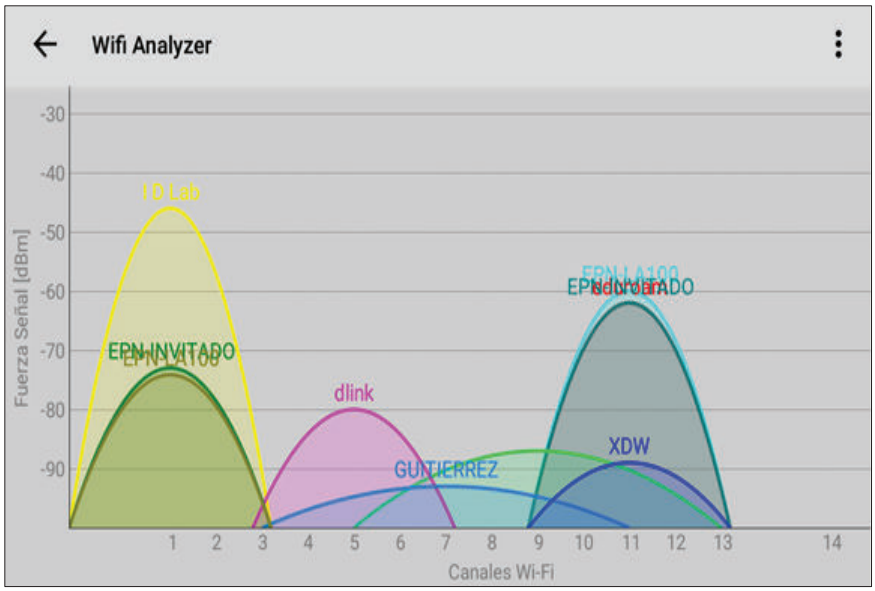

Figura 2.6. Captura de canales inalambricos ocupados en Wifi Analyzer

Una vez realizado el análisis de la información proporcionada por los dos tipos de software anteriormente mencionados, se ha determinado que se elegirán los canales 3 y 8 para cada AP respectivamente.

2.3.1.2 Parámetros inalámbricos para el diseño del prototipo SDN

A continuación, en la Tabla 2.2 se describen los parámetros de tipo inalámbrico de los *access points* que conforman la red propuesta.

[43] Wifi Analyzer: Aplicación gratuita de *smartphone* de para optimización de redes WiFi que permite conocer los canales ocupados por todas las redes que es capaz de captar el dispositivo.

Tabla 2.2. Parámetros inalámbricos de la red

Dispositivo	SSID	Esquema de seguridad	Contraseña	Modo de operación	Canal inalámbrico
AP_1	Red_AP_1	WEP o WPA	openflow12345	g	3
AP_2	Red_AP_2	WEP o WPA	openflow67890	g	8

2.3.2 DIRECCIONAMIENTO DE RED

Para el direccionamiento, se ha considerado que el prototipo está formado por: la red definida por software en sí y la red utilizada exclusivamente para conectar los dispositivos OpenFlow con el controlador.

2.3.2.1 Direccionamiento de red para la SDN

Tomando en cuenta que el objetivo del presente proyecto de titulación es el de demostrar la funcionalidad del prototipo de SDN inalámbrica, se ha determinado trabajar en un ambiente de red de área local inalámbrico (WLAN) del tipo infraestructura, además de la consideración que el prototipo se podría implementar en un laboratorio o aula con una capacidad máxima de 40 usuarios, se propone los siguientes requerimientos máximos para el direccionamiento IP:

- Número máximo de hosts fijos para el SW_1: 2.
- Número máximo de hosts inalámbricos para el AP_1: 19.
- Número máximo de hosts inalámbricos para el AP_2: 19.

Debido al pequeño número de direcciones IP requeridas, para realizar la distribución de direcciones de red se ha optado por utilizar una dirección IP de clase C 192.168.100.0/24. A continuación y luego de haber realizado un proceso de subneteo, en la Tabla 2.3 se presenta el direccionamiento IP a utilizar:

Tabla 2.3. Direccionamiento IP para la SDN

Nombre de Subred	Hosts requeridos	Hosts totales	Dirección	Mascara de red	Rango de direcciones IP asignables
AP_1	19	30	192.168.100.0	255.255.255.0	192.168.100.1 - 192.168.100.30
AP_2	19	30	192.168.100.32	255.255.255.0	192.168.100.33 - 192.168.100.62
SW_1	2	2	192.168.100.64	255.255.255.0	192.168.100.65 - 192.168.100.66

Es importante mencionar que para realizar pruebas sobre el prototipo bastará con utilizar un host por cada subred, es decir, solo se utilizarán tres hosts en total.

2.3.2.2 Direccionamiento de red para los dispositivos OpenFlow y el controlador

Por tratarse de una red de pequeña y para efectuar las conexiones de los dispositivos OpenFlow y el controlador, solo se utilizará una dirección de red por cada dispositivo.

Para evitar ambigüedades y no generar confusiones con la red de la sección 2.3.2.1, se utilizará un direccionamiento diferente.

La distribución de direcciones IP, a partir de la dirección IP de clase C 192.168.1.0/24 es la siguiente:

- Para el controlador la dirección IP: 192.168.1.2/24.
- Para el SW_1 la dirección IP: 192.168.1.3/24.
- Para el AP_1 la dirección IP: 192.168.1.4/24.
- Para el AP_2 la dirección IP: 192.168.1.5/24.

2.4 DISEÑO DE LA APLICACIÓN DE FIREWALL

En la presente sección se presentan las consideraciones de diseño que se utilizará en la aplicación de firewall sobre la SDN.

2.4.1 ANÁLISIS DE REQUERIMIENTOS DE LA APLICACIÓN DE FIREWALL

La aplicación de firewall sobre la SDN que se implementará, deberá cumplir los siguientes requerimientos:

a) La aplicación deberá poseer compatibilidad con OpenFlow versión 1.3.0.
b) Disponer en la aplicación de una REST API para responder las peticiones HTTP que se presenten mientras se ejecute la aplicación.
c) Las peticiones HTTP que lleguen hacia la aplicación serán utilizadas para: activar o desactivar el firewall, obtener información de la configuración del firewall e insertar o eliminar reglas de reenvío de tráfico en el firewall.
d) Las reglas de reenvío del tráfico en el firewall deberán basarse según las direcciones IP fuente y destino de los hosts a utilizarse.
e) Las reglas de reenvío de tráfico en el firewall deberán permitir o denegar el tráfico en la red de datos.
f) Las reglas de reenvío de tráfico en el firewall deberán basarse según el tipo de protocolo, los cuales pueden ser: ICMP, TCP o UDP.

2.4.2 FUNCIONAMIENTO DE LA APLICACIÓN DE FIREWALL SOBRE LA SDN

El funcionamiento básico de la aplicación de firewall sobre la SDN que se propone, se encuentra resumido en el diagrama de flujo, que se muestra en la Figura 2.7. Adicionalmente se desarrollará una interfaz gráfica de usuario la cual se empleará para configurar el firewall de una manera más rápida e intuitiva.

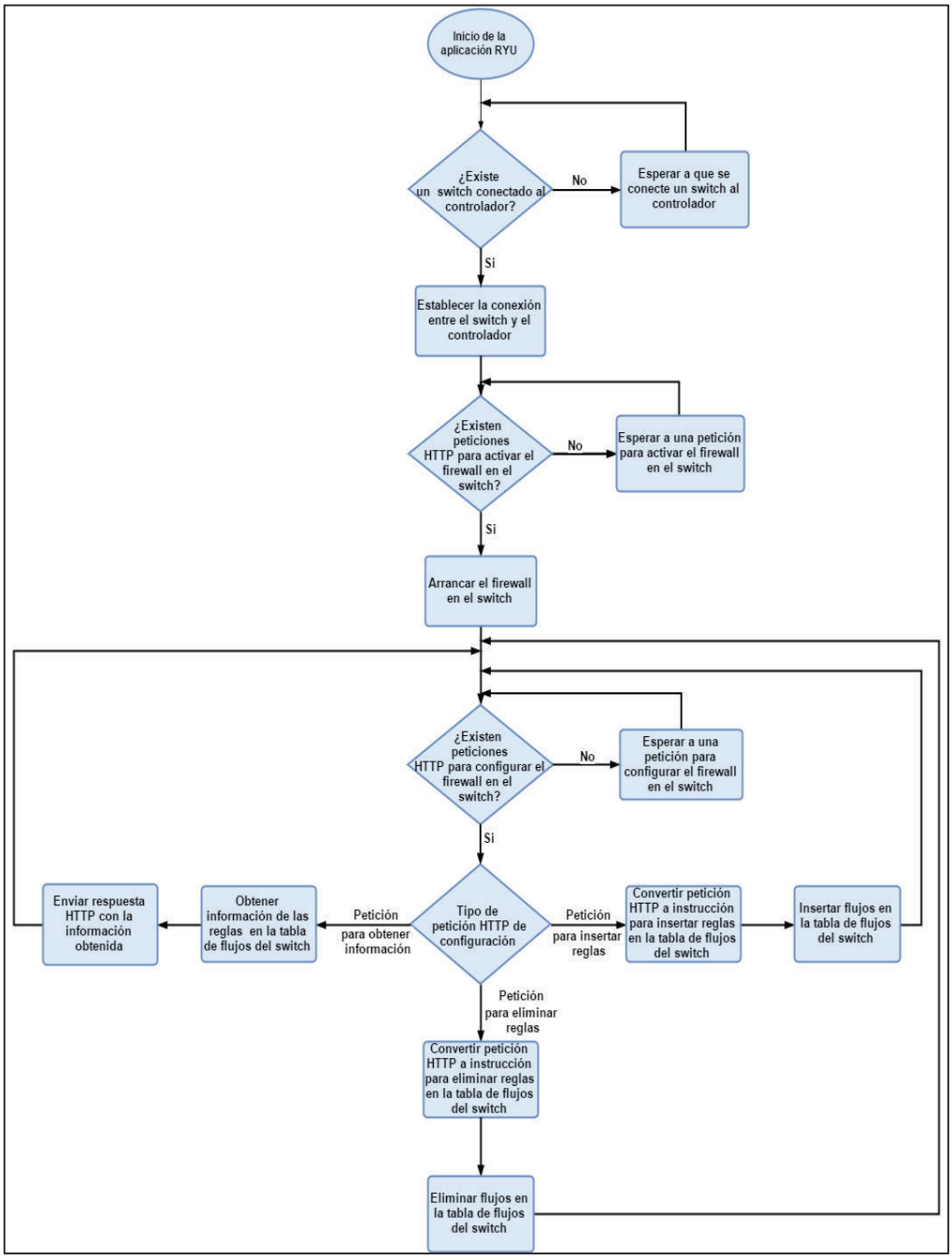

Figura 2.7. Diagrama de flujo de la aplicación de firewall

2.5 SIMULACIÓN DEL PROTOTIPO DE LA SDN

Para la simulación de una SDN, resulta bastante útil emplear el paquete de software libre Mininet, el cual está escrito en el lenguaje de programación Python y posibilita la prueba de SDN antes de que se implemente con equipos físicos [32].

Debido a que en Mininet no es posible simular dispositivos inalámbricos como *access points* y estaciones que se conecten a ellos, un grupo de investigadores de la Universidad de Campinas en Brasil desarrolló una extensión de Mininet llamada Mininet-WiFi [32]. Esta extensión añade nuevos archivos y clases para soportar la emulación de dispositivos y enlaces inalámbricos sobre una SDN con el protocolo OpenFlow.

2.5.1 INSTALACIÓN DE MININET-WIFI

El software por defecto, es compatible con las distribuciones de LINUX, Ubuntu, Debian y Fedora, entre otras, siempre que cumplan con los siguientes requerimientos:

- Kernel LINUX compilado con soporte de *namespace*[44] de red habilitado.
- Compatibilidad con un *switch* basado en software como Open vSwitch o el LINUX bridge.
- Compatibilidad con Python, bash[45], ping, iperf[46], entre otros paquetes de software.
- Permiso de usuario root [47](necesario para el acceso a las interfaces de red del computador).

Para la instalación de Mininet-WiFi se debe utilizar el Comando 2.1, el cual instala el software **git** y descarga el archivo que contiene todo el software, para posteriormente instalarlo a partir del script **`util/install.sh`.**

```
$ sudo apt-get install git
$ git clone https://github.com/intrig-unicamp/mininet-wifi
$ cd mininet-wifi
$ sudo util/install.sh –Wnfv
```

Comando 2.1. Instalación de Mininet-WiFi

---

[44] *namespace*: Medio para organizar clases dentro de un entorno, agrupándolas de un modo más lógico y jerárquico.

[45] bash: Programa informático, cuya función consiste en interpretar órdenes, y un lenguaje de programación de consola.

[46] iperf: Herramienta de software para medir el ancho de banda y la calidad de un enlace de red de datos. La herramienta es multiplataforma es decir para sistemas LINUX, Windows y MAC.

[47]root: En distribuciones LINUX, nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (monousuario o multiusuario). Normalmente es la cuenta de administrador.

2.5.2 ARCHIVOS AÑADIDOS EN MININET-WIFI

A continuación, se describen los archivos que fueron añadidos por Mininet-WiFi a la implementación original de Mininet:

- **`mininet/wifiAccessPoint.py`:** Contiene una implementación completa de un *access point*.
- **`mininet/wifiAdHocConnectivity.py`:** Realiza la comprobación de conectividad entre miembros de una red Ad-Hoc.
- **`mininet/wifiAssociationControl.py`:** Contiene técnicas de control para el proceso de asociación.
- **`mininet/wifiChannel.py`:** Contiene parámetros del canal inalámbrico.
- **`mininet/wifiDevices.py`:** Contiene especificaciones, provenientes de *datasheets*, de algunos dispositivos reales.
- **`mininet/wifiMeshRouting.py`:** Contiene los parámetros para emular redes inalámbricas en malla.
- **`mininet/wifiMobility.py`:** Posee los parámetros para emular movilidad inalámbrica.
- **`mininet/wifiMobilityModels.py`:** Contiene varios modelos de movilidad inalámbrica.
- **`mininet/wifiModule.py`:** Contiene el módulo principal para la simulación de nodos inalámbricos.
- **`mininet/wifiPlot.py`:** Posibilita el graficar la red.
- **`mininet/wifiPropagationModels.py`:** Contiene varios modelos de propagación.
- **`mininet/wifiReplaying.py`:** Replica simulaciones realizadas con anterioridad.
- **`mininet/vanet.py`:** Contiene los parámetros para emular escenarios de tipo VANET[48].

Cabe mencionar que Mininet-WiFi es una extensión de Mininet, el desarrollador de Mininet-WiFi no modificó ninguna funcionalidad de Mininet, solo se agregaron nuevas funcionalidades.

---

[48] VANET (Vehicular Ad-Hoc Network): Tipo de red de comunicación que utiliza vehículos como nodos de la red.

2.5.3 COMANDOS BÁSICOS DE MININET-WIFI

En esta sección se presentan algunos comandos básicos presentes en Mininet-WiFi.

2.5.3.1 Creación de una SDN en Mininet-WiFi

Para crear una red básica de dos estaciones y un *access point* se puede utilizar el Comando 2.2, que se puede apreciar a continuación:

```
$ sudo mn –wifi
```

Comando 2.2. Creación de una red básica en Mininet-WiFi

En el Comando 2.3 se observa un ejemplo de cómo el Comando 2.2 puede ser utilizado con varias opciones, que establecerán parámetros básicos en una red WiFi.

```
$ sudo mn --wifi --ssid=red_prueba --mode=g --channel=11
```

Comando 2.3. Creación de una red con parámetros en Mininet-WiFi

Donde:

--ssid: Opción para establecer el SSID de la red, también conocido como nombre de la red inalámbrica.

--mode: Opción para establecer el modo de transmisión de datos(modo a ,b ,g ,n ,ac) que actuará la red inalámbrica, los modos disponibles en el software son: modo b, modo g y modo n.

--channel: Con esta opción se establece el canal inalámbrico donde operará la red inalámbrica.

2.5.3.2 Cambios de tipo de topologías y número de estaciones en Mininet-WiFi

Como se mencionó en la sección anterior la topología por defecto es un *access point* conectado a dos estaciones. Es posible cambiar el tipo de topología y el número de estaciones que contendrá está topología, utilizando la opción **--topo**.

A continuación, se observa el Comando 2.4 utilizado para crear una red con una topología del tipo centralizada y con cinco estaciones, es decir, cinco estaciones conectadas a un solo *access point*.

```
$ sudo mn --wifi --topo single,5
```

Comando 2.4. Creación de una red con topología centralizada

En el Comando 2.5 se utiliza una topología del tipo lineal y tres estaciones, es decir la red se compone de tres *access points,* cada uno conectado a una estación y todos los *access points* conectados en línea por un enlace cableado.

```
$ sudo mn --wifi --topo linear,3
```

Comando 2.5. Creación de una red con topología lineal

2.5.3.3 Obtención de información de nodos en Mininet-WiFi

Una vez que la red se encuentre en funcionamiento, se puede obtener información de los nodos (estaciones, *access points y switches*) ejecutando los comandos que se describen en la Tabla 2.4.

Tabla 2.4. Comandos para la obtención de información de nodos en Mininet-WiFi

Comando	Información
`mininet-wifi>info [nombre de la estación]`	Información básica de una estación.
`mininet-wifi>py [nombre de la estación].params['posición']`	Información de la posición dada en coordenadas x-y de una estación.
`mininet-wifi>py [nombre de la estación].params['associatedTo']`	Información del AP a que está asociada una estación especifica.
`mininet-wifi>py [nombre de la estación].params['apsInRange']`	Información de los *access points* que se encuentren en rango de una estación.
`mininet-wifi>py [nombre de la estación].params['channel']`	Información del canal inalámbrico en que opera una estación.
`mininet-wifi>py [nombre de la estación].params['frequency']`	Información de la frecuencia en que opera una estación.
`mininet-wifi>py [nombre de la estación].params['mode']`	Información del modo en que opera una estación.
`mininet-wifi>py [nombre de la estación].params['txpower']`	Información de la potencia de transmisión de una estación.
`mininet-wifi>py [nombre de la estación].ssid`	Información del SSID de la red a la cual está asociada una estación.
`mininet-wifi>py [nombre del AP ].associatedStations`	Información de las estaciones que están asociadas a un AP.

2.5.3.4 Modificación de parámetros durante la ejecución de Mininet-WiFi

Una vez se encuentre establecida y funcionando una red en Mininet-WiFi, se pueden modificar parámetros y configuraciones de los nodos (estaciones, *access points y switches*) ejecutando los comandos que se describen en la Tabla 2.5.

Tabla 2.5. Comandos para la modificación de parámetros en Mininet-WiFi

Comando	Acción
`mininet-wifi>py [nombre de la estación].moveStationTo('[posición]')`	Modifica la posición de una estación.
`mininet-wifi>py [nombre de la estación]. setAntennaGain('[interfaz de red de estación], [valor de ganancia])`	Establece el valor de ganancia de la antena de una interfaz de red de una estación
`mininet-wifi>py [nombre de la estación]. setRange([valor de rango de cobertura de señal])`	Modifica el rango de cobertura de señal de una estación.

2.5.4 VND (versión SDN) [33]

Visual Network Description – VND (versión SDN) es una aplicación web con interfaz gráfica, la misma posibilita la generación de topologías y escenarios de redes de forma gráfica utilizando archivos NSDL[49], permitiendo simulación y análisis de diferentes escenarios de red. VND (versión SDN) puede exportar archivos compatibles para ejecutar en Mininet y en controladores OpenFlow. Estos archivos pueden contener escenarios y topologías de red, tablas de flujo y configuración de QoS.

Actualmente VND soporta la exportación de archivos para ser ejecutados en extensiones de Mininet, tales como Mininet-WiFi.

La aplicación web está disponible en el sitio web de su autor, el enlace a la aplicación web VND (versión SDN) es el siguiente: **http://www.ramonfontes.com/vnd**.

En la Figura 2.8 se puede apreciar, la interfaz gráfica para crear una SDN, con VND. Para el ejemplo se ha creado una red que está conformada por un controlador, un *access point* y dos estaciones inalámbricas.

[49] NSDL (*Network Scenario Description Language*): Framework que provee un lenguaje o conjunto de reglas, para realizar una descripción de redes cableadas o inalámbricas e información de las condiciones donde se evaluará dichos tipos de redes.

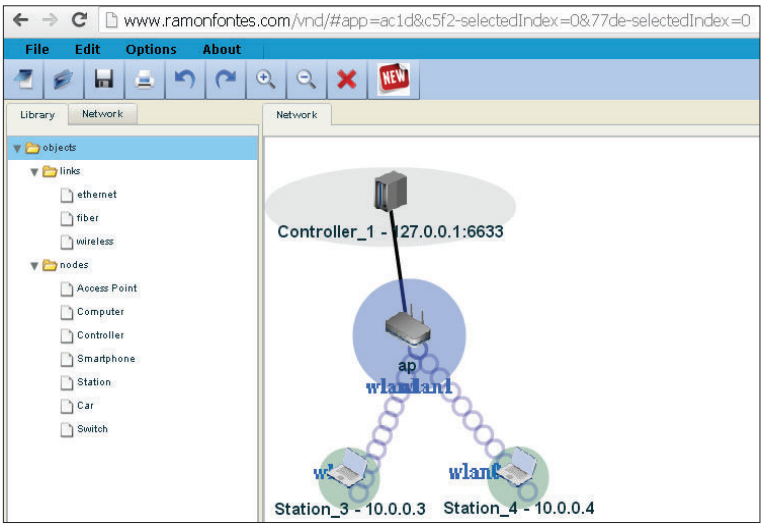

Figura 2.8. Interfaz gráfica de VND

En la Figura 2.9 por su parte, se puede observar un ejemplo de configuración de forma gráfica de un *access point*, en este tipo de dispositivo se pueden configurar los siguientes parámetros: nombre del AP, dirección MAC, SSID, canal inalámbrico, modo de transmisión de datos(modo a ,b , g ,n y ac), rango de cobertura de señal inalámbrica en metros, tipo de *switch* OpenFlow(*switch* OVS y *switch* cpqd[50]), versión de protocolo OpenFlow(1.0, 1.1, 1.2, 1.3 y 1.4) y posición del AP.

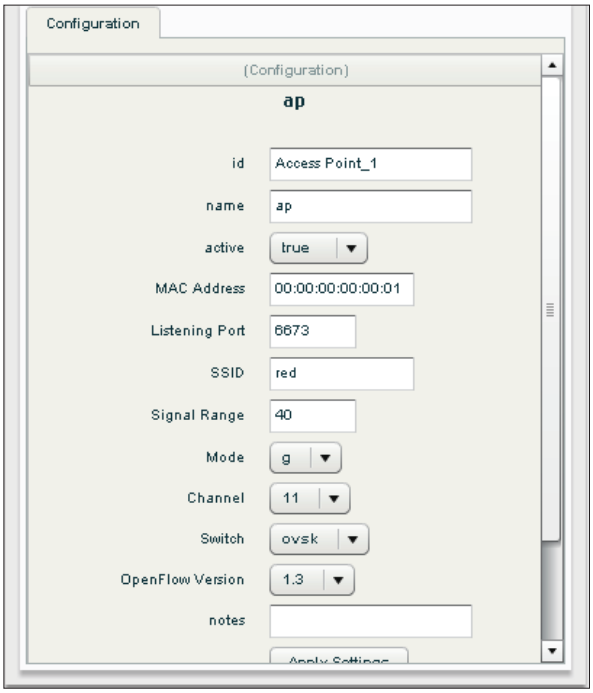

Figura 2.9. Ejemplo de configuración de forma gráfica de un *access point* en VND

[50] cpqd: *switch* basado en software compatible con OpenFlow 1.3.0.

Posterior a la creación y configuración de la red, VND ofrece la opción de exportar la red creada en un archivo compatible con Mininet-WiFi. El archivo es generado automáticamente por VND y constituye la equivalencia en código de la topología de red de la Figura 2.8.

Luego de la creación del archivo que lleva por nombre **`mininetCode83987.sh`**, cabe señalar que pese a que no posee la extensión **`.py`** de un archivo Python este se puede ejecutar mediante el Comando 2.6.

```
$ python [nombre del script]
```

Comando 2.6. Ejecución de un archivo Python

Al ejecutar el Comando 2.6 la salida producida es la que se presenta por pantalla en la Figura 2.10.

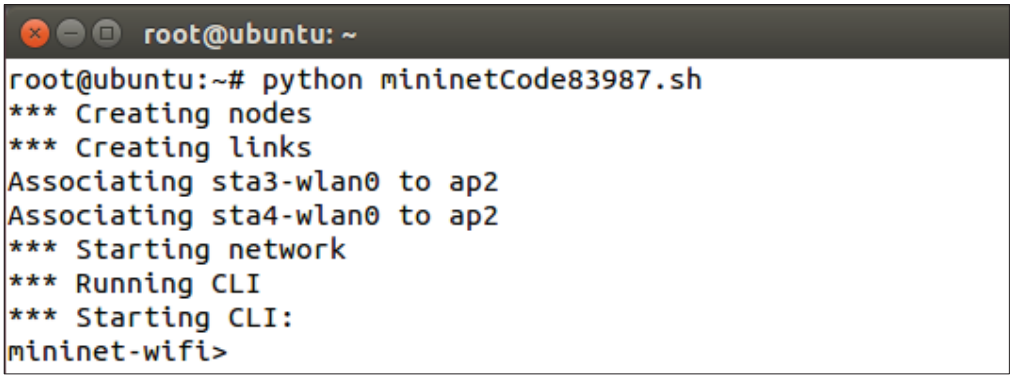

Figura 2.10. Ejecución del script generado por VND **`mininetCode83987.sh`**

Al iniciar la simulación se abrirá una ventana gráfica de Mininet-WiFi, misma que se puede observar en la Figura 2.11, donde se encuentra la posición y además el rango de cobertura de cada uno de los elementos de red.

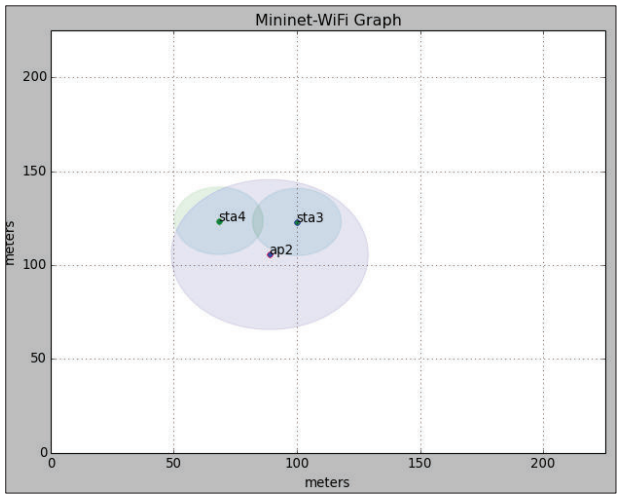

Figura 2.11. Ventana gráfica de Mininet-WiFi para el ejemplo de la red simulada

2.5.5 SIMULACIÓN DE LA SDN DISEÑADA

De acuerdo a los criterios de diseño presentados en la sección 2.3, para la simulación de la SDN, se consideran los parámetros de red inalámbricos y el direccionamiento IP de la red presentes en la sección 2.3.1 y 2.3.2.1 respectivamente.

Adicionalmente para la simulación se utilizará la ecuación para el cálculo de propagación FRIIS[51] el cual es utilizado para simular los ambientes WLAN en interiores.

En la Figura 2.12, se puede observar el diagrama de flujo del proceso de simulación de la red en Mininet-WiFi.

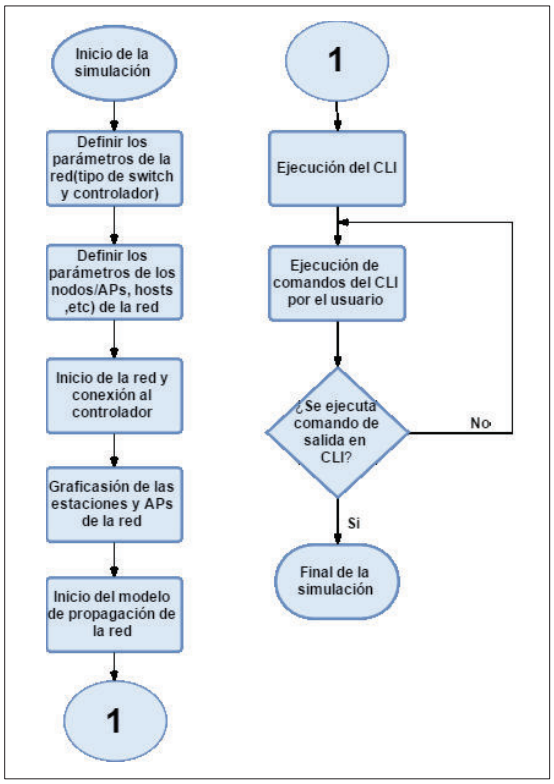

Figura 2.12. Diagrama de flujo de la simulación de la red

2.5.5.1 Explicación del código para simular la SDN diseñada

A continuación, se realiza una breve explicación, de las líneas del Código 2.1 , Código 2.2, Código 2.3 y Código 2.4, pertenecientes al archivo **`tesisSDN.py`** creado para la simulación en Mininet-WiFi, el código completo se encuentra en el **Anexo A**.

[51] FRIIS: Ecuación que se utiliza para calcular la potencia recibida desde una antena, cuando se transmite desde otra antena separadas por una distancia y que operan a una frecuencia o de longitud de onda.

```
1 from mininet.net import Mininet
2 from mininet.node import  Controller, RemoteController, OVSKernelSwitch
3 from mininet.cli import CLI
4 from mininet.log import setLogLevel
5 from mininet.link import TCLink
6 #Definir los parametros de la red, como el tipo de enlace, controlador y tipo de
  switch.
7 def topology():
8    net = Mininet( controller=Controller, link=TCLink, switch=OVSKernelSwitch  )
```

Código 2.1. Primera sección del archivo **`tesisSDN.py`** para simulación en Mininet-WiFi

Línea 1-5: Declaración de los módulos y librerías básicas que se utilizaran en la ejecución del script**,** entre las cuales se tienen: **`Mininet`**, **`Controlller`**, o**`VSKernelSwitch`**, **`CLI`**, entre otros.

Línea 7-8: Se define los parámetros básicos de la SDN, como por ejemplo: tipo de *switch* que se utilizará, que en este caso será el Open vSwitch Kernel, y qué modelo de enlace se simulará, que en este caso será TCLink[52].

```
10   #Definir los parametros del AP_1 como SSID, modo, canal, posicion, rango,
  protocolo, password y encriptacion.
11   ap1 = net.addAccessPoint( 'ap1', ssid="Red_AP_1", mode="g", channel=3,
  position='1.625,8.5,1.5',range='10', protocols='OpenFlow13',
  passwd='openflow12345', encrypt='wep', mac='00:00:00:00:00:02')
12   #Definir los parametros del AP_2 como SSID, modo, canal, posicion, rango,
  protocolo, password y encriptacion.
13   ap2 = net.addAccessPoint( 'ap2', ssid="Red_AP_2", mode="g", channel=8,
  position='5,8.5,1.5', protocols='OpenFlow13',range='10',
  passwd='openflow67890', encrypt='wep', mac='00:00:00:00:00:03')
14   #Definir los parametros de  sta3, numero de interfaces, posicion, direccion IP,
  rango, password y encriptacion.
15   sta3 = net.addStation( 'sta3', wlans= 1,position='1.625,1,1',
  ip="192.168.100.2/24",range='10', passwd='openflow12345', encrypt='wep' )
16   #Definir los parametros de  sta4, numero de interfaces, posicion, direccion IP,
  rango, password y encriptacion.
17   sta4 = net.addStation( 'sta4', wlans= 1,position='5,1,1',
  ip="192.168.100.34/24",range='10',passwd='openflow67890', encrypt='wep')
18   h5 = net.addHost( 'h5', ip="192.168.100.65/24" )
19   #Definir los parametros del switch y como protocolo.
20   s6 = net.addSwitch( 's6', protocols='OpenFlow13', mac='00:00:00:00:00:01')
21   #Definir los parametros del controlador como la conexion remota, direccion IP y
  puerto.
22   c0 = net.addController('c0', controller=RemoteController, ip='127.0.0.1',
  port=6633 )
```

Código 2.2. Segunda sección del archivo **`tesisSDN.py`** para simulación en Mininet-WiFi

Línea 11-13: Se crean y configuran los AP SDN de acuerdo al diseño propuesto y se establece la versión del protocolo OpenFlow que se va a utilizar, que en este caso es la versión 1.3.0, adicionalmente se configuran los parámetros inalámbricos de red como el SSID, el modo de transmisión de datos, el canal, el esquema de seguridad y el *password* de la red, así como la posición espacial donde se ubicará en la simulación.

[52] TCLink: Clase que implementa un enlace para el canal de transmisión de las interfaces de red en el software Mininet.

Es importante señalar que se utilizó el esquema de seguridad WEP, debido a que el *switch* OpenFlow utilizando Open vSwitch, presenta un problema de incompatibilidad con otros esquemas de seguridad como WPA o WPA2 [34].

Línea 15-18: Se crean y configuran las estaciones fijas e inalámbricas con el direccionamiento IP.

En el caso de las estaciones inalámbricas se configuraron parámetros adicionales como posición espacial que tendrá en la simulación, número de interfaces inalámbricas y el *password* que utilizará para asociarse a los AP, que en este caso estará cifrado con el esquema de seguridad WEP.

Línea 20: Se crea y configura el *switch* SDN de acuerdo al diseño propuesto y se establece la versión del protocolo OpenFlow que se va a utilizar que este caso es la versión 1.3.0.

Línea 22: Se crea y configura el controlador con los parámetros de dirección IP y puerto.

```
27  #Creacion de enlaces cableados e inalambricos de la red
28  net.addLink(h5, s6, bw=100)
29  net.addLink(ap1, s6, bw=100)
30  net.addLink(ap2, s6, bw=100)
31  net.addLink(ap1, sta3)
32  net.addLink(sta4, ap2)
33  #Conexion de los APs y switch al controlador
34  print "*** Iniciando la red"
35  net.build()
36  c0.start()
37  s6.start( [c0] )
38  ap1.start( [c0] )
39  ap2.start( [c0] )
```

Código 2.3. Tercera sección del archivo **`tesisSDN.py`** para simulación en Mininet-WiFi

Línea 28-30: Se crean los enlaces del *switch* SDN a los *access points*, además se establece la velocidad del enlace cableado a 100Mbps, velocidad utilizada en redes Fast Ethernet.

Línea 31-32: Se realiza la asociación de las estaciones inalámbricas a los AP.

Línea 35: Se configura la creación de la red, a partir de todos los parámetros previamente configurados en el archivo Python.

Línea 36-39: Se establece el arranque del *switch*, AP y controlador, además de establecer las conexiones de los dispositivos OpenFlow al controlador, la conexión deberá ser de cada uno de los dispositivos hacia el controlador.

```
41    net.seed(12)
42    net.plotGraph(max_x=6.5, max_y=8.5 , max_z=4)
43    #Definir el tipo de propagacion que se utilizara
44    net.propagationModel("friisPropagationLossModel", sL = 2)
45    #Instanciacion de la CLI
46    print "*** Ejecutando CLI"
47    CLI( net )
48    #Metodo para detener la red
49    print "*** Deteniendo la red"
50    net.stop()
51  #Instanciacion de toda la simulacion, a partir del metodo main
52  if __name__ == '__main__':
53    #Metodo para visualizar los mensajes de la red
54    setLogLevel( 'info' )
55    #Creacion de la red
56    topology()
```

Código 2.4. Cuarta sección del archivo **`tesisSDN.py`** para simulación en Mininet-WiFi

Linea 42: Se configuran los valores espaciales de la ventana gráfica de Mininet-WiFi, ventana que contendrá los dispositivos inalámbricos.

Linea 44: Se establece el modelo de propagación a utilizar en la simulación, el cual en este caso es el modelo de FRIIS.

Linea 47: Se instancia el CLI de Mininet-WiFi.

Linea 50: Se establece el método para detener o finalizar la simulación de la red.

Linea 52-56: Además se presenta la definición del método **`main`** necesario para iniciar la simulación de la SDN y el nivel de información que se presentará por pantalla.

2.5.5.2 Inicio de la simulación de la SDN diseñada

Para la simulación se debe ejecutar el Comando 2.6 con el nombre del archivo que se guardó al momento de crear el script de la sección 2.5.5.1 que en este caso es **`tesisSDN.py`**, para obtener una salida por pantalla como se observa en la Figura 2.13

```
root@ubuntu: /home/ubuntu
root@ubuntu:/home/ubuntu# python tesisSDN.py
*** Creando los nodos(estaciones fijas e inalambricas
ontrolador)***
*** Configurando nodos WiFi
*** Creando enlaces
(100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (
sociating sta3-wlan0 to ap1
Associating sta4-wlan0 to ap2
*** Iniciando la red
*** Configuring hosts
(100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (
CLI
*** Starting CLI:
mininet-wifi> xterm sta3 sta4 h5
```

Figura 2.13. Ejecución de la simulación del prototipo SDN

En la Figura 2.14, se puede observar la ventana gráfica de Mininet-WiFi que contiene una representación de los elementos de la parte inalámbrica de la red simulada, es decir dos *access points* y dos estaciones inalámbricas.

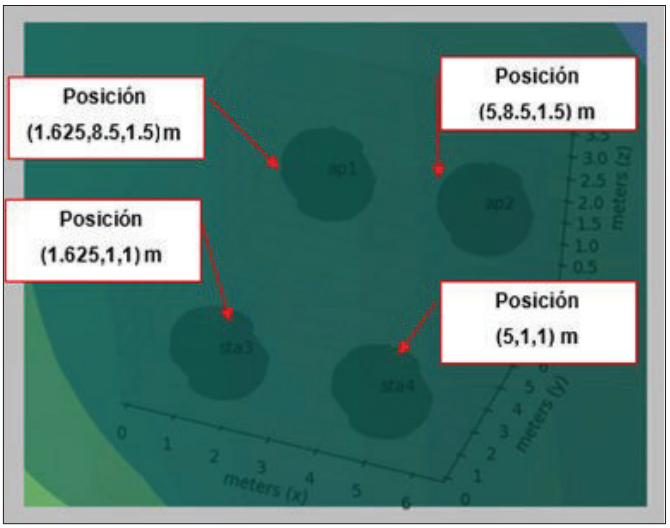

Figura 2.14. Venta gráfica de Mininet-WiFi con elementos inalámbricos de la red simulada

Con el software xterm[53], se puede acceder a la configuración de cada elemento de la red simulada. A continuación, se presenta el Comando 2.7, con el cual se puede acceder a los distintos dispositivos de red, en este caso un host fijo(h5) y dos estaciones inalámbricas (sta3 y sta4).

```
mininet-wifi> xterm h5 sta3 sta4
```

Comando 2.7. Acceso mediante xterm a "h5," sta3" y "sta4"

Luego de la ejecución del comando existirá la posibilidad de revisar la configuración de cada equipo simulado. En la Figura 2.15 se puede observar, por ejemplo, el direccionamiento IP del host "h5", en tanto que en la Figura 2.16 y Figura 2.17 se puede observar el direccionamiento IP de la estación ""sta3 y "sta4", respectivamente.

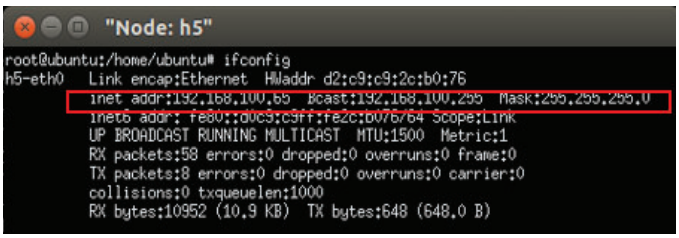

Figura 2.15. Configuración IP del host "h5"

[53] xterm: Emulador de terminal para el sistema de ventanas X Window System.

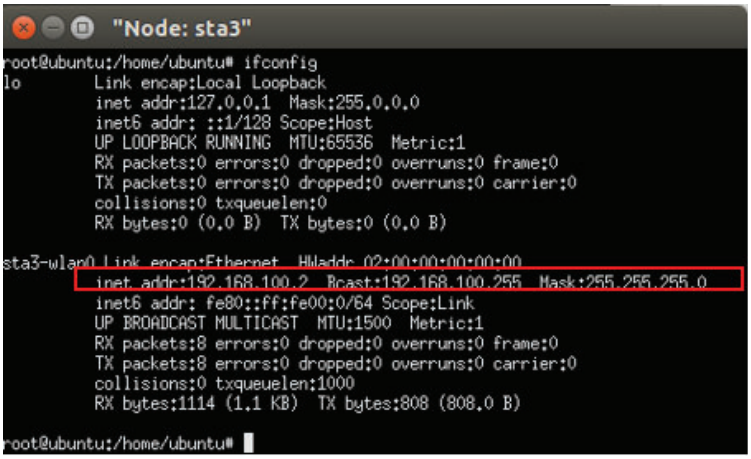

Figura 2.16. Configuración IP de la estación “sta3”

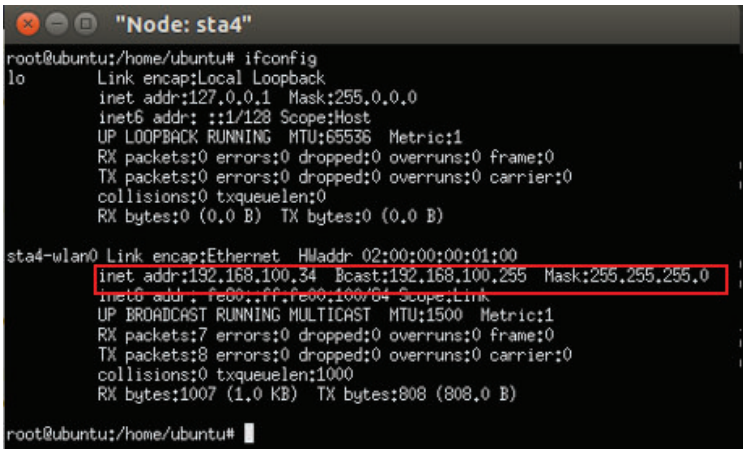

Figura 2.17. Configuración IP de la estación “sta4”

También es posible revisar la configuración inalámbrica de las estaciones “sta3” y “sta4”, en la Figura 2.18 y Figura 2.19 se muestran parámetros como el SSID de la red a la cual está asociada, la dirección MAC del *access point* a la cual se encuentra asociada la estación, y más parámetros como la tasa de trasmisión de datos, potencia de trasmisión, nivel de señal, entre otros.

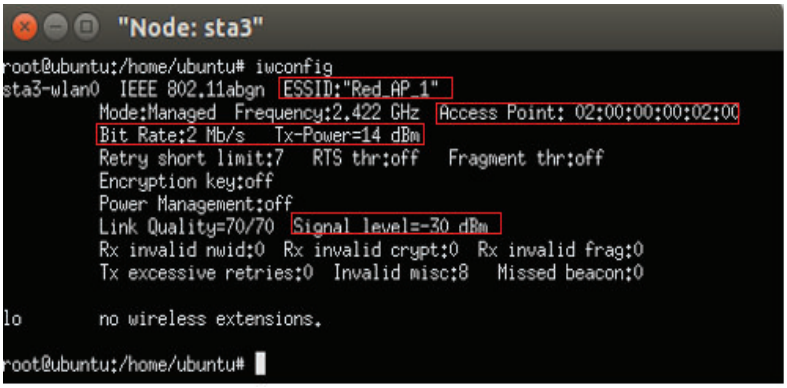

Figura 2.18. Configuración inalámbrica de red de la estación “sta3”

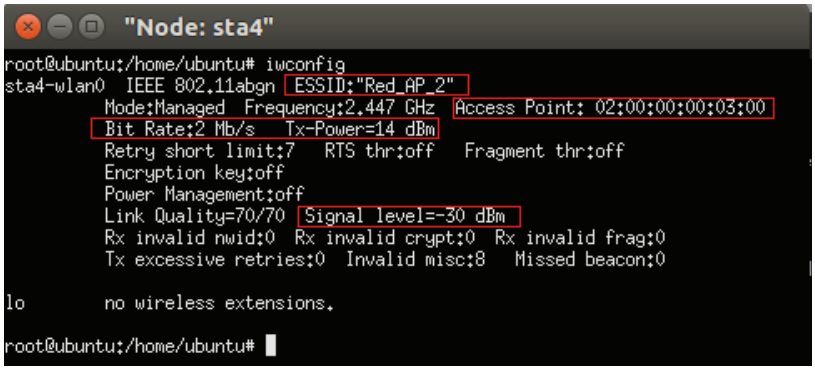

Figura 2.19 Configuración inalámbrica de red de la estación "sta4"

En cuanto a la obtención de más parámetros de red inalámbrica, de cómo por ejemplo, canal de trasmisión, esquema de seguridad utilizado, entre otros se pueden observar en la Figura 2.20 y Figura 2.21 para cada red simulada correspondientemente.

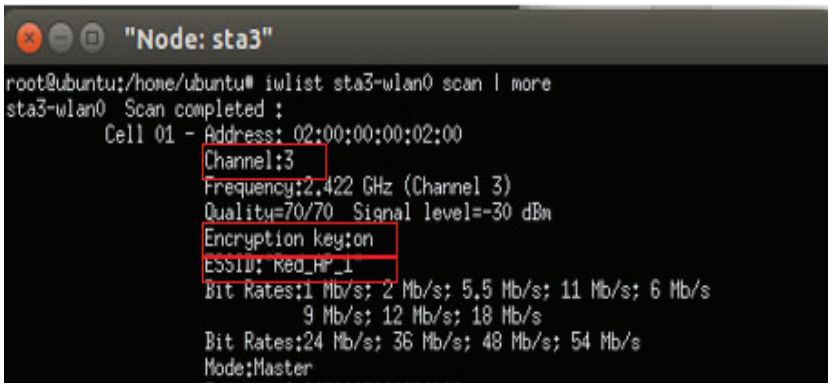

Figura 2.20. Configuración de la red inalámbrica "Red_AP_1"

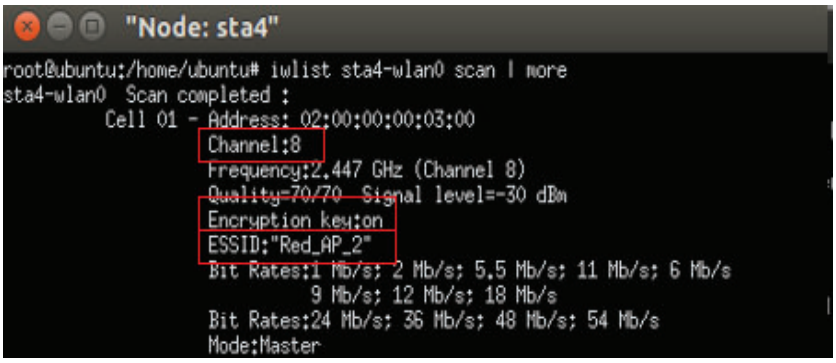

Figura 2.21. Configuración de la red inalámbrica "Red_AP_2"

2.6 IMPLEMENTACIÓN DEL PROTOTIPO DE SDN

En la presente sección se presentarán las distintas etapas de la implementación del prototipo de SDN, siguiendo el diseño propuesto en la sección 2.3 y utilizando los componentes de hardware y software mencionados en la sección 2.2.

2.6.1 IMPLEMENTACIÓN DEL SWITCH OPENFLOW EN LA PLACA RASPBERRY PI

Para el *switch* se utilizó, la plataforma de hardware libre, Raspberry Pi 2 modelo B porque se determinó que las características de este modelo, descritas en la Tabla 1.2, son suficientes para que esta plataforma pueda actuar como *switch*. Cabe indicar que se requirió adicionalmente de la utilización de adaptadores USB – Ethernet, para ampliar la capacidad de puertos de red ya que la Raspberry Pi cuenta con un solo puerto de este tipo.

Para el software, se utilizó la distribución de LINUX Raspbian OS, la cual es compatible con la Raspberry Pi, y en la cual se le instaló el *switch* basado en software Open vSwitch 2.5.0, con el fin de agregar la compatibilidad con el protocolo OpenFlow.

En la Figura 2.22 se puede observar el diagrama de bloques del *switch* OpenFlow implementado.

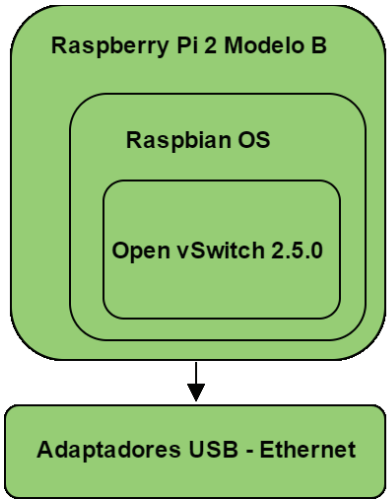

Figura 2.22. Diagrama de bloques de *switch* OpenFlow implementado

Para la implementación del *switch* OpenFlow sobre la placa Raspberry Pi, se deben considerar los siguientes puntos:

- Instalar la distribución Raspbian en la placa Raspberry Pi.
- Instalar el *switch* virtual Open vSwitch en Raspbian.
- Instalar y configurar los adaptadores USB-Ethernet y el direccionamiento IP del dispositivo.
- Configurar el *switch* virtual Open vSwitch con los parámetros de la SDN.

A continuación, se describen en detalle los puntos indicados anteriormente.

2.6.1.1 Instalación de la distribución Raspbian en la placa Raspberry Pi

El proceso detallado de la instalación de Raspbian en la Raspberry Pi se encuentra en el **Anexo B**.

2.6.1.2 Instalación del switch virtual Open vSwitch en Raspbian

Para instalar el software, el dispositivo debe disponer de una conexión a Internet y el primer paso es actualizar los repositorios de software de Raspbian, esto se lo efectúa ejecutando el Comando 2.8.

```
$ sudo apt-get update
```

Comando 2.8. Actualización de repositorios de software en Raspbian

2.6.1.2.1 Instalación de cabeceras de kernel para Raspberry Pi

Posteriormente para la instalación de Open vSwitch es necesario descargar las cabeceras de kernel desde los repositorios git de Raspberry Pi, para esto se debe descargar el *script* **`rpi-source`** desde Internet, ejecutando el Comando 2.9.

```
$ sudo wget https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source -O /usr/bin/rpi-source && sudo chmod +x /usr/bin/rpi-source && /usr/bin/rpi-source -q --tag-update
```

Comando 2.9. Descarga del script **`rpi-source`** desde Internet

Antes de ejecutar el script descargado se deben instalar las librerías de software ejecutando el Comando 2.10, el Comando 2.11 y el Comando 2.12, estas librerías se encuentran descritas en la Tabla 2.6.

Tabla 2.6. Librerías requisito para la instalación de cabeceras de kernel

Librería	Descripción
`libncurses5-dev`	Métodos independientes de terminal para actualizar caracteres en pantalla de manera óptima.
`bc`	Lenguaje de procesamiento numérico de precisión arbitraria. La sintaxis es similar a C, pero difiere en muchas áreas sustanciales. Soporta la ejecución interactiva de sentencias.
`gcc-4.9`	Conjunto de compiladores creados por el proyecto GNU. gcc es software libre. Estos compiladores se consideran estándar para los sistemas operativos derivados de UNIX, de código abierto y también de propietarios, como Mac OS X.
`g++-4.9`	Alias de GNU C++, un conjunto gratuito de compiladores de C++ que forma parte del gcc.

```
$ sudo apt-get install bc libncurses5-dev -y
```

Comando 2.10. Instalación de librerías “bc” y “libncurses5

```
$ sudo update-alternatives –install /usr/bin/gcc
/usr/bin/gcc-4.9 50
```

Comando 2.11. Instalación de la librería gcc

```
$ sudo update-alternatives –install /usr/bin/g++ g++
/usr/bin/g++-4.9 50
```

Comando 2.12. Instalación de la librería `g++`

A continuación, se procede a ejecutar el *script* **`rpi-source`** utilizando el Comando 2.13, el cual descargará el kernel de LINUX lo descomprimirá y lo compilará, obteniendo así las cabeceras de kernel de LINUX.

```
$ rpi-source
```

Comando 2.13. Ejecución del script **`rpi-source`**

2.6.1.2.2 Instalación de Open vSwitch sobre Raspberry Pi

Antes de instalar Open vSwitch se deben instalar las siguientes librerías y dependencias, ejecutando el Comando 2.14, estas librerías se encuentran descritas en la Tabla 2.7.

Tabla 2.7. Librerías requisito para la instalación de Open vSwitch

Librería	Descripción
`libssl`	Librería que brinda soporte a los protocolos TLS[54] y SSL.
`libssl-dev`	Librería para desarrolladores de aplicaciones con soporte a los protocolos TLS y SSL.
`autoconf`	Genera diversos ficheros que permiten construir un programa adaptado al entorno en que se va a ejecutar.
`automake`	Herramienta que genera automaticamente archivos`Makefile.in desde archivos llamados `Makefile.am'.
`build-essential`	Contiene una lista informativa de paquetes que se consideran esenciales para la construcción de paquetes Debian.
`fakeroot`	Permite crear un entorno en el que parece que se tiene permisos de root, pero no es así.
`module-asistant`	Herramienta diseñada para facilitar la instalación de módulos mediante una serie de menús que permite elegir los módulos a descargar e instalar.
`uuid-runtime`	Genera un UUID de 128 bits.

[54] TLS (*Transport Layer Security*): permite y garantiza el intercambio de datos en un entorno seguro y privado entre dos entes, el usuario y el servidor.

```
$ sudo apt-get install libssl libssl-dev autoconf automake
libtool build-essential fakeroot module-asistant uuid-
runtime -y
```

Comando 2.14. Instalación de librerías y dependencias para Open vSwitch

Luego se debe descargar el archivo del tipo **tar.gz** que contiene Open vSwitch en su versión 2.5.0, desde su sitio web oficial, esto se lo realiza utilizando el Comando 2.15. Una vez descargado el archivo se lo descomprime y se ejecuta el script para preparar la instalación de Open vSwitch, ejecutando el Comando 2.16 y el Comando 2.17.

```
$ sudo wget http://openvswitch.org/releases/openvswitch-
2.5.0.tar.gz
```

Comando 2.15. Descarga del archivo tar.gz que contiene Open vSwitch

```
$ tar -xvzf openvswitch-2.5.0.tar.gz
```

Comando 2.16. Descompresión del archivo openvswitch-2.5.0.tar.gz

```
$ ./openvswitch-2.5.0/boot.sh
```

Comando 2.17. Ejecución para preparar la instalación de Open vSwitch

Una vez terminado todo el proceso de preparación para la instalación, se debe ingresar al directorio donde se encuentra Open vSwitch y con el Comando 2.18, se buscan todas las dependencias de software faltantes para Raspbian, obteniendose una salida por pantalla similar a la que se muestra en la Figura 2.23.

```
$ dpkg-checkbuilddeps
```

Comando 2.18. Búsqueda de dependencias de software faltantes

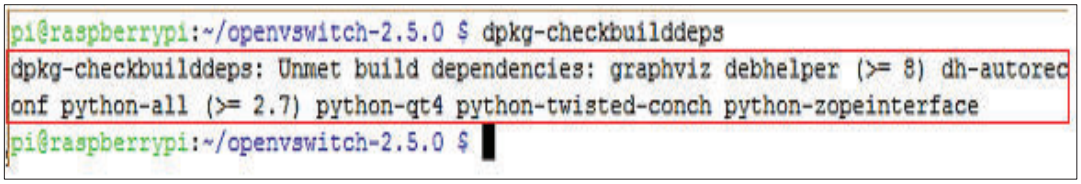

Figura 2.23. Obtención de dependencias de software faltantes

Luego se debe ejecutar el Comando 2.19 y Comando 2.20 para instalar las dependencias faltantes, las cuales se detallan en la Tabla 2.8.

```
$ sudo apt-get install graphviz debhelper dh-autoreconf
python-all python-qt4 -y
```

Comando 2.19. Instalación de dependencias faltantes de software

```
$ sudo apt-get install python-twisted-conch python-
zopeinterface -y
```

Comando 2.20. Instalación de otras dependencias de software faltantes

Tabla 2.8. Dependencias faltantes para la instalación de Open vSwitch

Librería	Descripción
graphviz	Software generador de grafos. Es open source y es independiente del lenguaje de programación que le llame.
debhelper	Colección de programas que se pueden utilizar en un archivo de reglas de debian para automatizar tareas comunes relacionadas con la creación de paquetes debian.
dh-autoreconf	Extensión para la librería debhelper, que permite auto configurar varias veces cuando se construye un programa.
python-all	Este paquete es un paquete de dependencia utilizado como dependencia de compilación para otros paquetes para evitar dependencias codificadas en tiempos de ejecución específicos de Python.
python-qt4	API para ell desarrollo de interfaces gráficas de usuario utilizando Qt4 y Python.
python-twisted-conch	Implementación cliente/servidor del protocolo SSH, utilizando un framework twisted.
python-zopeinterface	Entorno de desarrollo para la creación de sitios o aplicaciones web usando un servidor de aplicaciones web orientado al objeto, escrito en el lenguaje de programación Python.

Para la obtención de los archivos binarios de extensión tipo **.deb**, se debe utilizar el Comando 2.21, especificando que se ejecutará en 4 procesos paralelos y que no se realizarán comprobaciones de los archivos que se vayan obteniendo, ya que así se obtendrán los archivos binarios de forma más rapida. Después que culmine el proceso, se debe volver al directorio donde se encuentra la carpeta de instalación y empezar a instalar el software a partir de los archivos .**deb**. En primer lugar, se debe instalar el software base para Open vSwitch, a partir de la ejecución del Comando 2.22.

```
$ DEB_BUILD_OPTIONS='parallel=4 nocheck' fakeroot
debian/rules binary
```

Comando 2.21. Obtención de archivos binarios **deb**

```
$ sudo dpkg -i openvswitch-datapath-source_2.5.0-1_all.deb
```

Comando 2.22. Instalación de software Open vSwitch base

Posteriormente se debe construir el módulo de kernel de Open vSwitch para esto se debe ejecutar el Comando 2.23 .

```
$ sudo module-assistant build --kernel-dir ./linux
openvswitch-datapath
```

Comando 2.23. Construcción del módulo de kernel de Open vSwitch

Para finalizar, se debe instalar el módulo de kernel de Open vSwitch previamente creado con la utilización del Comando 2.24, además de instalar los componentes necesarios de Open vSwitch e instalar el *switch* userspace a partir de la ejecución del Comando 2.25 y el Comando 2.26, respectivamente.

```
$ sudo dpkg -i openvswitch-datapath-module-
4.1.19+_2.5.0-1_armhf.deb
```

Comando 2.24. Instalación del módulo de kernel Open vSwitch

```
$ sudo dpkg -i openvswitch-common_2.4.90-1_armhf.deb
```

Comando 2.25. Instalación de componentes de Open vSwitch

```
$ sudo dpkg -i openvswitch-switch_2.4.90-1_armhf.deb
```

Comando 2.26. Instalación del *switch* userspace Open vSwitch

Luego de utilizar los últimos comandos, Open vSwitch se iniciará y se configurará automáticamente, ahora se puede verificar si la instalación fue exitosa ejecutando el Comando 2.27, donde de ser el caso se observará la salida por pantalla que se puede apreciar en la Figura 2.24.

```
$ ovs-vswitchd --version
```

Comando 2.27. Verificación de instalación de Open vSwitch

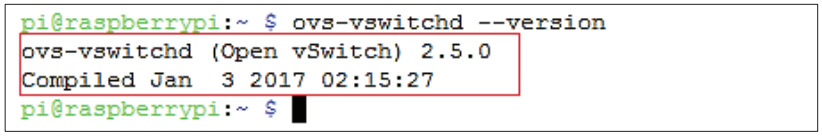

Figura 2.24. Open vSwitch instalado en Raspbian

2.6.1.3 Instalación y configuración de los adaptadores USB-Ethernet y el direccionamiento IP del dispositivo.

Para la instalación y configuración de los adaptadores USB-Ethernet, en primer lugar se debe configurar el direccionamiento IP según el criterio de diseño propuesto en la sección 2.3.2. La configuración se lo realiza mediante el archivo de configuración **dchpd.conf** localizado en el directorio **/etc** de Raspbian. Al archivo de configuración se le agrega el nombre de interfaz, su dirección IP estática y su

máscara de red, la puerta de enlace es opcional ya que no se la utilizará porque se trabajará en una red de área local sin conexión hacia Internet. Las líneas agregadas al archivo se pueden apreciar en la Figura 2.25.

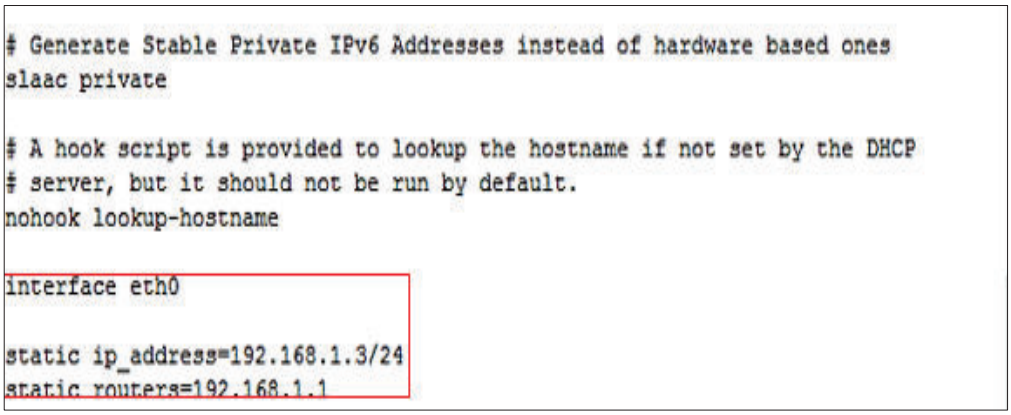

Figura 2.25. Archivo de configuración IP en Raspbian

Posteriormente para instalar y configurar las interfaces USB-Ethernet es necesario editar el archivo **dhcpd.conf** agregando el mismo número de interfaces a los puertos USB que se conectarán, en este caso serán un total de tres, por lo tanto, a las interfaces se les nombrará eth1, eth2 y eth3.

Todas las interfaces tendrán la dirección IP estática 0.0.0.0, lo cual indica que estas interfaces no necesitarán una dirección IP ya que serán puertos OpenFlow para la SDN implementada, las líneas agregadas al archivo se muestran en la Figura 2.26.

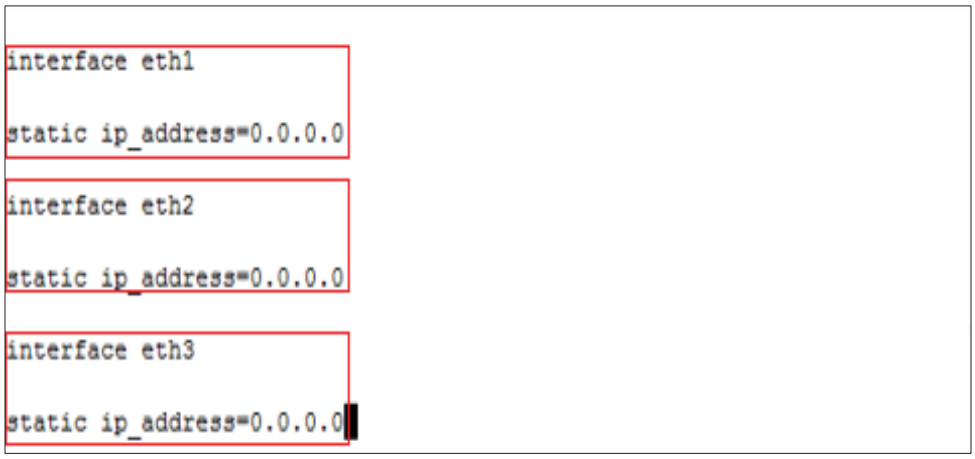

Figura 2.26. Archivo de configuración de interfaces USB-Ethernet en Raspbian

Finalmente se conectan las interfaces USB-Ethernet en los puertos USB de la placa y se reinicia el dispositivo para que las interfaces se detecten en el arranque, posteriormente, con el comando `ifconfig`, se verifica si la configuración fue exitosa, tal como se muestra en la Figura 2.27.

```
pi@raspberrypi:~ $ ifconfig | tail -n +10
eth1      Link encap:Ethernet  HWaddr 00:0e:c6:87:93:7e
          inet6 addr: fe80::b777:ace3:cba0:e6c2/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth2      Link encap:Ethernet  HWaddr 00:0e:c6:87:93:3a
          inet6 addr: fe80::2eb4:e931:2135:b798/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth3      Link encap:Ethernet  HWaddr 00:0e:c6:87:93:68
          inet6 addr: fe80::18aa:217a:973f:e489/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Figura 2.27. Interfaces USB-Ethernet configuradas en Raspbian

2.6.1.4 Configuración del switch virtual Open vSwitch en Raspbian

Para configurar el *switch* basado en software Open vSwitch, se deben utilizar una serie de comandos, y para ejecutarlos se ha considerado que la mejor opción para ejecutarlos es a través de un script bash de LINUX, el cual se muestra dividido en el: Código 2.5, Código 2.6, y Código 2.7. A continuación se describen las líneas de código más importantes para la configuración del *switch* OpenFlow implementado. El codigo completo del switch se encuentra en el **Anexo C**.

```
1  #Declaracion de variables
2  #IP del controlador OpenFlow al cual se conectara el switch
3  IP_CTL=192.168.1.2
4  #ID del data path del switch
5  DP_ID=0000000000000001
6  #Nombre de la interfaz bridge OVS que se creara
7  SW=br0
8  #Interfaces de red que se utilizaran como puertos OpenFlow
9  PUERTOS="eth1 eth2 eth3"
10 <<Comentario
11 Variable para utilizar el comando ovs-vsctl especifando la
12 IP del switch y el puerto TCP a que servira de escucha
13 Comentario
14 VSCTL="ovs-vsctl"
15 #Variable para utilizar la base de datos de configuracion de OVS
16 OVSDB=/etc/openvswitch/conf.db
```

Código 2.5. Primera sección del script bash del *switch* OpenFlow

Línea 3-5: Se declaran las variables que contienen la direcciones IP del controlador OpenFlow y el identificador del *data path* del *switch*.

Línea 7-9: Se declaran las variables que contienen el nombre del interfaz *bridge* Open vSwitch y las interfaces de red de la Raspberry Pi que serán utilizadas como puertos OpenFlow.

Línea 14-16: Se declara la variable que servirá para simplificar el uso del comando `ovs-vsctl`, además se declara la variable que representará la configuración de la base datos de Open vSwitch.

```
17 # Subrutina para esperar que los puertos esten listos
18 espera_puertos() {
19   puerto=$1
20   while ! `netstat -na | grep $puerto` ; do
21     echo -n .
22     sleep 1
23   done
24 }
25 #Detiene el proceso del servidor de base de datos ovsdb-server
26 pkill -9 ovsdb-server
27 #Limpiar las interfaces
28 ifconfig br0 down
29 ifconfig ovs-system down
30 <<Comentario
31 Detiene el proceso del switch ovs-vswitchd y eliminar archivos
32 temporales de la configuracion de la base de datos OVS
33 Comentario
34 pkill -9 ovs-vswitchd
35 rm /etc/openvswitch/conf.db.~lock~
36 #Remueve la base de datos OVSDB y luego vuelve a crearla(recargarla)
37 rm -f $OVSDB
38 ovsdb-tool create $OVSDB /usr/share/openvswitch/vswitch.ovsschema
39 # Iniciar el servidor de base de datos ovsdb-server
40 ovsdb-server --remote=punix:/var/run/openvswitch/db.sock &
41 # Iniciar el switch ovs-vswtichd
42 ovs-vswitchd  --pidfile=ovs-vswitchd.pid --overwrite-pidfile -- &
```

Código 2.6. Segunda sección del script bash del *switch* OpenFlow

Línea 18-24: Subrutina para esperar que las interfaces de red estén listas.

Línea 26-35: Se detiene el proceso del servidor de base de datos de Open vSwitch, además se detiene el proceso del *switch* incluyendo la eliminación de configuraciones temporales de la base de datos de Open vSwitch.

Línea 37-40: Se elimina la base de datos de Open vSwitch, se la vuelve a crear y se inicia el servidor de base de datos con la nueva base de datos.

Línea 42: Se inicia el proceso del *switch*.

```
43 # Agregar la interfaz bridge OVS y configurarlo para OpenFlow v. 1.3.0
44 $VSCTL add-br $SW
45 $VSCTL set bridge $SW protocols=OpenFlow13
46 #Ciclo For para ir añadiendo las interfaces de red a la interfaz bridge OVS
47 for i in $PUERTOS ; do
48   PUERTO=$i
49     ifconfig $PUERTO up
50   $VSCTL add-port $SW $PUERTO
51 done
52 #Establecer el identificador data path del switch ya que este es aleatorio por defecto
53 $VSCTL set bridge $SW other-config:datapath-id=$DP_ID
54 #Conectar el switch al controlador OpenFlow
55 $VSCTL set-controller $SW tcp:$IP_CTL:6633
```

Código 2.7. Tercera sección del script bash del *switch* OpenFlow

Línea 44-45: Se añade la interfaz *bridge* anteriormente creada y además se la configura para que utilice el protocolo OpenFlow v. 1.3.0.

Linea 47-51: Lazo *for* para ir añadiendo las interfaces de red de la Raspberry Pi como puertos OpenFlow del *switch*.

Línea 53-55: Establecer el identificador *data path* del *switch* ya que este es aleatorio por defecto, además de conectarse al controlador a partir de su dirección IP y puerto que en este caso es el puerto TCP 6633[55].

El funcionamiento del script bash se resume en el diagrama de flujo que se muestra en la Figura 2.28.

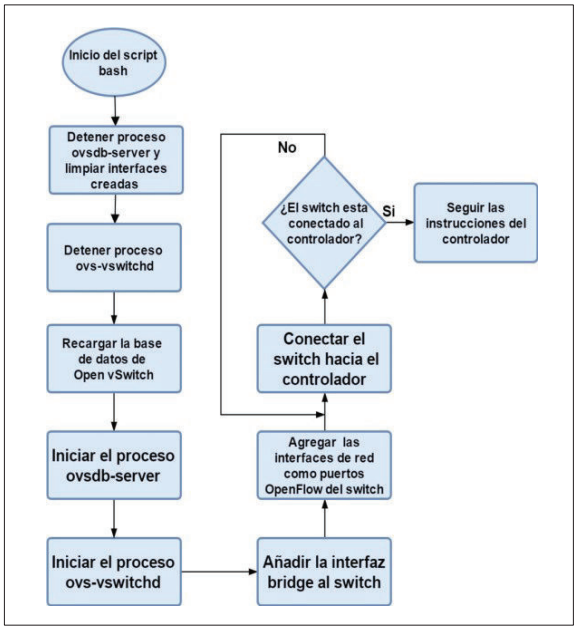

Figura 2.28. Diagrama de flujo del script bash **`inicio_sw.sh`**

[55] El puerto TCP 6633 es utilizado por defecto por el controlador Ryu para escuchar conexiones de *switches* OpenFlow [50].

Para que el script **inicio_sw.sh** se ejecute automáticamente en el arranque de la Raspberry Pi, se debe añadir la línea que se muestra en la Figura 2.29 en el archivo de configuración **/etc/rc.local**.

```
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
#Script de inicio del switch
/bin/sh /home/pi/inicio_sw.sh
exit 0
```

Figura 2.29. Configuración para el arranque automático del script **inicio_sw.sh**

2.6.2 IMPLEMENTACIÓN DE LOS ACCESS POINTS COMPATIBLES CON EL PROTOCOLO OPENFLOW SOBRE LAS PLACAS RASPBERRY PI

Para los *access points* se utilizó, al igual que para el *switch*, la plataforma de hardware libre, Raspberry Pi modelo B+ porque se determinó que las características de este modelo, descritas en la Tabla 1.2, son suficientes para que esta plataforma pueda actuar como *access point*. Cabe indicar que adicionalmente se requirió la utilización de adaptadores USB- Ethernet para ampliar la capacidad de puertos de red ya que el Raspberry Pi cuenta con un solo puerto de este tipo, además de un adaptador USB-WiFi, para añadir conectividad inalámbrica al dispositivo. En cuanto al software, se utilizó la distribución basada en LINUX OpenWrt[56], la cual es compatible con la Raspberry Pi, y a la cual se le instaló el *switch* basado en software Open vSwitch 2.5.0, con el fin de agregarle compatibilidad con el protocolo OpenFlow. En la Figura 2.30, se puede apreciar el diagrama de bloques del *access point* OpenFlow implementado.

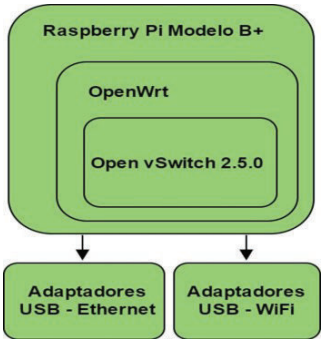

Figura 2.30. Diagrama de bloques de *access point OpenFlow* implementado

[56] OpenWrt: Firmware basado en una distribución de LINUX empotrada en dispositivos tales como routers personales.

Para la implementación de los *access points* OpenFlow, se deben seguir los siguientes pasos:

- Instalar la distribución OpenWrt en las placas Raspberry Pi.
- Instalar el *switch* virtual Open vSwitch en OpenWrt.
- Instalar y configurar las direcciones IP de los adaptadores USB-Ethernet y USB-WiFi.
- Configurar los parámetros inalámbricos del dispositivo.
- Configurar el *switch* virtual Open vSwitch con los parámetros de la SDN.

2.6.2.1 Instalación de la distribución OpenWrt en las placas Raspberry Pi

Para instalar la distribución OpenWrt en la placa Raspberry Pi, se debe disponer de una tarjeta microSD que sea compatible con el modelo B+ de la Raspberry Pi a utilizar. La capacidad del almacenamiento a utilizar es de 16GB, y la tarjeta es de clase 10, la cual es la clase recomendada para realizar instalaciones de distribuciones en dispositivos Raspberry Pi, por su alta velocidad de escritura y lectura de datos en comparación a otras clases. Por otro lado, para instalar la distribución OpenWrt, se debe disponer del archivo de tipo imagen que se descarga del sitio web oficial de OpenWrt [35]. Luego de descargar el archivo este se lo debe descomprimir y posteriormente se debe copiar el archivo imagen con un software para copia de imágenes en unidades de almacenamiento extraíbles, en este caso, al igual que para Raspbian se utilizó el software Win2DiskImager, disponible para su descarga en su sitio web oficial [36], ejecutado sobre una maquina con el sistema operativo Windows 10. Una vez ejecutado el software se debe seleccionar la imagen en el disco que en este caso tiene el nombre: **openwrt-brcm2708-bcm2708-rpi-ext4-sdcard.img**, como se observa en la Figura 2.31.

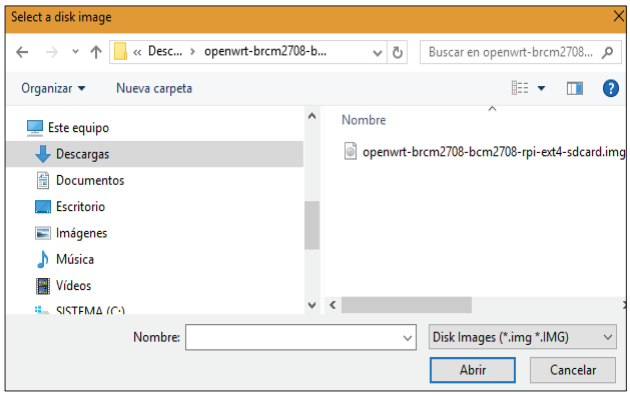

Figura 2.31. Dialogo de selección de imagen de OpenWrt en Win2DiskManager

Una vez seleccionada el archivo de imagen, se procede con la copia de la distribución en la tarjeta microSD, para eso se debe seleccionar la ubicación de la tarjeta y luego ejecutar la opción “Write”, ver Figura 2.32. Finalmente una vez se complete el proceso de escritura de la imagen en la tarjeta, se tendrá lista la distribución para utilizarla en la Raspberry Pi.

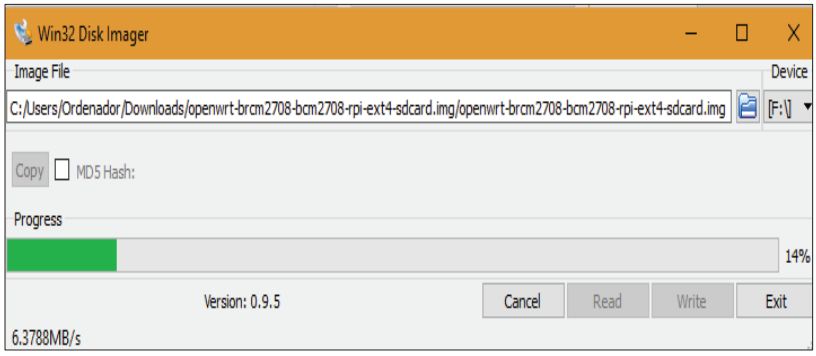

Figura 2.32. Proceso de escritura de imagen de OpenWrt en la tarjeta microSD

Por defecto los parámetros de red del dispositivo estarán configurados con la dirección IP 192.168.1.1, este parámetro será utilizado para el acceso remoto vía SSH a la placa.

Para ingresar a la placa remotamente se debe utilizar un cliente SSH, en este caso se utilizará el cliente para sistemas operativos Windows PuTTY, que se puede descargar su sitio web [37].

A continuación, en la Figura 2.33, se puede observar el acceso vía SSH mediante PuTTY a la placa Raspberry Pi, y para empezar la configuración del dispositivo se debe acceder con el usuario “root”, por defecto OpenWrt no tiene *password* en el usuario ”root” por lo que por seguridad es recomendable asignar uno.

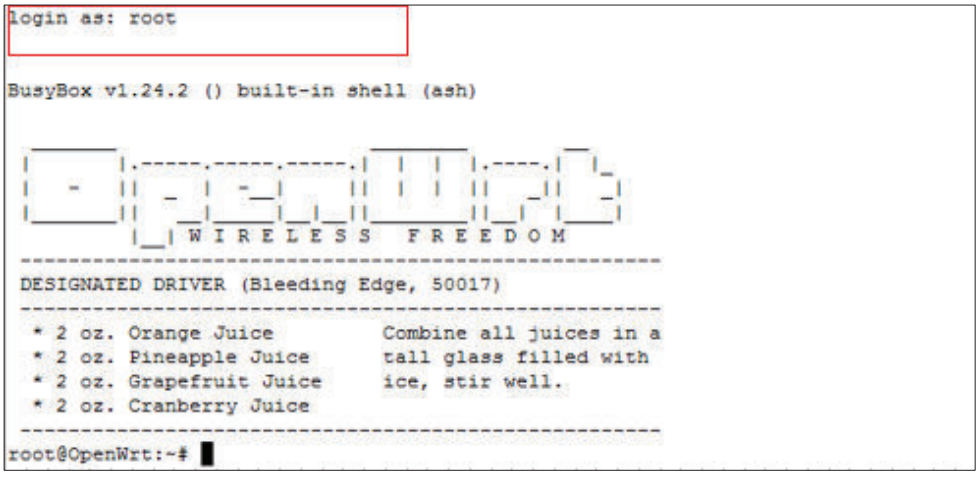

Figura 2.33. Ingreso vía SSH a OpenWrt utilizando PuTTY

Para asignar un *password* se debe utilizar el Comando 2.28. Una vez realizado el cambio de *password* se obtiene la salida por pantalla, que se observa en la Figura 2.34.

```
# passwd
```

Comando 2.28. Cambio de *password* en OpenWrt

```
root@OpenWrt:~# passwd
Changing password for root
New password:
Bad password: too weak
Retype password:
Password for root changed by root
root@OpenWrt:~#
```

Figura 2.34. Proceso de cambio de *password* finalizado

2.6.2.2 Instalación del switch virtual Open vSwitch en OpenWrt

Previo a proceder con la instalación de los paquetes de software en OpenWrt, se debe realizar unos pasos de configuración previos ya que se utilizó una versión *trunk*[57], la cual contiene la versión 2.5.0 de Open vSwitch que se requiere.

En esta versión de OpenWrt se encuentra desactualizado el archivo de configuración de los enlaces para realizar instalaciones de paquetes de software, el cual se encuentra localizado en la ruta: **/etc/opkg/distfeeds.conf**. De no actualizarse el archivo no se permitirá la instalación de ningún software, para solucionar este inconveniente se debe modificar el archivo, afin de que luzca como se muestra en la Figura 2.35.

```
root@OpenWrt:~# vim /etc/opkg/distfeeds.conf
src/gz designated_driver_base http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/base
src/gz designated_driver_kernel http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/kernel
src/gz designated_driver_luci http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/luci
src/gz designated_driver_management http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/managemen
src/gz designated_driver_packages http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/packages
src/gz designated_driver_routing http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/routing
src/gz designated_driver_telephony http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/telephony
src/gz designated_driver_targets http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/targets
```

Figura 2.35. Archivo de configuración modificado para instalación de paquetes

Adicionalmente para instalar el software, el dispositivo debe disponer de una conexión a Internet. Posterior al paso revisado, para actualizar el repositorio de

[57] Trunk: Ramal en desarrollo que puede contener código fuente experimental que está siendo modificado activamente. La versión en desarrollo de OpenWrt es compatible con hardware adicional.

paquetes de software de OpenWrt, se debe utilizar el Comando 2.29. Una vez actualizados los repositorios de software se visualizará una salida por pantalla similar a la que se puede apreciar en la Figura 2.36.

```
# opkg update
```

Comando 2.29. Actualización de repositorio de paquetes en OpenWrt

```
root:OpenWrt:# opkg update
Downloading https://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/base/
Packages.gz
Signature check passed.
Update list of available packages in /var/opkg-lists/designated_driver_base.
Downloading https://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/luci/
Packages.gz
Signature check passed.
Update list of available packages in /var/opkg-lists/designated_driver_luci.
Downloading https://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/kernel/
Packages.gz
Signature check passed.
Update list of available packages in /var/opkg-lists/designated_driver_kernel.
Downloading https://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/packages/routing/
Packages.gz
Signature check passed.
Update list of available packages in /var/opkg-lists/designated_driver_routing.
```

Figura 2.36. Proceso de actualización de repositorio de paquetes en OpenWrt

Una vez actualizados los repositorios de paquetes de software en OpenWrt, para instalar Open vSwitch, se debe ejecutar el Comando 2.30.

```
# opkg install openvswitch
```

Comando 2.30. Instalación de Open vSwitch en OpenWrt

Para realizar la verificación de la instalación de Open vSwitch se ejecuta el Comando 2.27 y se obtendrá una salida como la que se muestra en la Figura 2.37.

```
root@OpenWrt:~# ovs-vswitchd --version
ovs-vswitchd (Open vSwitch) 2.5.0
Compiled Dec 18 2016 15:02:09
root@OpenWrt:~#
```

Figura 2.37. Open vSwitch instalado en OpenWrt

2.6.2.3 Instalación y configuración IP de los adaptadores USB-Ethernet y USB-WiFi en OpenWrt

Para instalar y configurar los adaptadores USB-Ethernet y USB-WiFi, se deben instalar los *drivers* correspondientes, la instalación de los mismos se indica a continuación.

2.6.2.3.1 Instalación y configuración IP del adaptador USB-Ethernet en OpenWrt

En primer lugar, se debe buscar el *driver* específico del adaptador USB-Ethernet, el cual corresponde a un dispositivo ASIX-AX88178 de la familia de *drivers* USB-ASIX para OpenWrt, para instalarlo se utiliza el Comando 2.31.

```
# opkg install kmod-usb-net-asix
```

Comando 2.31. Instalación del *driver* del adaptador USB-Ethernet en OpenWrt

Posteriormente se debe editar el archivo de configuración de las interfaces en OpenWrt, el archivo de configuración lleva por nombre **network** y se localiza en la ruta **/etc/config/**. Se debe configurar el nombre de la interfaz y establecer el tipo de direccionamiento IP como estático, esta configuración es necesaria ya que la interfaz será utilizada como puerto OpenFlow para la SDN implementada, las líneas que se deben agregar se muestran en la Figura 2.38.

```
root@OpenWrt:~# vim /etc/config/network
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fdfb:3406:cb11::/48'

config interface 'lan'
        option type 'bridge'
        option ifname 'eth0'
        option proto 'dhcp'

config interface 'eth1'
        option ifname 'eth1'
        option proto 'static'
```

Figura 2.38. Configuración de la interfaz USB-Ethernet en OpenWrt

Finalmente se conecta la interfaz USB-Ethernet en uno de los puertos USB de la placa y se reinicia el dispositivo para que las interfaces se detecten en el arranque, después se verifica si la instalación fue exitosa ejecutando el Comando 2.32, y de ser el caso se obtendrá una salida por pantalla como se observa en la Figura 2.41.

```
# dmesg
```

Comando 2.32. Obtención de información de los dispositivos conectados

```
root@OpenWrt:~# dmesg
[   16.292063] IPv6: ADDRCONF(NETDEV_CHANGE): br-lan: link becomes ready
[   18.271611] br-lan: port 1(eth0) entered forwarding state
[   21.702839] random: nonblocking pool is initialized
[  329.191632] usb 1-1.2: new high-speed USB device number 4 using dwc_otg
[  329.315312] usb 1-1.2: New USB device found, idVendor=0b95, idProduct=1780
[  329.324045] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber
=3
[  329.333140] usb 1-1.2: Product: AX88178
[  329.338713] usb 1-1.2: Manufacturer: ASIX Elec. Corp.
[  329.345415] usb 1-1.2: SerialNumber: 008552
[  329.673019] asix 1-1.2:1.0 eth1: register 'asix' at usb-20980000.usb-1.2, ASI
X AX88178 USB 2.0 Ethernet, 00:0e:c6:87:93:68
root@OpenWrt:~#
```

Figura 2.39. Adaptador USB-Ethernet configurado en OpenWrt

2.6.2.3.2 Instalación y configuración IP del adaptador USB-WiFi

Para la instalación del adaptador USB-WiFi es necesario buscar el *driver* específico para el correcto funcionamiento de la interfaz, el *driver* para este caso, corresponde a un dispositivo Ralink RT5370 de la familia de drivers RT2800, el mismo que se instala utilizando el comando Comando 2.33.

```
# opkg install kmod-rt2800-usb
```

Comando 2.33. Instalación del *driver* del adaptador USB-WiFi en OpenWrt

Una vez que se conecta el adaptador USB-WiFi a la Raspberry Pi y se reinicia la placa para que se detecte el adaptador inalámbrico en el arranque, por defecto OpenWrt deshabilita las interfaces inalámbricas, para habilitarlas se debe ejecutar el Comando 2.34, el cual detecta la interfaz y crea un archivo de configuración nombrado **wireless** en la ruta **/etc/config/**.

```
# wifi detect >/etc/config/wireless
```

Comando 2.34. Creación del archivo de configuración inalámbrica en OpenWrt

Una vez creado el archivo **wireless**, es necesario editarlo para habilitar la interfaz inalámbrica, modificando la opción *disabled*, como se muestra en la Figura 2.40, las demás opciones se configurarán en próximas secciones.

```
root@OpenWrt:~# vim /etc/config/wireless
config wifi-device  radio0
        option type     mac80211
        option channel  11
        option hwmode   11g
        option path     'platform/soc/20980000.usb/usb1/1-1/1-1.2/1-1.2:1.0'
        option htmode   HT20
        option disabled '0'
```

Figura 2.40. Archivo de configuración de interfaz inalámbrica en OpenWrt

Adicionalmente se debe crear una interfaz inalámbrica, a partir de la edición del archivo **network** localizado en la ruta **/etc/config/**, especificando el nombre de la interfaz, y configurando su direccionamiento IP estático y su máscara de red, según el diseño propuesto en la sección 2.3.2. Las líneas agregadas al archivo se muestran en la Figura 2.41.

```
root@OpenWrt:~# vim /etc/config/network

config interface 'wifi'
        option proto 'static'
        option netmask '255.255.255.0'
        option ipaddr '192.168.100.1'
```

Figura 2.41. Creación interfaz inalámbrica en OpenWrt

Finalmente para activar la interfaz inalámbrica se ejecuta el Comando 2.35 y se reinicia la Raspberry Pi, para que la interfaz inalámbrica sea detectada en el arranque del dispositivo.

```
# wifi up
```

Comando 2.35. Habilitación de la interfaz inalámbrica en OpenWrt

2.6.2.3.3 Configuración IP de la Raspberry Pi con OpenWrt

Según el diseño propuesto en la sección 2.3.2.2, se debe configurar una dirección IP para que el *access point* se comunique con el dispositivo controlador, esto a partir de la edición del archivo **network** localizado en la ruta **/etc/config/**, se debe especificar el nombre de la interfaz, establecer el direccionamiento IP como estático, añadir la dirección IP y su máscara de red, esto se realiza añadiendo las líneas que se muestran en la Figura 2.42.

```
root@OpenWrt:~# vim /etc/config/network

config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config interface 'lan'
        option type 'bridge'
        option _orig_ifname 'eth0'
        option _orig_bridge 'true'
        option proto 'static'
        option ifname 'eth0'
        option netmask '255.255.255.0'
        option ipaddr '192.168.1.4'
```

Figura 2.42. Configuración IP en OpenWrt

2.6.2.4 Configuración de los parámetros inalámbricos de la red en OpenWrt

Para la configuración de los parámetros inalámbricos de la red, propuesta en la sección 2.3.1, se pueden utilizar dos formas: la primera es a partir de la edición de archivos de configuración de OpenWrt y la segunda es mediante la interfaz web gráfica de configuración de OpenWrt. En la presente sección se presenta las dos formas de configuración.

2.6.2.4.1 Configuración de los parámetros inalámbricos de la red en OpenWrt mediante archivos de configuración

Para configurar los parámetros inalámbricos de la red propuestos en la sección de diseño, se debe editar el archivo de configuración **wireless**, que se encuentra

ubicado en la ruta **/etc/config/**. En la Figura 2.43, se presenta el archivo **wireless** con los parámetros configurados de la red inalámbrica. Es importante señalar que se utilizó el esquema de seguridad WEP, debido a que el *switch* OpenFlow utilizando Open vSwitch presenta un problema de incompatibilidad con otros esquemas de seguridad como WPA o WPA2 [34].

```
root@OpenWrt:~# vim /etc/config/wireless
config wifi-device 'radio0'
        option type 'mac80211'
        #Canal inalambrico
        option channel '3'
        #Modo de transmision
        option hwmode '11g'
        option path 'platform/soc/20980000.usb/usb1/1-1/1-1.2/1-1.2:1.0'
        #Potencia de transmision
        option txpower '20'
        option country 'EC'

config wifi-iface
        option device 'radio0'
        option network 'wifi'
        #Modo de operacion
        option mode 'ap'
        #Nombre de red
        option ssid 'Red_AP_1'
        #Esquema de seguridad
        option encryption 'wep-open'
        #Numero de llave WEP
        option key '1'
        #Password
        option key1 's:openflow12345'
```

Figura 2.43. Archivo de configuración **wireless** de OpenWrt

En OpenWrt el servicio DHCP viene configurado por defecto y es recomendable deshabilitarlo, ya que de lo contrario enviará paquetes DHCP que no serán utilizados en el AP implementado. La configuración se encuentra en el archivo **dhcp**, el cual se encuentra localizado en la ruta **/etc/config.** Como se utilizará un direccionamiento estático en la red no será necesario utilizar un servidor DHCP por lo tanto, se debe deshabilitar el servicio modificando el archivo de configuración **dhcp** como se puede observar en la Figura 2.44.

```
root@OpenWrt:~# vim /etc/config/dhcp
config dhcp
        option interface 'wifi'
        option ignore '1'
```

Figura 2.44. Archivo de configuración **dhcp**

Para completar la configuración de los parámetros de red inalámbricos, es necesario configurar el firewall interno de OpenWrt, esto se lo realiza mediante la edición del archivo de configuración **`firewall`**, el cual se encuentra en la ruta

/etc/config/, se debe asignar una zona de firewall[58], la cual contendrá las reglas para aceptar o rechazar tráfico de datos que ingresan o salen de la red `wifi`, la configuración se puede observar en la Figura 2.45.

```
root@OpenWrt:~# cat /etc/config/firewall | tail -n14
config zone
        option name wifi
        list network wifi
        option input ACCEPT
        option output ACCEPT
        option forward REJECT
```

Figura 2.45. Archivo de configuración `firewall` en OpenWrt

Para que las configuraciones que se efectuaron en la placa surtan efecto, es necesario reiniciar el dispositivo.

2.6.2.4.2 Configuración de los parámetros inalámbricos de la red en OpenWrt mediante interfaz gráfica web

La versión de OpenWrt de la presente implementación, por defecto carece de la interfaz gráfica web LuCI[59], por lo tanto para instalarla se debe ejecutar el Comando 2.36.

```
# opkg install luci-ssl
```

Comando 2.36. Instalación de la interfaz web gráfica LuCI

Con la interfaz gráfica ya instalada, para acceder a la interfaz web se debe acceder al dispositivo vía navegador web, utilizando el mismo nombre de usuario y password, con los cuales se accedió vía SSH en la sección 2.6.2.1, como se muestra en la Figura 2.46.

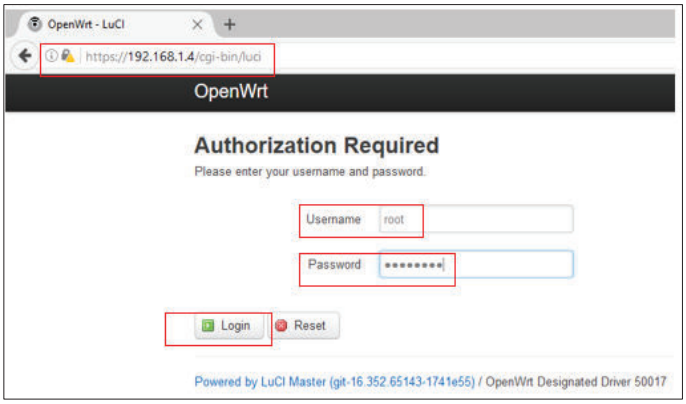

Figura 2.46. Acceso a la interfaz web LuCI

[58] Zona de firewall: Delimitación de un conjunto de redes donde se establecen reglas de reenvío de tráfico interno o externo a una red.
[59] LuCI: Interfaz de configuración web fácil y amigable para el usuario compatible con dispositivos OpenWrt.

La primera configuración a realizar mediante LuCI es la configuración de la interfaz inalámbrica, donde se deben modificar los siguientes parámetros: modo de transmisión, potencia de transmisión, canal inalámbrico, modo de operación, número de clave WEP, nombre de red, esquema de seguridad y contraseña. La configuración mediante LuCI se puede observar en la Figura 2.47, Figura 2.48 y Figura 2.49.

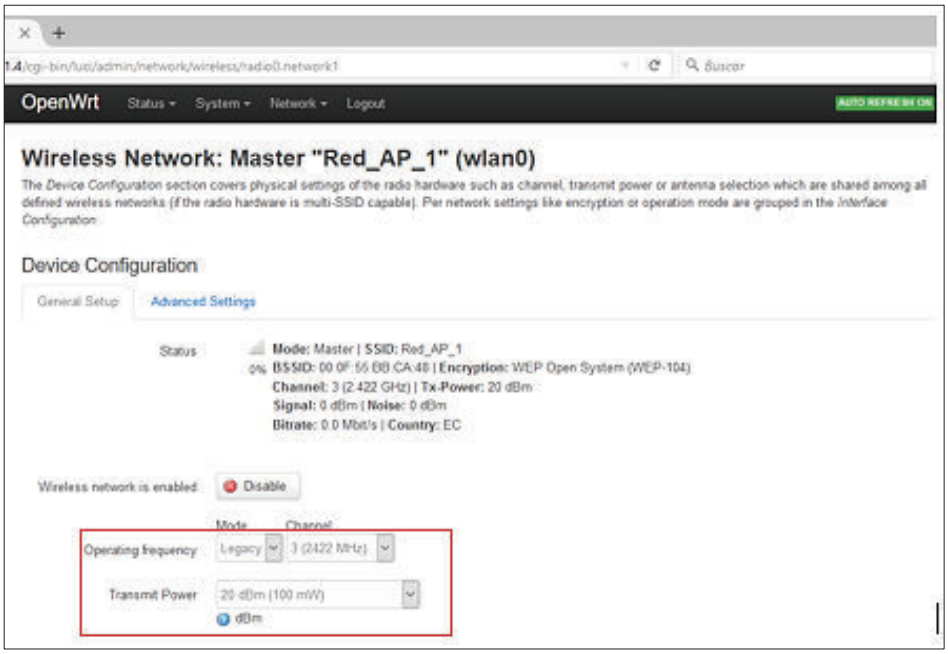

Figura 2.47. Configuración general del dispositivo vía LuCI

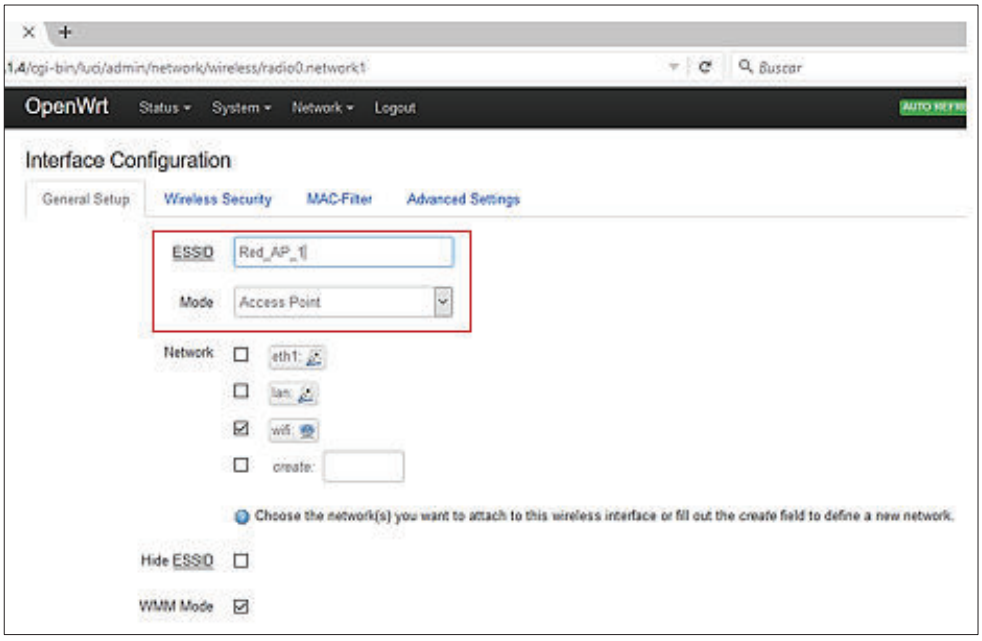

Figura 2.48. Configuración general de la interfaz inalámbrica vía LuCI

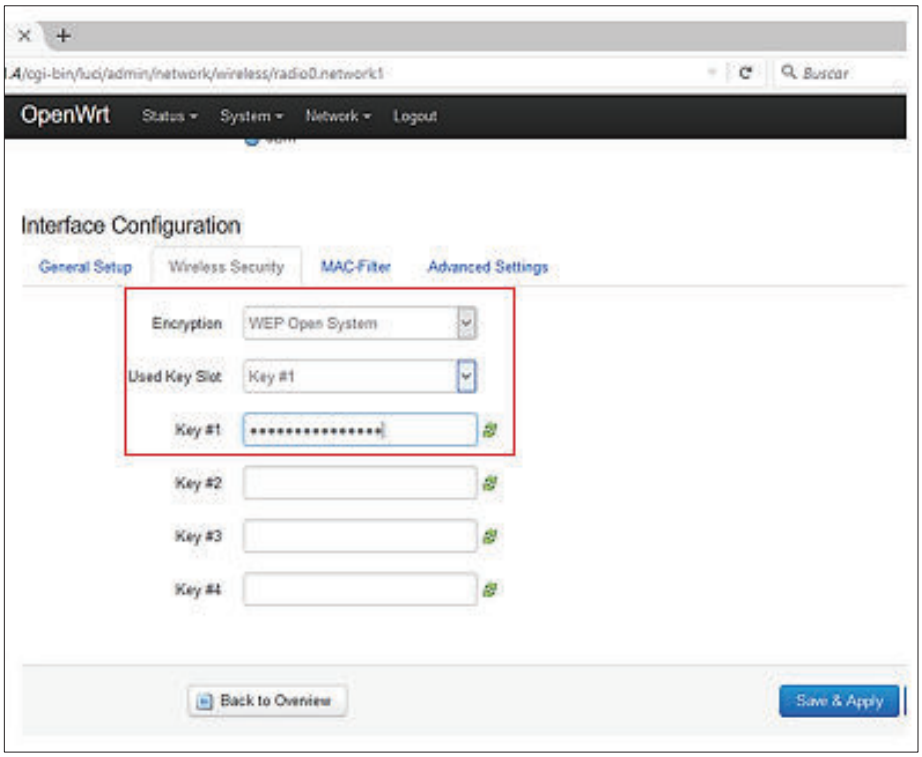

Figura 2.49. Configuración de seguridad inalámbrica vía LuCI

Adicionalmente es necesario configurar el servidor DHCP del dispositivo, y tomando en cuenta que se trabajará con un direccionamiento IP estático en la red, es mejor desactivar el servidor DHCP en la Raspberry Pi, la configuración se puede observar en la Figura 2.50.

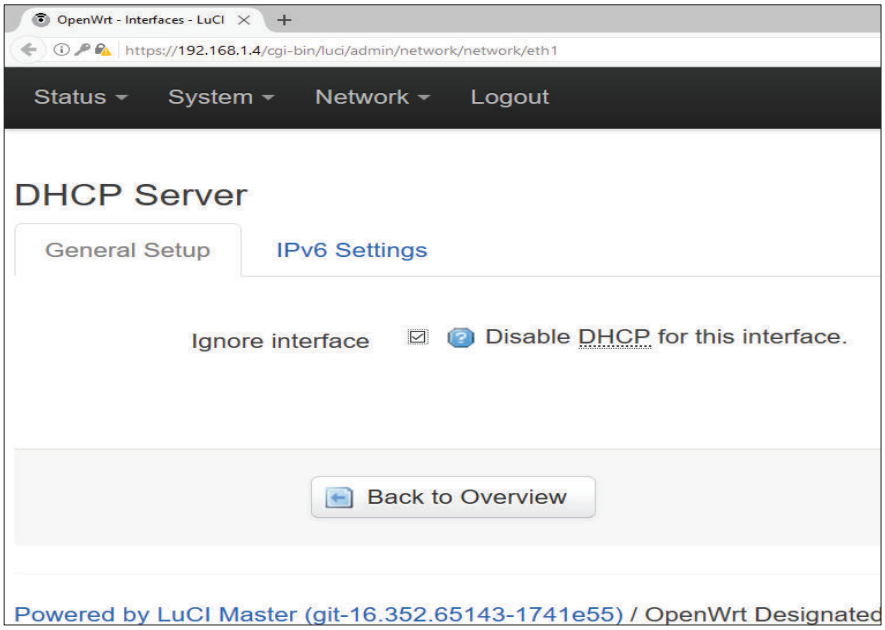

Figura 2.50. Configuración del servidor DHCP vía LuCI

Se debe configurar el firewall interno del dispositivo, mediante la asignación de las reglas para el envío de tráfico que ingresa y sale de la Raspberry Pi, tal como se muestra en la Figura 2.51.

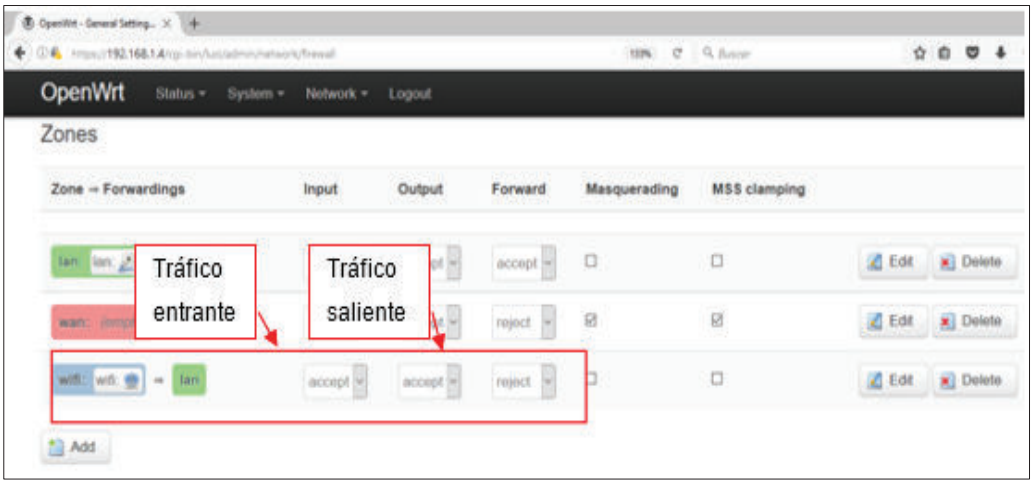

Figura 2.51. Configuración del firewall vía LuCI

Para que los cambios que se realizaron en todo el proceso de configuración a través de la interfaz web LuCI surtan efecto se debe guardar y aplicar (*Save and Apply*).

2.6.2.5 Configuración del switch virtual Open vSwitch en OpenWrt

Para configurar el *switch* basado en software Open vSwitch, se deben utilizar una serie de comandos, los cuales se encuentran dentro de un script bash de LINUX, el cual se encuentra dividido en los siguientes segmentos: Código 2.8, Código 2.9, y Código 2.10. Las líneas de código más importantes para la configuración del AP OpenFlow implementado se presentan a continuación y el código completo se encuentra en el **Anexo D**.

```
5  #Declaracion de variables
6  #IP del AP que servira para las conexiones con la OVSDB
7  IP_SW=192.168.1.4
8  #IP del controlador OpenFlow al cual se conectara el AP
9  IP_CTL=192.168.1.2
10 #ID del data path del switch
11 DP_ID=0000000000000002
12 #Nombre de la interfaz bridge OVS que se creara
13 SW=br0
14 #Interfaces de red que se utilizaran como puertos OpenFlow
15 PUERTOS="wlan0 eth1"
16 <<Comentario
17 Variable para utilizar el comando ovs-vsctl especifando la
18 IP del switch y el puerto TCP a que servira de escucha
19 Comentario
20 VSCTL="ovs-vsctl --db=tcp:$IP_SW:9000"
21 #Variable para utilizar la base de datos de configuracion de OVS
22 OVSDB=/tmp/ovs-vswitchd.conf.db
```

Código 2.8. Primera sección del script bash del AP OpenFlow

Línea 7-11: Se declaran las variables que contienen las direcciones IP del AP, del controlador OpenFlow y el identificador del *data path* del *switch*.

Línea 13-15: Se declaran las variables que contienen el nombre de la interfaz *bridge* Open vSwitch y de las interfaces de red de la Raspberry Pi que serán utilizadas como puertos OpenFlow.

Línea 20-22: Se declara la variable que se utilizará para simplificar el uso del comando `ovs-vsctl`, además se declara la variable que representará la configuración de la base datos de Open vSwitch.

```
23 # Subrutina para esperar que los puertos esten listos
24 espera_puertos() {
25   puerto=$1
26   while ! `netstat -na | grep $puerto` ; do
27     echo -n .
28     sleep 1
29   done
30 }
31 #Detiene el proceso del servidor de base de datos ovsdb-server
32 /usr/bin/killall ovsdb-server
33 <<Comentario
34 Detiene el proceso del switch ovs-vswitchd y eliminar archivos
35 temporales de la configuracion de la base de datos OVS
36 Comentario
37 /usr/bin/killall ovs-vswitchd
38 rm /tmp/.ovs-vswitchd.conf.db.~lock~
39 #Remueve la base de datos OVSDB y luego vuelve a crearla(recargarla)
40 rm -f $OVSDB
41 ovsdb-tool create $OVSDB /usr/share/openvswitch/vswitch.ovsschema
42 # Iniciar el servidor de base de datos ovsdb-server
43 ovsdb-server $OVSDB --remote=ptcp:9000:$IP_SW &
44 #Esperar por el puerto escucha 9000 del ovsdb-server por 5 segundos
45 sleep 5
46 # Iniciar el switch ovs-vswtichd
47 ovs-vswitchd tcp:$IP_SW:9000 --pidfile=ovs-vswitchd.pid --overwrite-pidfile -- &
```

Código 2.9. Segunda sección del script bash del AP OpenFlow

Línea 24-30: Subrutina para esperar que las interfaces de red estén listas.

Línea 32-38: Se detiene el proceso del servidor de base de datos de Open vSwitch, además se detiene el proceso del *switch* incluyendo la eliminación de configuraciones temporales de la base de datos de Open vSwitch.

Línea 40-43: Se elimina la base de datos de Open vSwitch, se la vuelve a crear y se inicia el servidor de base de datos con la nueva base de datos y especificando la IP del AP y el puerto TCP de escucha número 9000[60].

[60] El puerto TCP de escucha de `ovsdb-server` debe ser uno que no esté ocupado por otro servicio, en este caso se escogió el puerto TCP número 9000.

Línea 47: Se inicia el proceso del *switch* configurado con la dirección IP del AP y el puerto TCP de escucha número 9000.

```
49 $VSCTL add-br $SW
50 $VSCTL set bridge $SW protocols=OpenFlow13
51 #Ciclo For para ir añadiendo las interfaces de red a la interfaz bridge OVS
52 for i in $PUERTOS ; do
53    PUERTO=$i
54       ifconfig $PUERTO up
55    $VSCTL add-port $SW $PUERTO
56 done
57 #Establecer el identificador data path del switch ya que este es aleatorio por defecto
58 $VSCTL set bridge $SW other-config:datapath-id=$DP_ID
59 #Conectar el switch al controlador OpenFlow
60 $VSCTL set-controller $SW tcp:$IP_CTL:6633
```

Código 2.10. Tercera sección del script bash del AP OpenFlow

Línea 49-50: Se añade la interfaz *bridge* anteriormente creada y además se la configura para que utilice el protocolo OpenFlow v. 1.3.0.

Linea 52-56: Lazo for para ir añadiendo las interfaces de red de la Raspberry Pi como puertos OpenFlow del AP.

Línea 58-60: Establecer el identificador *data path* del *switch* ya que este es aleatorio por defecto, además de conectarse al controlador a partir de su dirección IP y puerto que este caso es el puerto TCP 6633 [15].

El funcionamiento del script bash se resume en el diagrama de flujo que se muestra en la Figura 2.52.

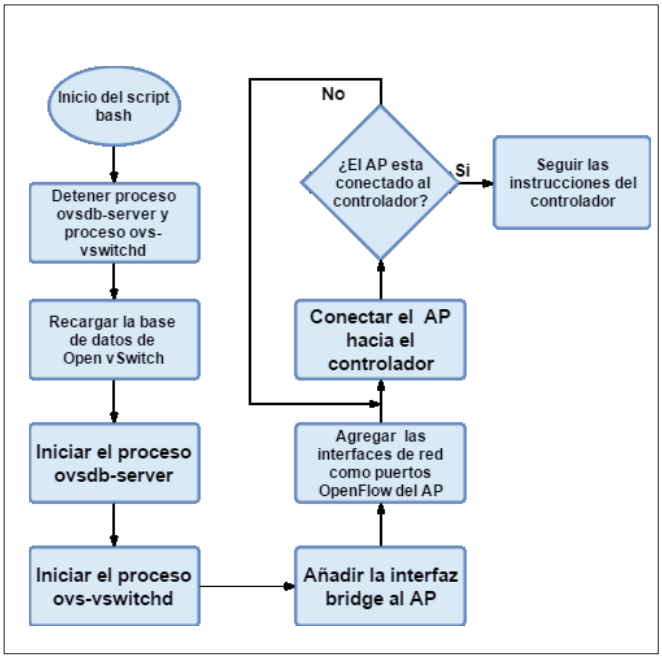

Figura 2.52. Diagrama de flujo del script bash **`inicio_ap_1.sh`**

Para que el script **inicio_ap_1.sh** se ejecute automáticamente en el arranque de la Raspberry Pi, se debe añadir el script en el siguiente directorio **/etc/init.d**, y se debe ejecutar el Comando 2.37 para habilitar el script en el arranque.

```
# /etc/init.d/inicio_ap_1.sh enable
```

Comando 2.37. Habilitación del script **inicio_ap_1.sh** en el arranque

Una vez realizada la implementación del primer AP, para la implementar el segundo AP es necesario seguir los mismos pasos del primer AP considerando los criterios de diseño propuestos para el segundo AP, en la sección 2.3.

2.6.3 IMPLEMENTACIÓN DEL CONTROLADOR OPENFLOW SOBRE LA PLACA RASPBERRY PI

A diferencia del *switch* y el AP, y en vista de que el dispositivo controlador de una SDN debe poseer características de procesamiento computacionales moderadamente altas, se determinó que se deberá utilizar la plataforma de hardware libre Raspberry Pi 3 modelo B, ya que, según la información descrita en la Tabla 1.2, posee las mejores características de rendimiento de todos los modelos hasta la fecha de las placas Raspberry Pi.

La distribución de LINUX que se utilizó es Raspbian OS, a la cual se le instaló el software controlador Ryu, así el dispositivo podrá comunicarse mediante el protocolo OpenFlow con los demás dispositivos (*switch, access points*) y además permitirá la ejecución de una aplicación de *firewall*.

En la Figura 2.53, se puede observar el diagrama de bloques del dispositivo controlador OpenFlow implementado.

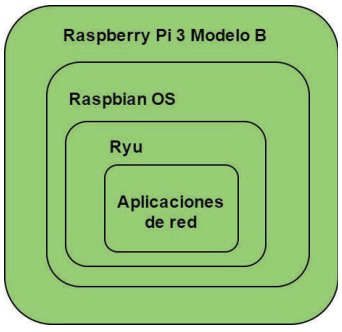

Figura 2.53. Diagrama de bloques del dispositivo controlador OpenFlow implementado

Los pasos a seguir para la implementación del controlador OpenFlow son los siguientes:

- Instalar el software controlador Ryu sobre Raspbian
- Configurar el direccionamiento IP del dispositivo.

2.6.3.1 Instalación del software controlador Ryu sobre Raspbian

En primer lugar, se deben actualizar los repositorios de paquetes de Raspbian como se indicó en la sección 2.6.1.2, luego como prerrequisito para instalar el controlador basado en software Ryu se debe instalar el software instalador de paquetes de Python conocido como `pip`[61] y además la librería `python-dev`[62], para instalar estos paquetes de software se debe ejecutar el Comando 2.38, como se muestra a continuación.

```
$ sudo apt-get install python-pip python-dev -y
```

Comando 2.38. Instalación de PIP y librería python-dev

Para instalar el controlador Ryu incluyendo sus librerías y para su correcto funcionamiento, se debe ejecutar el Comando 2.39, utilizando el instalador de paquetes de Python `pip`, tal como se puede observar continuación.

```
$ sudo pip install ryu greenlet repoze.lru stevedore
```

Comando 2.39. Instalación del controlador RYU por medio de pip

El paquete `six` [63] de Python normalmente suele estar desactualizado, esta es una dependencia de software del controlador Ryu, por lo tanto para evitar conflictos en el funcionamiento de Ryu es recomendable actualizar este paquete ejecutando el Comando 2.40.

```
$ sudo python pip –upgrade six
```

Comando 2.40. Actualización del paquete six de Python

Para comprobar que el controlador Ryu está instalado correctamente, se debe ejecutar el Comando 2.41, si la salida por pantalla es similar a la mostrada en la Figura 2.54 significará que la instalación fue correcta.

```
$ ryu-manager [nombre de la aplicación de Ryu]
```

Comando 2.41. Comando para inicializar el controlador Ryu

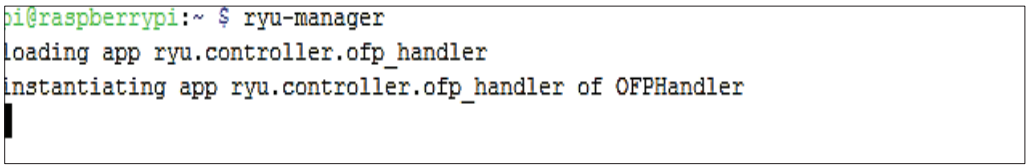

Figura 2.54. Ejecución de comprobación de Ryu

[61] pip: Sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.
[62] python-dev: Conjunto de archivos para generar cabeceras y librería estándar.
[63] python-six: Librería utilizada para facilitar la compatibilidad entre Python 2 y Python 3.

2.6.3.2 Configuración del direccionamiento IP del dispositivo controlador

Para configurar el direccionamiento IP según el criterio de diseño propuesto, se sigue un proceso similar al observado en la sección 2.6.1.3. La configuración se lo realiza mediante el archivo de configuración **dchpd.conf** localizado en el directorio **/etc** de Raspbian. Al archivo de configuración se le agrega el nombre de interfaz, su dirección IP estática y su máscara de red, la puerta de enlace es opcional ya que no se la utilizará porque se trabajará en una red de área local sin conexión a Internet. Las líneas agregadas al archivo se pueden apreciar en la Figura 2.55.

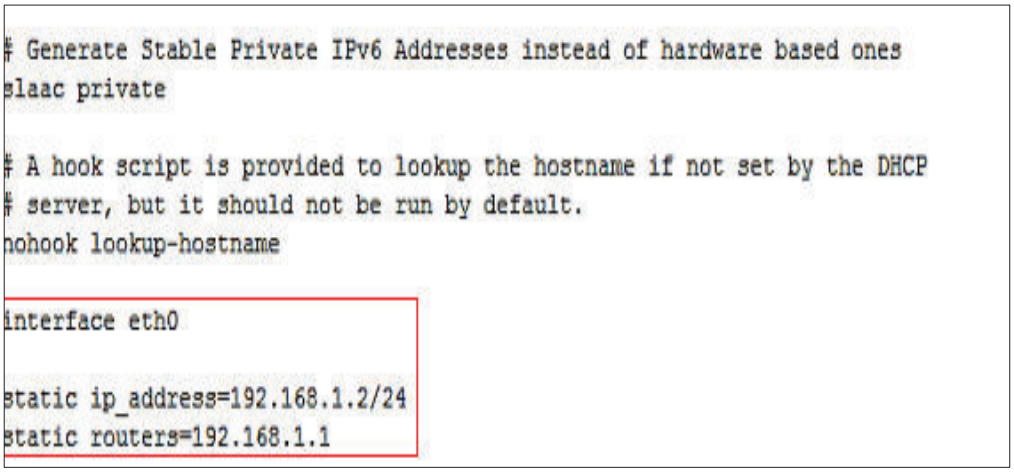

Figura 2.55. Configuración IP en el dispositivo controlador

2.6.4 IMPLEMENTACIÓN DE LA APLICACIÓN DE FIREWALL

El framework de Ryu provee de aplicaciones de ejemplo para que desarrolladores de software o estudiantes del campo de las SDN puedan aprender y comprender de una mejor manera el cómo desarrollar aplicaciones para SDN. Por lo tanto, se ha escogido a la aplicación **`rest_firewall.py`** para a partir de la misma desarrollar la aplicación de firewall del presente proyecto.

La funcionalidad de la aplicación de firewall será la de filtrar el tráfico de datos entre las estaciones (fijas e inalámbricas) de la SDN, el filtrado se realizará en función de al menos los siguientes parámetros: dirección IP fuente y destino, tipo de protocolo (ICMP, TCP o UDP), número de puerto y si él envió de tráfico se permite o se deniega.

La inserción y remoción de reglas se las realizará ejecutando comandos HTTP (GET, POST, etc.) utilizando la REST API que ofrece el controlador Ryu.

La aplicación se ha desarrollado a partir de las librerías que ofrece Ryu escritas en Python, utilizando un total de seis clases, las cuales serán descritas a continuación.

2.6.4.1 Diagrama de clases de la aplicación de firewall

A continuación, en la Figura 2.56 se presenta el diagrama de clases de la aplicación de *firewall* el cual está conformado por seis clases escritas en un único archivo con el nombre **`firewall.py`**, el código completo se encuentra en el **Anexo E**.

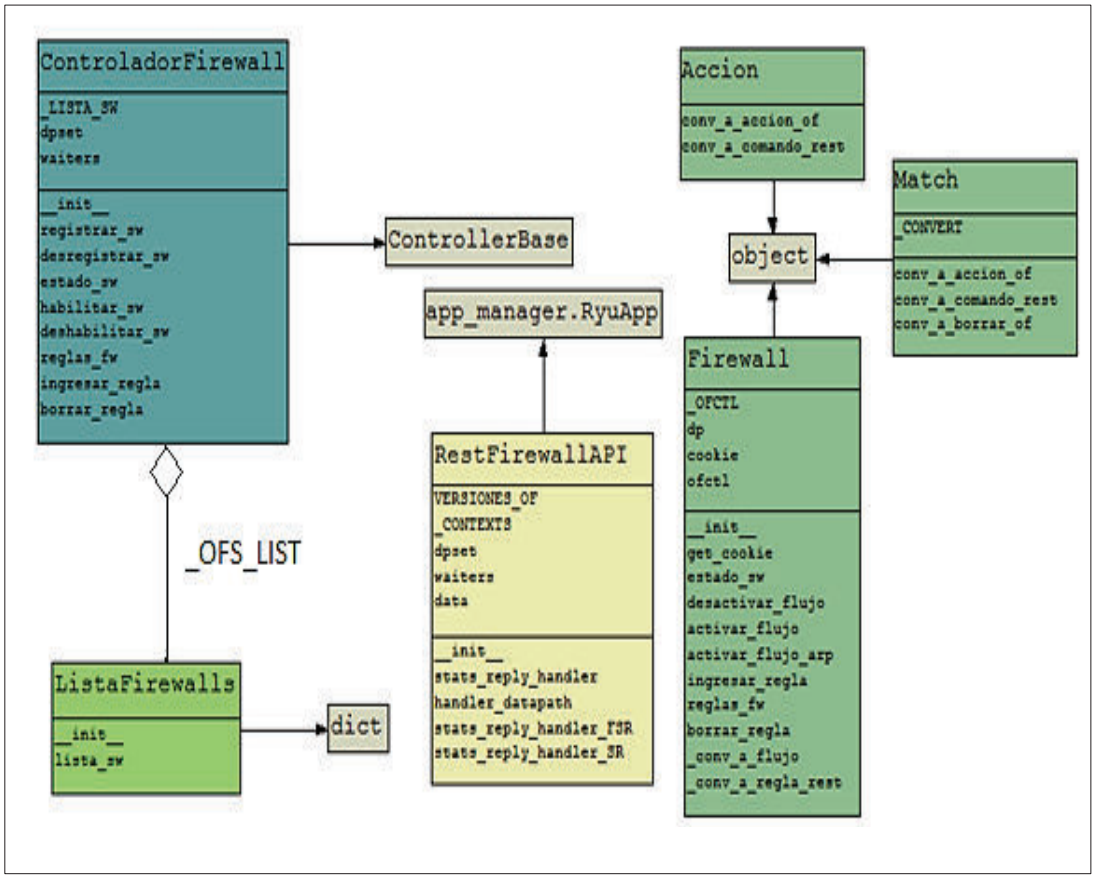

Figura 2.56. Diagrama de clases de la aplicación de firewall

2.6.4.2 Clase de construcción de la API REST del Firewall

La clase `RestFirewallAPI` será la encargada de la construcción de la API REST de la aplicación de *firewall*, es decir definirá una relación entre las peticiones HTTP y la ejecución de instrucciones programadas sobre el controlador OpenFlow. Además es la primera clase que se instancia en la aplicación y es utilizada para inicializar el resto de clases de la aplicación.

A continuación, se presentan los segmentos del código fuente más importantes de la clase.

En el Código 2.11, se puede observar la declaración de la clase `RestFirewallAPI`, así como la declaración de variables que se definirán en el método constructor de la clase.

```
37 """Clase para inicializar las demas clases, es la primera clase
38 que se instancia en la aplicacion en RYU"""
39 class RestFirewallAPI(app_manager.RyuApp):
40     #Lista que contiene las version del protocolo OpenFlow v.1.3)
41     VERSIONES_OF = [ofproto_v1_3.OFP_VERSION]
42     """Se define los contextos para poder utilizar objetos de la clase DPSet y
43     de la clase WSGIApplication"""
44     _CONTEXTS = {'dpset': dpset.DPSet,
45                  'wsgi': WSGIApplication}
46     """Se definen en el constructor,  los contextos "wsgi" y "dpset" a utilizarse
47     en el resto de la aplicacion"""
48     def __init__(self, *args, **kwargs):
49         super(RestFirewallAPI, self).__init__(*args, **kwargs)
50         self.dpset = kwargs['dpset']
51         wsgi = kwargs['wsgi']
52         """Se crea dos diccionarios para y se la da valores a sus respectivas llaves 'dpset'
53         y 'waiters' en funcion del contexto dpset"""
54         self.waiters = {}
55         self.data = {}
56         self.data['dpset'] = self.dpset
57         self.data['waiters'] = self.waiters
58         """El constructor obtiene la instancia de la aplicacion WSGI para registrarla en la
59         clase ControladorFirewall en funcion del diccionario 'data' """
60         wsgi.registory['ControladorFirewall'] = self.data
61         #Definicion de la ruta donde se empezara a recibir las peticiones HTTP
62         ruta = '/firewall'
63         """Creaccion de un diccionario que contendra los sw en funcion de su id de 16 caracteres
64         alfanumericos"""
65         id_dp = {'switchid': dpid_lib.DPID_PATTERN + r'|TODOS'}
```

Código 2.11. Definición de la clase `RestFirewallAPI`

A continuación, se realiza una explicación breve de las líneas más importantes del Código 2.11.

Línea 41: Se establece que se utilizará el protocolo OpenFlow v.1.3.0 en la aplicación.

Línea 44-45: Se define los contextos para poder utilizar objetos de la clase `DPSet` y de la clase `WSGIApplication`.

Línea 48-51: Se definen los contextos **wsgi**[64] y **dpset**[65] en el constructor de la clase y permite que se utilicen en el resto de la aplicación.

Línea 54-57: Se crea dos diccionarios: data y waiters, y se la da valores a sus respectivas llaves dpset y waiters [66]en función del contexto dpset. El diccionario data almacena los datos de tipo HTTP y el diccionario waiters las peticiones HTTP.

Línea 60: El constructor obtiene la instancia de la aplicación WSGI para registrarla en la clase `ControladorFirewall` en función del diccionario 'data'.

Línea 62: Definición de la ruta donde se empezará a recibir las peticiones HTTP.

---

[64] wsgi: Módulo de Ryu que ofrece una interfaz simple y universal entre servidores web y aplicaciones web o frameworks

[65] dpset: Módulo de Ryu para el manejo de *switches* mediante el protocolo OpenFlow.

[66] waiters: llave de un diccionario que almacenara las peticiones que llegaran a los *switches*.

Línea 65: Creación de un diccionario que contendrá el identificador de los *switches* en función de su identificador *data path* de 16 caracteres alfanuméricos.

En el Código 2.12, se puede observar el mapeo o relación de las peticiones HTTP, entre las cuales se tiene las peticiones: PUT, GET, POST y DELETE, con las instrucciones programadas sobre el controlador OpenFlow.

```
81      #Creacion de la URI para deshabilitar el firewall sobre un switch a partir de su id
82      uri = ruta + '/module/disable/{switchid}'
83      """Establecimiento de una conexion entre las peticiones HTTP y la aplicacion con el fin
84      de deshabilitar el firewall sobre un switch a partir de su id"""
85      wsgi.mapper.connect('firewall', uri,
86                  controller=ControladorFirewall, action='deshabilitar_sw',
87                  conditions=dict(method=['PUT']),
88                  requirements=id_dp)
89      """Creacion de la URI para obtener, establecer o eliminar las reglas de envio de trafico de un
90      switch a partir de su id"""
91      uri = ruta + '/reglas/{switchid}'
92      """Establecimiento de una conexion entre las peticiones HTTP y la aplicacion con el fin
93      de obtener las reglas de envio de trafico de un switch a partir de su id"""
94      wsgi.mapper.connect('firewall', uri,
95                  controller=ControladorFirewall, action='reglas_fw',
96                  conditions=dict(method=['GET']),
97                  requirements=id_dp)
98      """Establecimiento de una conexion entre las peticiones HTTP y la aplicacion con el fin
99      de establecer las reglas de envio de trafico de un switch a partir de su id"""
100     wsgi.mapper.connect('firewall', uri,
101                 controller=ControladorFirewall, action='ingresar_regla',
102                 conditions=dict(method=['POST']),
103                 requirements=id_dp)
104     """Establecimiento de una conexion entre las peticiones HTTP y la aplicacion con el fin
105     de eliminar las reglas de envio de trafico de un switch a partir de su id"""
106     wsgi.mapper.connect('firewall', uri,
107                 controller=ControladorFirewall, action='borrar_regla',
108                 conditions=dict(method=['DELETE']),
109                 requirements=id_dp)
```

Código 2.12. Mapeo de peticiones HTTP con el controlador

A continuación, se realiza una explicación breve de las líneas más importantes del Código 2.12.

Línea 82-88: Creación de la URI[67] para deshabilitar el firewall sobre un *switch* a partir de su identificador *data path*, además se establece una conexión entre las peticiones HTTP y la aplicación.

Línea 91-97: Creación de la URI para obtener, establecer o eliminar las reglas de envío de trafico de un *switch* a partir de su identificador *data path*, además se establece una conexión entre las peticiones HTTP y la aplicación.

Línea 100-103: Se establece una conexión entre las peticiones HTTP y la aplicación con el fin de establecer las reglas de envío de tráfico de un *switch* a partir de su identificador *data path*.

[67] URI (*Uniform Resource Identifier*): Cadena de caracteres que identifica los recursos de una red de forma unívoca.

Línea 106-109: Se establece una conexión entre las peticiones HTTP y la aplicación con el fin de eliminar las reglas de envío de tráfico de un *switch* a partir de su identificador *data path*.

En el Código 2.13, se manejan los eventos de conexión o desconexión de un *switch* y se los delega a una instancia de la clase `ControladorFirewall`.

```
136    """Maneja los eventos de conexion o desconexion de un switch y los delega a una instancia
137    de la clase ControladorFirewall"""
138    @set_ev_cls(dpset.EventDP, dpset.DPSET_EV_DISPATCHER)
139    def handler_datapath(self, ev):
140        if ev.enter:
141            ControladorFirewall.registrar_sw(ev.dp)
142        else:
143            ControladorFirewall.desregistrar_sw(ev.dp)
```

Código 2.13. Manejo de eventos de conexión/desconexión de un *switch*

2.6.4.3 Clase que contiene los switches que se conecten a la aplicación de firewall

La clase `ListaFirewalls` representa un diccionario de *switches*, es decir las instancias de la clase `ryu.controller.controller.Datapath`, la clase contiene un único método llamado **`lista_sw`** el cual retorna una instancia de un *data path* a partir de su identificador representado como cadena de texto. La clase se puede apreciar en el Código 2.14.

```
154 #Clase que contendra un diccionario con los switches conectados a la aplicacion
155 class ListaFirewalls(dict):
156     def __init__(self):
157         super(ListaFirewalls, self).__init__()
158     """Metodo que retornara la instancia del datapath a partir de su identificador"""
159     def lista_sw(self, dp_id):
160         #Si el identificador de switch es nulo salta una execpsion de "switch no conectado"
161         if len(self) == 0:
162             raise ValueError('switch no conectado.')
163         dps = {}
164         #Si el identificador de switch es TODOS, las acciones se realizaran para todos los switches
165         if dp_id == 'TODOS':
166             dps = self
167         else:
168             try:
169                 dpid = dpid_lib.str_to_dpid(dp_id)
170             #Si el identificador de switch posee una sintaxis incorrecta salta una excepcion de "switch invalido"
171             except:
172                 raise ValueError('ID del switch invalido.')
173             #Si el switch no se encuentra conectado a la aplicacion salta una excpecion de "sw no conectado al firewall"
174             if dpid in self:
175                 dps = {dpid: self[dpid]}
176             else:
177                 msg = 'firewall sw no conectado. : switchID=%s' % dp_id
178                 raise ValueError(msg)
179         return dps
```

Código 2.14. Clase `ListaFirewalls`

2.6.4.4 Clase para el manejo de métodos en tiempo de ejecución

La clase `ControladorFirewall`, define los métodos invocados por la API REST en tiempo de ejecución, además define los métodos para el manejo de conexiones y desconexiones de los *switches*. Los métodos de esta clase se presentan en la Tabla 2.9. Tanto la conexión y desconexión de un *switch* crea o destruye instancias de la clase `Firewall`. La API REST se encargará de expresar las tablas de flujo del *switch* en forma de reglas de un firewall.

Tabla 2.9. Métodos de la clase `ControladorFirewall`

Métodos invocados por la API REST	
Método	**Descripción**
`estado_sw`	Conecta las acciones que se deben realizar con la API REST para obtener el estado de el(los) estado(s) del(os) *switch*(es) conectados al controlador.
`habilitar_sw`	Conecta las acciones que se deben realizar con la API REST para habilitar el firewall sobre un *switch* a partir de su id.
`deshabilitar_sw`	Conecta las acciones que se deben realizar con la API REST para obtener las reglas de envío de trafico de un *switch* a partir de su id.
`reglas_fw`	Conecta las acciones que se deben realizar con la API REST para obtener las reglas de envío de trafico de un *switch* a partir de su id.
`ingresar_regla`	Conecta las acciones que se deben realizar con la API REST para establecer las reglas de envío de trafico de un *switch* a partir de su id.
`borrar_regla`	Conecta las acciones que se deben realizar con la API REST para eliminar las reglas de envío de trafico de un *switch* a partir de su id.
Métodos para el manejo de conexiones y desconexiones de un *switch*	
Método	**Descripción**
`registrar_sw`	Método estático para registrar los *switches* a partir de los datapath.
`desregistrar_sw`	Método estático para eliminar los *switches* registrados a partir de los datapath.

En el Código 2.15, se puede observar la declaración de la clase `ControladorFirewall`, así como la declaración de variables que se definirán en el método constructor de la clase.

```
180 #Clase heredada de la clase ControllerBase
181 class ControladorFirewall(ControllerBase):
182     #Se crea una lista a partir de instaciar la clase ListaFirewalls
183     _LISTA_SW = ListaFirewalls()
184     """Se define el constructor de la clase y se declara los valores para los datos para la
185     aplicacion WSGI"""
186     def __init__(self, req, link, data, **config):
187         super(ControladorFirewall, self).__init__(req, link, data, **config)
188         self.dpset = data['dpset']
189         self.waiters = data['waiters']
```

Código 2.15. Definición de la clase `ControladorFirewall`

2.6.4.5 Clase de modificación de mensajes de estados OpenFlow

La clase `Firewall`, que contiene la funcionalidad relacionada a escribir flujos al *switch* es instanciada cuando existen llamadas por la Interfaz API REST para mantener así una abstracción del estado de las reglas del firewall.

En el Código 2.16, se puede observar la declaración de la clase `Firewall`, así como la declaración de variables que se definirán en el método constructor de la clase, además se establece la versión del protocolo con la que se trabajará, es decir OpenFlow v. 1.3.0. Finalmente se establece una excepción en el caso de que al controlador se intente conectar un *switch* que no posea la versión OpenFlow establecida.

```
308 #Clase que contiene la funcionalidad para la escritura de flujos en un switch
309 class Firewall(object):
310     _OFCTL = {ofproto_v1_3.OFP_VERSION: ofctl_v1_3}
311     def __init__(self, dp):
312         super(Firewall, self).__init__()
313         self.dp = dp
314         version = dp.ofproto.OFP_VERSION
315         self.cookie = 0
316         #Maneja el error de version OpenFlow no compatible en un switch a traves de una excepcion
317         if version not in self._OFCTL:
318             raise OFPUnknownVersion(version=version)
319         self.ofctl = self._OFCTL[version]
```

Código 2.16. Declaración de la clase `Firewall`

En el Código 2.17, se puede observar la definición del método `activar_flujo`, el cual es utilizado para activar el envío de flujos en un *switch*.

```
352    #Funcion que activa  el envio de flujos en un switch
353    def activar_flujo(self):
354        cookie = 0
355        prioridad = PRIORIDAD_ESTADO_FLUJO
356        match = {}
357        actions = []
358        flow = self._conv_a_flujo(cookie=cookie, priority=prioridad,
359                                  match=match, actions=actions)
360        cmd = self.dp.ofproto.OFPFC_DELETE_STRICT
361        self.ofctl.mod_flow_entry(self.dp, flow, cmd)
362        msg = {'resultado': 'correcto',
363               'detalles': 'firewall ejecutandose.'}
364        switch_id = '%s: %s' % ('switch_id',
365                                dpid_lib.dpid_to_str(self.dp.id))
366        return {switch_id: msg}
```

Código 2.17. Método `activar_flujo`

En el Código 2.18, se puede observar la definición del método `ingresar_regla`, el cual es utilizado para establecer una regla de envío de tráfico del firewall en un *switch*, además realiza el manejo de las excepciones para reglas ingresadas no válidas.

```
379    def ingresar_regla(self, cookie, rest):
380        prioridad = int(rest.get('prioridad', 0))
381        if prioridad < 0 or PRIORIDAD_FLUJO_ACL_MAX < prioridad:
382            raise ValueError('Valor no valido de prioridad. Ingrese de [0-%d]'
383                             % PRIORIDAD_FLUJO_ACL_MAX)
384        match = Match.conv_a_accion_of(rest)
385        actions = Accion.conv_a_accion_of(self.dp, rest)
386        flow = self._conv_a_flujo(cookie=cookie, priority=prioridad,
387                                  match=match, actions=actions)
388        cmd = self.dp.ofproto.OFPFC_ADD
389        try:
390            self.ofctl.mod_flow_entry(self.dp, flow, cmd)
391        except:
392            raise ValueError('Parametro de regla invalido.')
393        msg = {'resultado': 'correcto',
394               'detalles': 'Regla ingresada. : regla_id=%d' % cookie}
395        switch_id = '%s: %s' % ('switch_id',
396                                dpid_lib.dpid_to_str(self.dp.id))
397        return {switch_id: msg}
```

Código 2.18. Método `ingresar_regla`

2.6.4.6 Clase para manejar la operación emparejamiento del *switch*

La clase `Match` realiza el manejo de la operación de emparejamiento del *switch*, convirtiendo acciones que provengan de la API REST a acciones que se realizan en los *switches* OpenFlow en una SDN y viceversa. En el Código 2.19, se puede observar la declaración la clase `Match`, donde se encuentra la definición de un

diccionario que contiene los diferentes tipos de protocolos como por ejemplo: TCP, UDP, ICMP, etc; con los cuales se realizará la tarea de emparejamiento de reglas en el *switch*.

```
477 class Match(object):
478     """Se crea un diccionario con los protocolos que soportara el firewall
479     para el establecimiento de reglas de reenvio de trafico"""
480     _CONVERT = {'dl_type':
481                 {'ARP': ether.ETH_TYPE_ARP,
482                  'IPv4': ether.ETH_TYPE_IP},
483                 'nw_proto':
484                 {'TCP': inet.IPPROTO_TCP,
485                  'UDP': inet.IPPROTO_UDP,
486                  'ICMP': inet.IPPROTO_ICMP}}
```

Código 2.19. Definición de la clase `Match`

En el Código 2.20, se puede observar la definición del método estático `conv_a_accion_of`, el cual es utilizado para convertir de una regla de la REST API a una regla de reenvió de tráfico OpenFlow a partir de los protocolos que intervendrán en el tráfico de la red.

```
487     """Metodo estatico para convertir de una regla de la REST API a una regla de reenvio
488     de trafico OpenFlow a partir de los protocolos que intervedran en el trafico de la red"""
489     @staticmethod
490     def conv_a_accion_of(rest):
491         match = {}
492         set_dltype_flg = False
493         for key, value in rest.items():
494             if (key == 'nw_src' or key == 'nw_dst'
495                     or key == 'nw_proto'):
496                 if ('dl_type' in rest) is False:
497                     set_dltype_flg = True
498                 elif (rest['dl_type'] != 'IPv4'
499                       and rest['dl_type'] != 'ARP'):
500                     continue
501             elif key == 'tp_src' or key == 'tp_dst':
502                 if (('nw_proto' in rest) is False
503                     or (rest['nw_proto'] != 'TCP'
504                         and rest['nw_proto'] != 'UDP')):
505                     continue
506             if key in Match._CONVERT:
507                 if value in Match._CONVERT[key]:
508                     match.setdefault(key, Match._CONVERT[key][value])
509                 else:
510                     raise ValueError('Parametro de regla invalido. : key=%s' % key)
511             else:
512                 match.setdefault(key, value)
513             if set_dltype_flg:
514                 match.setdefault('dl_type', ether.ETH_TYPE_IP)
515         return match
```

Código 2.20. Método `conv_a_accion_of`

En el Código 2.21, se puede observar la definición del método estático `conv_a_comando_rest`, el cual es utilizado para convertir de una regla de

reenvío de tráfico OpenFlow a una regla de la REST API a partir de los protocolos que intervendrán en el tráfico de la red.

```
516   """Metodo estatico para convertir de una regla de reenvio de trafico OpenFlow
517   a una regla de la REST API a partir de los protocolos que intervendran en el trafico de la red"""
518   @staticmethod
519   def conv_a_comando_rest(openflow):
520       of_match = openflow['match']
521       match = {}
522       for key, value in of_match.items():
523           if key == 'dl_src' or key == 'dl_dst':
524               if value == mac.haddr_to_str(mac.DONTCARE):
525                   continue
526           elif key == 'nw_src' or key == 'nw_dst':
527               if value == '0.0.0.0':
528                   continue
529           elif value == 0:
530               continue
531           if key in Match._CONVERT:
532               conv = Match._CONVERT[key]
533               conv = dict((value, key) for key, value in conv.items())
534               match.setdefault(key, conv[value])
535           else:
536               match.setdefault(key, value)
537       return match
```

Código 2.21. Método estático `conv_a_comando_rest`

2.6.4.7 Clase para manejar la operación acción del *switch*

La clase `Accion`, que se puede apreciar en el Código 2.22, realiza el manejo de la operación acción del *switch*, convirtiendo acciones que provengan de la API REST a acciones que se realizan en los *switches* OpenFlow en una SDN y viceversa.

```
553 """Clase que realiza las conversiones entre las distintas acciones de la REST API
554  y las acciones  OpenFlow"""
555 class Accion(object):
556     #Metodo estatico para convertir las acciones de la REST API a una accion OpenFlow
557     @staticmethod
558     def conv_a_accion_of(dp, rest):
559         value = rest.get('actions', 'ACEPTAR')
560         if value == 'ACEPTAR':
561             out_port = dp.ofproto.OFPP_NORMAL
562             action = [{'type': 'OUTPUT',
563                        'port': out_port}]
564         elif value == 'RECHAZAR':
565             action = []
566         else:
567             raise ValueError('Tipo de accion invalida.')
568         return action
569     #Metodo estatico para convertir las acciones OpenFlow a una accion  de la REST API
570     @staticmethod
571     def conv_a_comando_rest(openflow):
572         if 'actions' in openflow:
573             if len(openflow['actions']) > 0:
574                 action = {'actions': 'ACEPTAR'}
575             else:
576                 action = {'actions': 'RECHAZAR'}
577         else:
578             action = {'actions': 'Unknown action type.'}
579 
580         return action
```

Código 2.22. Clase `Accion`

La clase contiene dos métodos:

- Método `conv_a_accion_of`: Convierte una acción que provengan de la API REST a una acción que se realice en los *switches* OpenFlow.
- Método `conv_a_comando_rest`: Convierte una acción que se realice en los *switches* OpenFlow a una consulta o respuesta que provenga de la API REST.

2.6.5 IMPLEMENTACIÓN DE LA INTERFAZ GRÁFICA CLIENTE FIREWALL

La interfaz gráfica de usuario del cliente para la configuración del firewall llamada **`cliente_firewall.py`** se desarrolló utilizando Python, el código fuente en su totalidad se encuentra disponible en el **Anexo F**. A continuación se describe brevemente la funcionalidad de la interfaz.

2.6.5.1 Sección de ingreso de reglas de reenvió de trafico

En la sección de ingreso de reglas de reenvió de tráfico es posible actualizar la topología de red mediante el control de usuario de tipo botón llamado “Actualizar red”.También se puede activar o desactivar todo el tráfico de la red empleando un control de usuario de tipo interruptor, además es posible ingresar los parámetros de las reglas en el firewall, los cuales son los siguientes: dirección MAC fuente, dirección MAC destino, dirección IP fuente, dirección IP destino, tipo de protocolo (TCP, UDP o ICMP), número de puerto fuente, número de puerto destino, identificador del *switch* o AP OpenFlow y la acción de aceptar o rechazar el tráfico. Para ingresar la regla se debe utilizar el control de usuario de tipo botón llamado “Ingresar regla”.

Cuando alguno de los campos de la sección no es ingresado, el valor por defecto es “cualquiera” representado por el símbolo “*”. En la Figura 2.57 se presenta la sección y sus componentes.

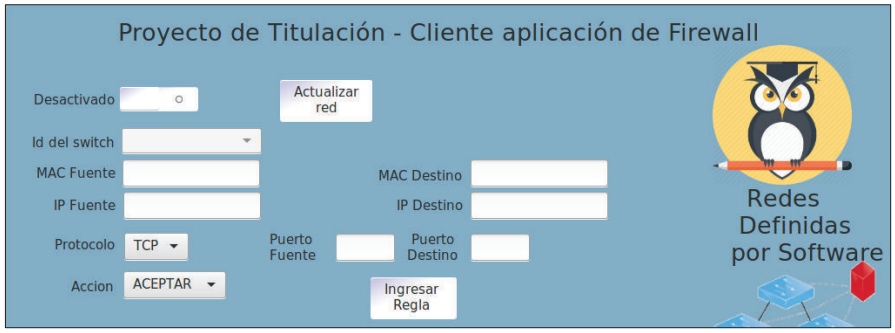

Figura 2.57. Sección de ingreso de reglas

2.6.5.2 Sección para visualización de reglas de reenvió de trafico

En la sección para visualización de reglas de reenvió de tráfico es posible visualizar las reglas que se hayan ingresado en la aplicación de firewall, las reglas se visualizarán por *switch* o AP, esto a partir de su identificador.

Adicionalmente es posible eliminar las reglas ingresadas mediante el control de usuario de tipo botón llamado “Borrar regla”. En la Figura 2.58 se puede observar la sección y sus componentes.

Figura 2.58. Sección para visualización de reglas

CAPÍTULO 3: PRUEBAS Y RESULTADOS

En este capítulo se presenta un conjunto de pruebas básicas que se realizaron para comprobar el intercambio de flujos entre los dispositivos implementados OpenFlow *switch*, AP y controlador. Además se presentan pruebas de generación de tráfico, tanto en la simulación, como en el prototipo real para analizar el *throughput* y el *jitter* en la SDN propuesta. Posteriormente se presentan pruebas del funcionamiento de la aplicación de firewall y los costos referenciales del prototipo.

3.1 PRUEBAS DE INTERCAMBIO DE FLUJOS OPENFLOW EN LOS DISPOSITIVOS DE IMPLEMENTADOS

Para las pruebas realizadas, sobre el controlador SDN implementado se ejecutó la aplicación de ejemplo **`simple_switch13.py`**, que se incluye en el framework Ryu, para que los dispositivos conectados al controlador actúen como *switch* de capa 2 y se produzca un intercambio de flujos de prueba.

Cuando se realiza un intercambio de flujos OpenFlow, existen varios mensajes básicos que se intercambian entre el *switch* OpenFlow y el controlador SDN [38], dicho intercambio de mensajes se puede apreciar en la Figura 3.1. En la presente sección se analizará el intercambio de mensajes capturados utilizando el analizador de tráfico de red Wireshark [39].

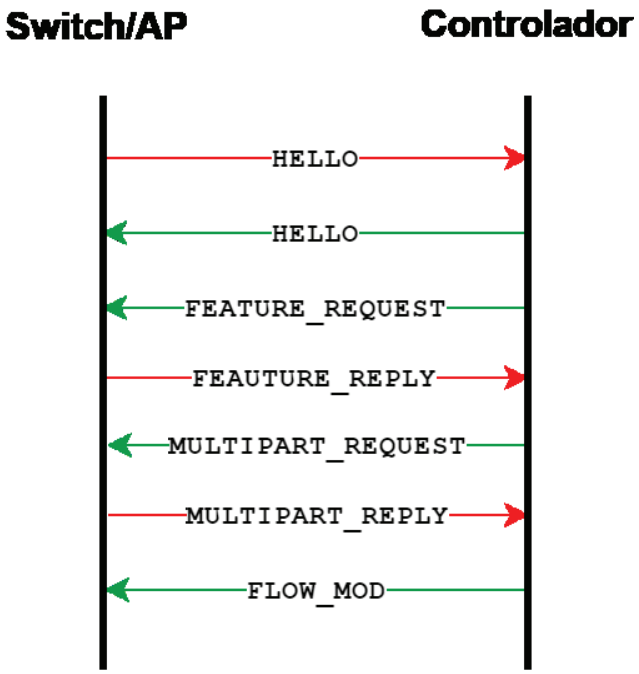

Figura 3.1. Mensajes entre dispositivos OpenFlow y el controlador

A continuación se presenta una breve descripción de los mensajes OpenFlow [19] de la Figura 3.1, los cuales se analizarán posteriormente:

- **Hello:** Son mensajes intercambiados entre el controlador y el *Switch*, se utilizan para establecer la conexión entre los dos.
- **Features:** Se usa para establecer la conexión con el *Switch* mediante el protocolo TLS, con esta solicitud el *switch* proporciona al controlador los parámetros para establecer la conexión.
- **Multipart:** Se utiliza para que el controlador pida al *switch* información sobre varios aspectos de su estado.
- **Flow-modify:** Se envía cuando una entrada es añadida a la tabla de flujos de un *switch*.

3.1.1 PRUEBA DE INTERCAMBIO DE FLUJOS OPENFLOW EN LOS DISPOSITIVOS IMPLEMENTADOS

Para las pruebas se utilizó dos topologías de red de prueba: la primera se compone de las implementaciones del controlador y el switch OpenFlow y de dos estaciones, tal como se muestra en la Figura 3.2, y la segunda dispone de las implementaciones del controlador y el AP OpenFlow y de dos estaciones, como se muestra en la Figura 3.3,

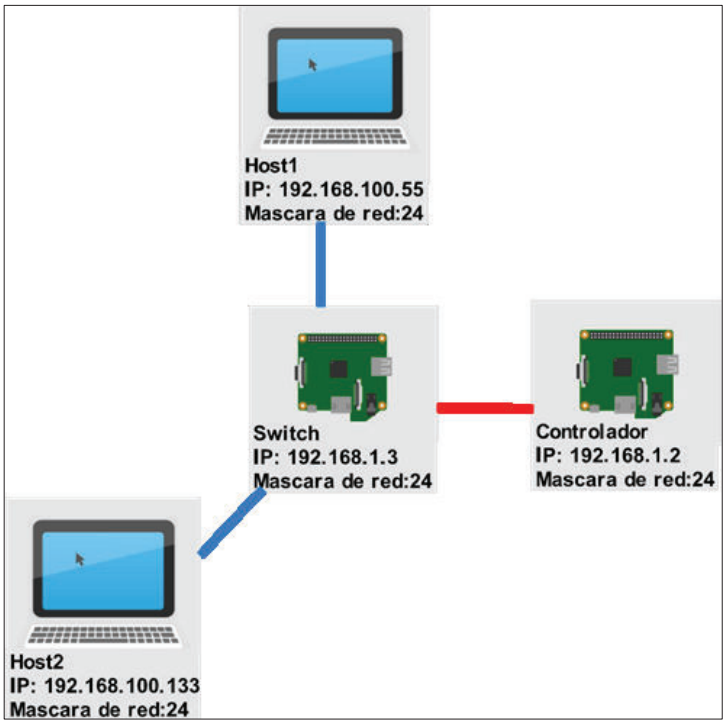

Figura 3.2. Topología de la SDN para prueba de flujos en el *switch*

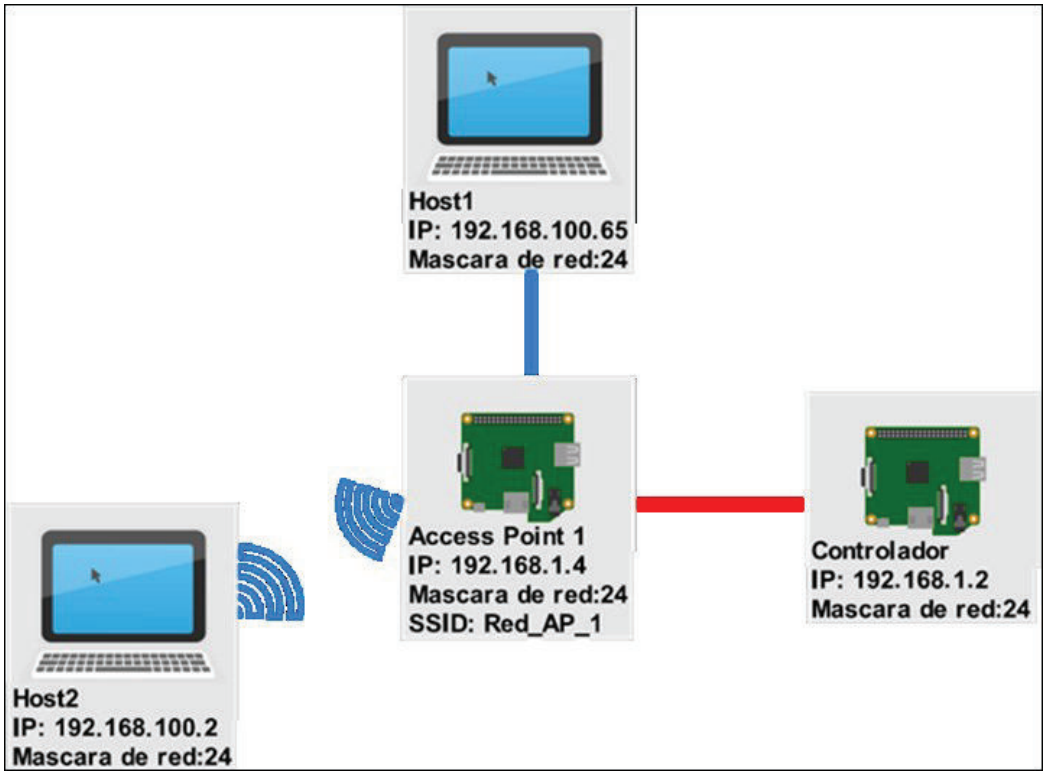

Figura 3.3. Topología de la SDN para prueba de flujos en el AP

3.1.1.1 Mensajes HELLO entre los dispositivos y el controlador

En la Figura 3.4 y Figura 3.5 se puede observar el intercambio de mensajes del tipo **HELLO** [40], entre los dispositivos y el controlador planteados en los escenarios de la la Figura 3.2 y Figura 3.3. A continuación en la Tabla 3.1 se presentan de manera resumida los campos capturados durante las pruebas realizadas.

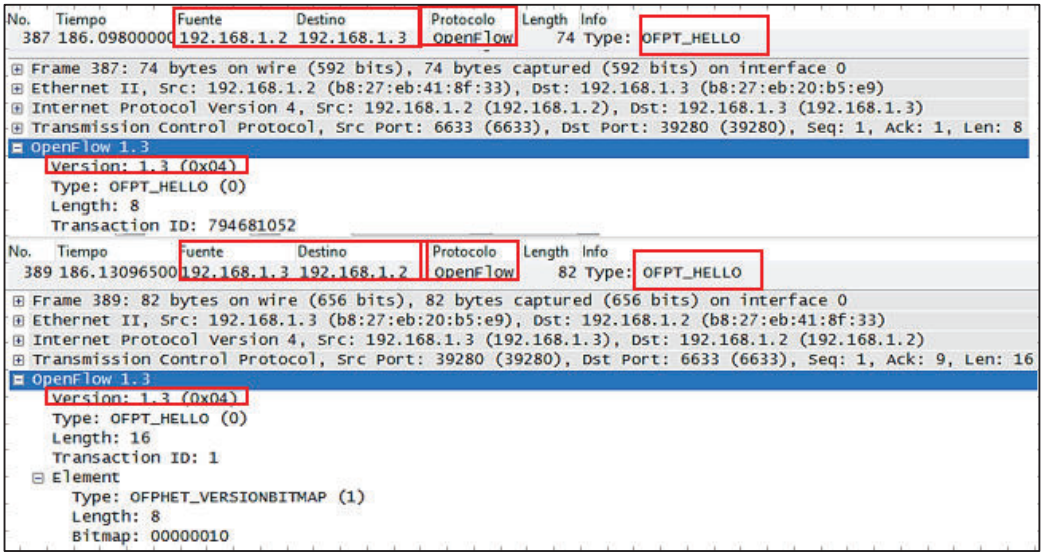

Figura 3.4. Mensajes **HELLO** entre el *switch* y el controlador

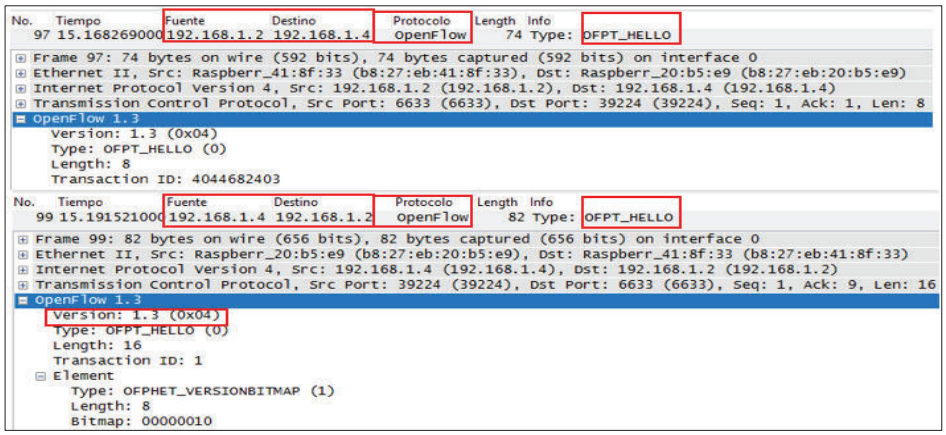

Figura 3.5. Mensajes **HELLO** entre el AP y el controlador

Tabla 3.1. Campos presentes en el intercambio de mensajes HELLO

Dispositivo	Petición/Respuesta	Protocolo	Versión	Tipo de mensaje
Controlador	Petición	OpenFlow	0x04(v 1.3.0)	HELLO
Switch	Respuesta	OpenFlow	0x04(v 1.3.0)	HELLO
AP	Respuesta	OpenFlow	0x04(v 1.3.0)	HELLO

3.1.1.2 Mensajes FEATURES entre los dispositivos y el controlador

En la Figura 3.6 y Figura 3.7 se puede observar el intercambio de mensajes del tipo **FEATURES_REQUEST** y **FEATURES_REPLY** [41], entre los dispositivos y el controlador planteados en los escenarios de la la Figura 3.2 y Figura 3.3. A continuación en la Tabla 3.2 se presentan de manera resumida los campos capturados durante las pruebas realizadas.

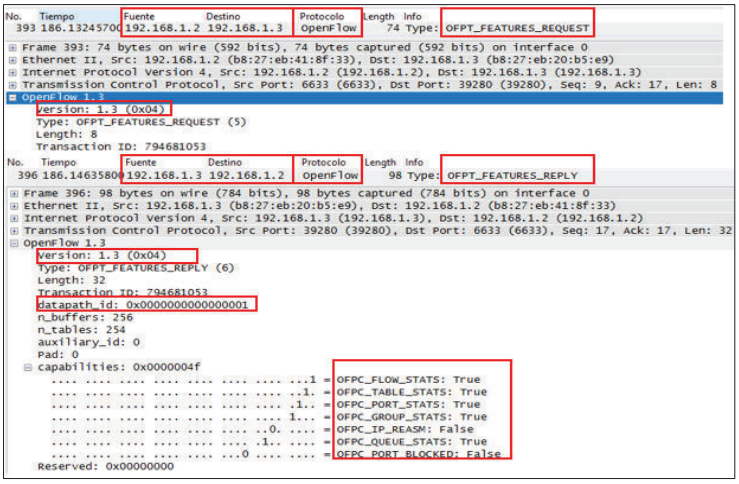

Figura 3.6. Mensajes FEATURES entre el *switch* y el controlador

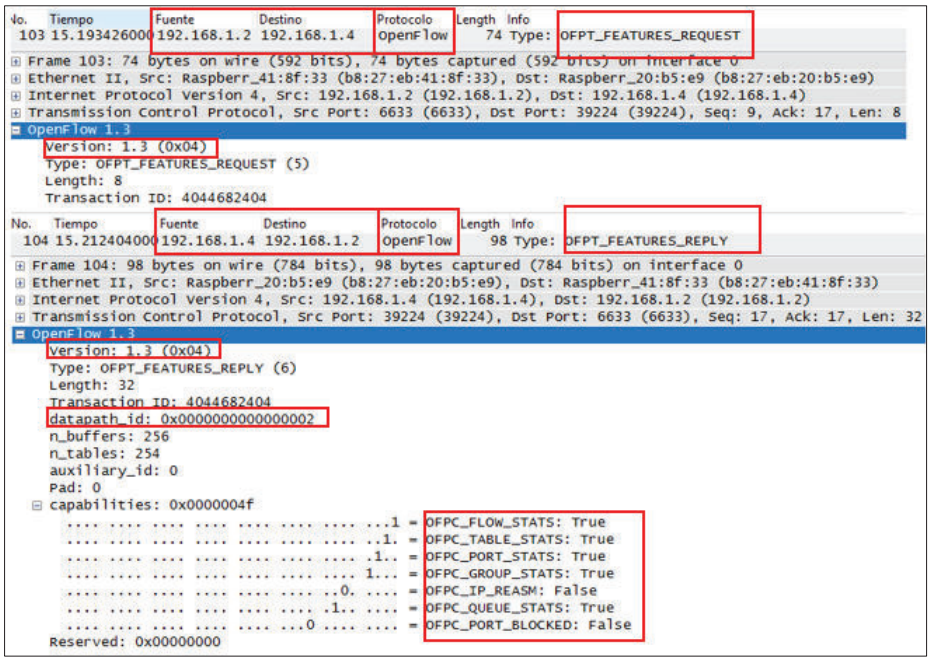

Figura 3.7. Mensajes `FEATURES` entre el AP y el controlador

Tabla 3.2. Campos presentes en el intercambio de mensajes `FEATURES`

Dispositivo	Controlador	Switch	AP
Petición/Respuesta	Petición	Respuesta	Respuesta
Protocolo	OpenFlow	OpenFlow	OpenFlow
Versión	0x04(v 1.3.0)	0x04(v 1.3.0)	0x04(v 1.3.0)
Tipo de mensaje	**`FEATURES_REQUEST`**	**`FEATURES_REPLY`**	**`FEATURES_REPLY`**
Respuesta	No aplica	Id del datapath y características del *switch*.	Id del datapath y características del AP.

3.1.1.3 Mensajes MULTIPART entre los dispositivos y el controlador

En la Figura 3.8 y Figura 3.9 se puede observar el intercambio de mensajes del tipo **`MULTIPART_REQUEST`** y **`MULTIPART_REPLY`** [42], entre los dispositivos y el controlador planteados en los escenarios de la la Figura 3.2 y Figura 3.3. A continuación en la Tabla 3.3 se presentan de manera resumida los campos capturados durante las pruebas realizadas.

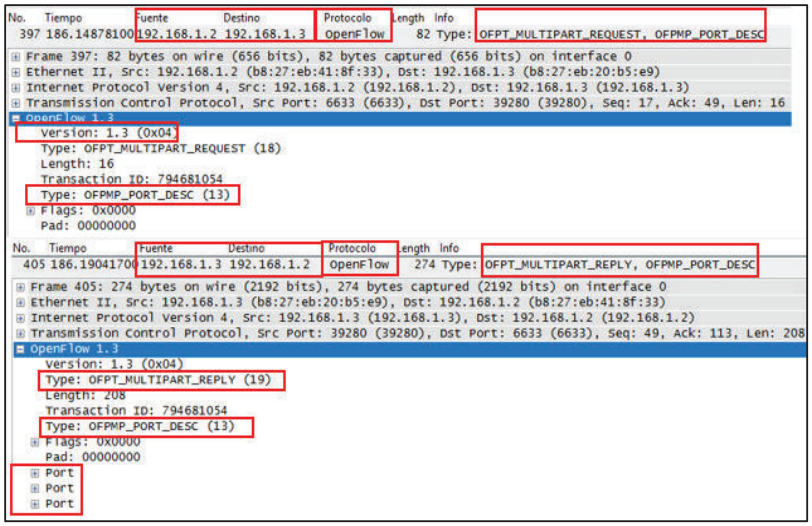

Figura 3.8. Mensajes `MULTIPART` entre el *switch* y el controlador

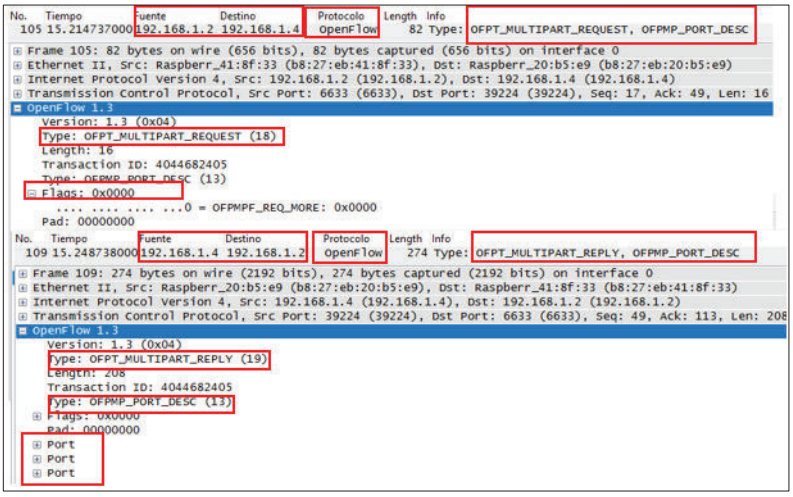

Figura 3.9. Mensajes `MULTIPART` entre el AP y el controlador

Tabla 3.3 Campos presentes en el intercambio de mensajes **`MULTIPART`**

Dispositivo	Controlador	Switch	AP
Petición/Respuesta	Petición	Respuesta	Respuesta
Protocolo	OpenFlow	OpenFlow	OpenFlow
Versión	0x04(v 1.3.0)	0x04(v 1.3.0)	0x04(v 1.3.0)
Tipo de mensaje	`MULTIPART_REQUEST`	`MULTIPART_REPLY`	`MULTIPART_REPLY`
Tipo de requerimiento	`PORT_DESC`	`PORT_DESC`	`PORT_DESC`
Respuesta	No aplica	Descripción de los puertos OpenFlow del *switch*.	Descripción de los puertos OpenFlow del AP.

3.1.1.4 Mensaje FLOW_MOD del controlador hacia los dispositivos

En la Figura 3.10 y Figura 3.11 se puede observar el envío del mensaje del tipo **FLOW_MOD** [43] por parte del controlador al *switch* y al AP, planteados en los escenarios de la Figura 3.2 y Figura 3.3, respectivamente.

A continuación en la Tabla 3.4 se presentan de manera resumida los campos capturados durante las pruebas realizadas.

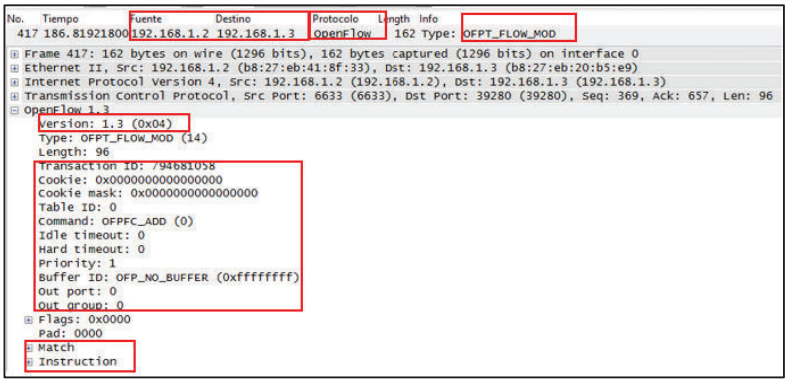

Figura 3.10. Mensaje FLOW_MOD del controlador hacia el *switch*

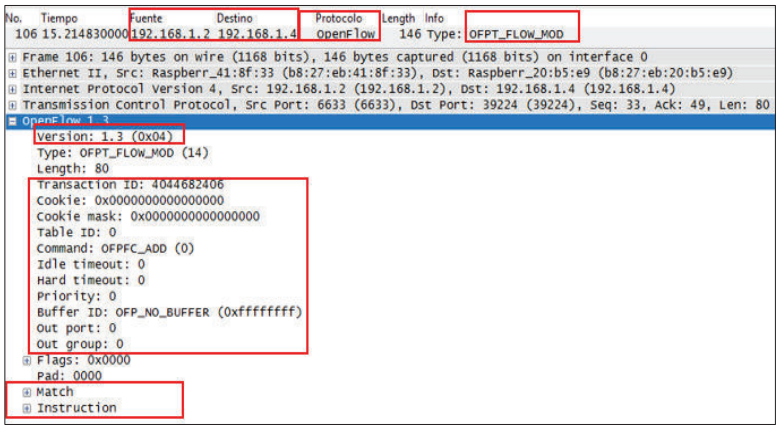

Figura 3.11. Mensaje FLOW_MOD del controlador hacia el AP

Tabla 3.4. Campos presentes en el envío del mensaje **FLOW_MOD**

Mensaje	Controlador-Switch	Controlador-AP
Protocolo	OpenFlow	OpenFlow
Tipo de mensaje	FLOW_MOD	FLOW_MOD
Parámetros a insertar en la tabla de flujo	Cookie, prioridad, puerto de salida, información de emparejamiento e instrucciones de manejo de flujos.	Cookie, prioridad, puerto de salida, información de emparejamiento e instrucciones de manejo de flujos.

3.2 PRUEBAS DE GENERACIÓN DE TRÁFICO DE DATOS EN LA SDN

Para las pruebas de generación de tráfico de datos, se utilizará la aplicación **`simple_switch_13.py`**, empleada en la sección 3.1, sobre el controlador SDN implementado, para que los dispositivos conectados al controlador actúen como *switch* de capa 2. Además para generar y medir el tráfico de red se empleará el software **`iperf`** [44] el cual ofrece la posibilidad de medir tanto tráfico del tipo TCP y UDP. La topología de la SDN se encuentra implementada de acuerdo al diseño propuesto en la sección 2.3, la cual se puede observar en la Figura 3.12. Además en la Tabla 3.5, se muestran las características más importantes de los computadores portátiles(laptops) utilizados para realizar las pruebas en el prototipo.

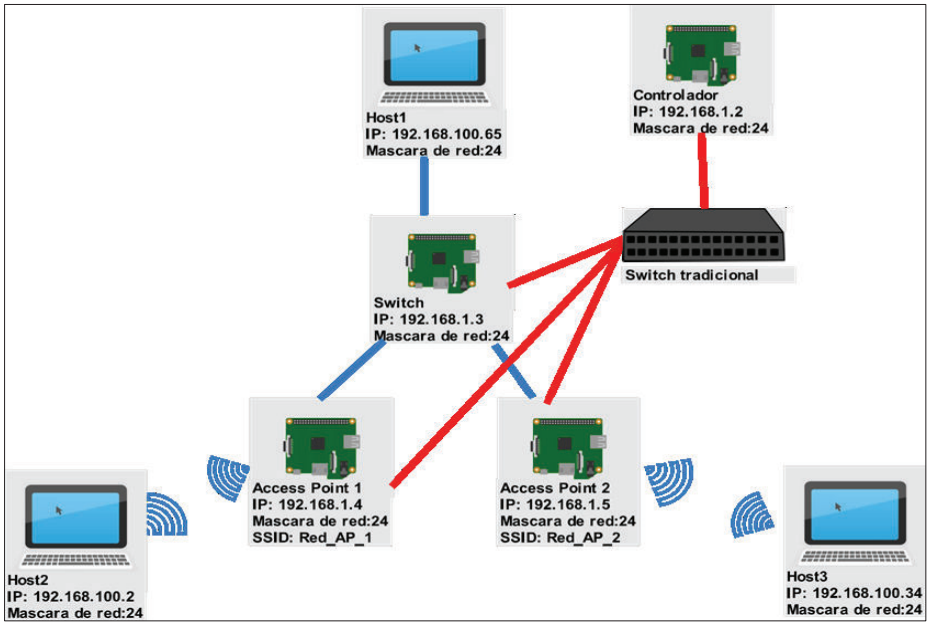

Figura 3.12. Topología de la SDN implementada

Tabla 3.5. Características de laptops para pruebas en el prototipo

Marca	Modelo	Arquitectura	Procesador	Memoria RAM	Sistema Operativo	Identificación en la red
Compaq	515	64 bits	Intel Core 2 Duo	3 GB	Ubuntu 14.04	Host1
Toshiba	Satellite C55t-b	64 bits	Intel Core i3 @ 1.70GHz	4 GB	Ubuntu 14.04	Host2
Compaq	515	64 bits	Intel Core 2 Duo	1 GB	Ubuntu 14.04	Host3

3.2.1 PRUEBAS DE THROUGHPUT TCP EN EL PROTOTIPO

El software **iperf** permite generar y medir el tráfico en una red, el Comando 3.1 y Comando 3.2, han sido utilizados para medir *throughput* del tipo TCP. Iperf funciona en modo cliente - servidor, es decir necesariamente se necesita de un servidor que recepte el tráfico generado, pero pueden conectarse varios clientes simultáneos que generen tráfico. En el presente trabajo, utilizando la topología de la Figura 3.12, se han planteado dos escenarios: el primero consiste en realizar pruebas de tráfico entre dos estaciones inalámbricas (Host2 y Host3) y el segundo consiste en realizar pruebas de tráfico entre una estación inalámbrica(Host2) y una cableada(Host1). Todos los valores de *throughput*, *jitter*, uso de CPU y consumo de RAM, que se encuentran representados en las gráficas de la presente sección, se encuentran presentes en el **Anexo G**.

```
$ iperf -s
```

Comando 3.1. Servidor de tráfico TCP en el software **iperf**

```
$ iperf -c [IP del servidor de tráfico]
```

Comando 3.2. Cliente de tráfico TCP en el software **iperf**

3.2.1.1 Pruebas de throughput TCP entre dos estaciones inalámbricas

Para las pruebas se utilizó el Comando 3.1 y Comando 3.2 , generando el máximo tráfico soportado por las interfaces de red durante un periodo de 300 segundos, es decir cinco minutos y los resultados que se obtuvieron tanto en la simulación como en el prototipo real se pueden apreciar en la Figura 3.13.

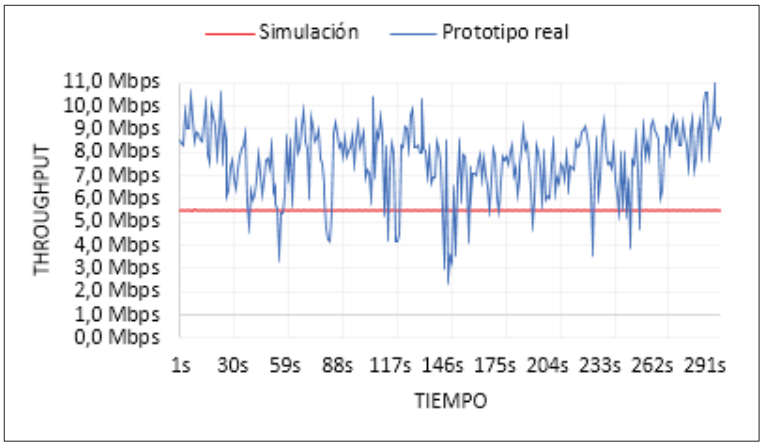

Figura 3.13. *Throughput* TCP entre dos estaciones inalámbricas

Analizando la Figura 3.13, se puede observar que en la simulación la generación de tráfico TCP permanece de manera regular con un valor aproximado de 5,5Mbps a lo largo de todo el periodo de análisis, mientras que en el prototipo real hay

variaciones propias de una comunicación donde intervienen dos nodos inalámbricos (dos AP) y también debido a problemas en las actualizaciones de entradas en las tablas de flujos tanto a nivel de paquete [45] o de flujo [46] de cada *switch* o AP. Adicionalmente se puede apreciar que en este escenario se alcanzó un mayor *throughput* TCP en el prototipo real que en la simulación, debido a que las interfaces inalámbricas que utiliza Mininet-WiFi se encuentran programadas para alcanzar un *throughput* de aproximadamente 6Mbps.

En la Figura 3.14 se puede apreciar que en el escenario de prueba el AP_1 es el que hizo un uso mayor del CPU en comparación de los dispositivos implementados, ya que por el AP_1 cursó la mayor cantidad de tráfico de datos. En cuanto al consumo de memoria RAM el controlador es el dispositivo que hizo un mayor uso de este recurso, debido a que este maneja todas las conexiones de los dispositivos OpenFlow (*switch* y AP), todo esto se muestra en la Figura 3.15.

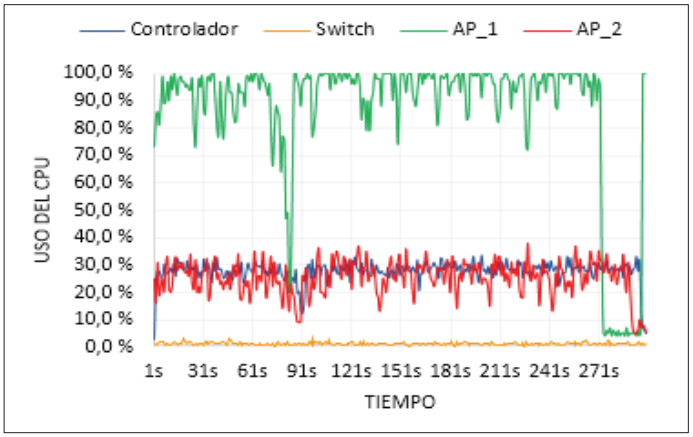

Figura 3.14. CPU de los dispositivos durante tráfico TCP entre dos estaciones inalámbricas

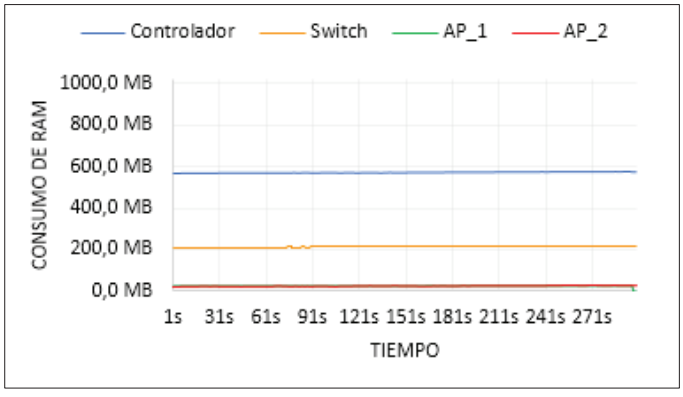

Figura 3.15. RAM de los dispositivos durante tráfico TCP entre dos estaciones inalámbricas

3.2.1.2 Pruebas de throughput TCP entre una estación inalámbrica y una estación cableada

Para las pruebas se utilizó el Comando 3.1 y Comando 3.2, generando el máximo tráfico soportado por las interfaces de red durante un periodo de 300 segundos es decir cinco minutos, los resultados que se obtuvieron tanto en la simulación como en el prototipo real se pueden apreciar en la Figura 3.16

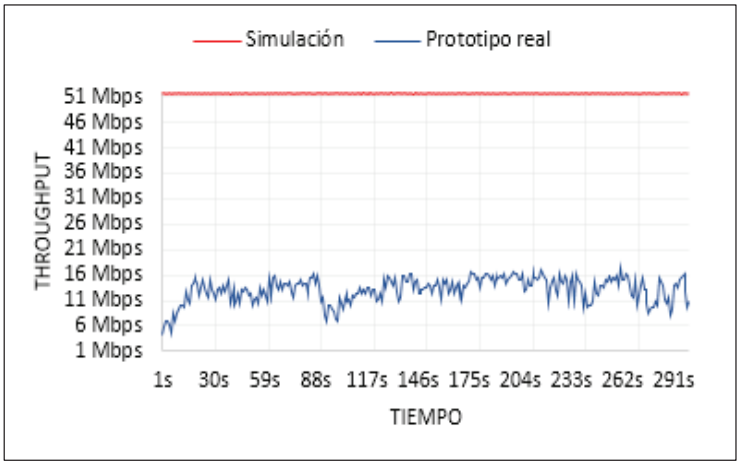

Figura 3.16. *Throughput* TCP entre una estación inalámbrica y una cableada

Analizando la Figura 3.16, se puede observar que en la simulación la generación de tráfico TCP permanece de manera regular con un valor aproximado de 50Mbps a lo largo de todo el periodo de análisis, mientras que en el prototipo real hay variaciones propias de una comunicación de donde interviene un nodo inalámbrico(un AP) y del problema de una ineficiente actualización de las tablas de flujo en los dispositivos que ya se mencionó en la sección 3.2.1.1. Además se puede apreciar que en este escenario se alcanzó un mayor *throughput* TCP en la simulación que en el prototipo real, y definitivamente un mayor *throughput* comparado al obtenido al escenario planteado anteriormente, debido a que la interfaz Fast-Ethernet que utiliza Mininet-WiFi se encuentra programada para alcanzar un *throughput* de aproximadamente 50Mbps.

En la Figura 3.17 se puede apreciar que en el escenario de prueba el AP_1 es el que hizo un uso mayor del CPU en comparación de los dispositivos implementados, ya que por el AP_1 cursó la mayor cantidad de tráfico de datos. En cuanto al consumo de memoria RAM el controlador es el dispositivo que hizo un mayor uso de este recurso, debido a que este manejaba todas las conexiones de los dispositivos OpenFlow (*switch* y AP), tal como se muestra en la Figura 3.18.

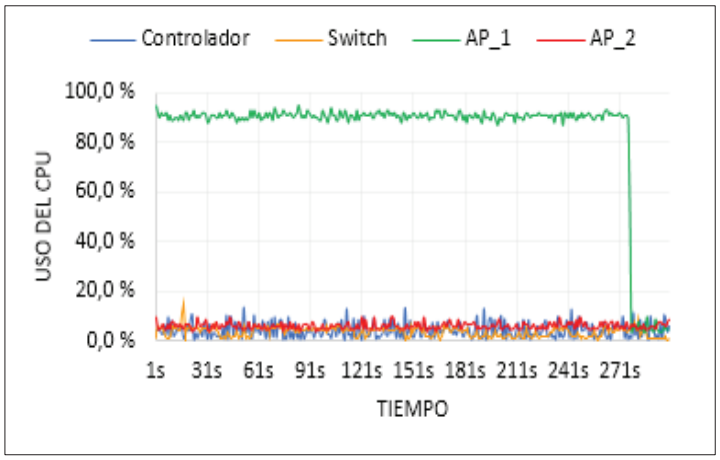

Figura 3.17. CPU de los dispositivos durante tráfico TCP entre una estación cableada y una inalámbrica

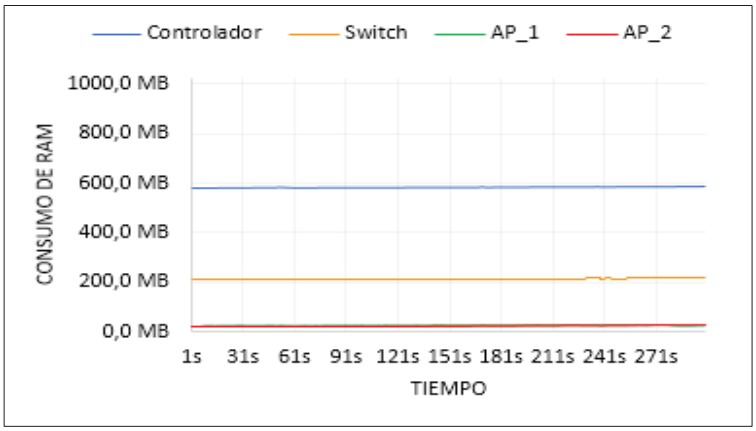

Figura 3.18. RAM de los dispositivos durante tráfico TCP entre una estación cableada y una inalámbrica

3.2.2 PRUEBAS DE THROUGHPUT UDP EN EL PROTOTIPO

El software **`iperf`** permite generar y medir el tráfico en una red, el Comando 3.3 y Comando 3.4, que se utilizaron para medir *throughput* del tipo UDP, funcionan de la manera cliente - servidor, es decir necesariamente se debe utilizar un servidor que recepte el tráfico generado, al cual pueden conectarse varios clientes simultáneos que generen tráfico. En el presente trabajo, utilizando la topología de la Figura 3.12, se han planteado dos escenarios: el primero consiste en realizar pruebas de tráfico entre dos estaciones inalámbricas (Host2 y Host3) y el segundo consiste en realizar pruebas de tráfico entre una estación inalámbrica(Host2) y una cableada(Host1).

```
$ iperf -s -u
```

Comando 3.3. Servidor de tráfico UDP en el software **`iperf`**

```
$ iperf -c [IP del servidor de tráfico] -u
```

Comando 3.4. Cliente de tráfico UDP en el software **`iperf`**

3.2.2.1 Pruebas de throughput UDP entre dos estaciones inalámbricas

Para las pruebas se utilizó el Comando 3.3 y Comando 3.4, , generando el máximo tráfico soportado por las interfaces de red durante un periodo de 300 segundos es decir cinco minutos, los resultados que se obtuvieron tanto en la simulación como en el prototipo real se pueden apreciar en la Figura 3.19.

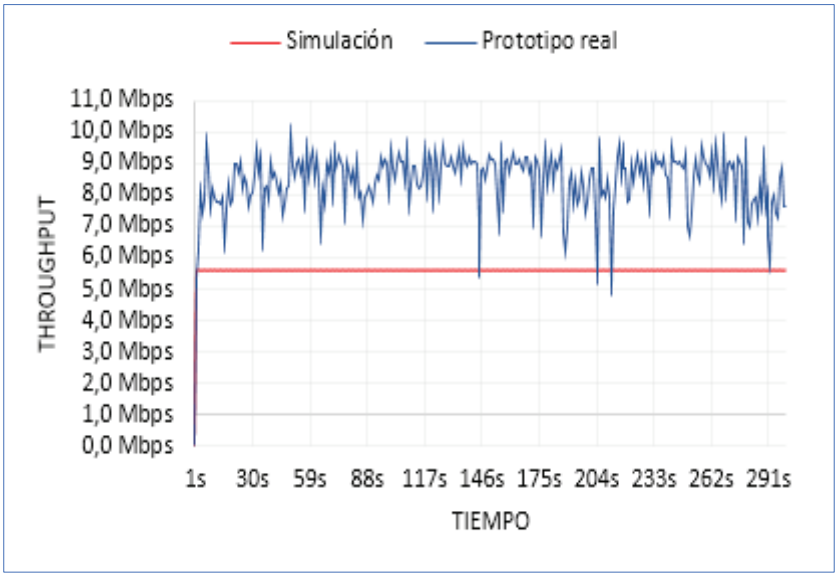

Figura 3.19. *Throughput* UDP entre dos estaciones inalámbricas

Analizando la Figura 3.19, se puede observar que en la simulación, la generación de tráfico UDP permanece de manera regular con un valor aproximado de 5,5Mbps a lo largo de todo el periodo de análisis, mientras que en el prototipo real existen variaciones propias de una comunicación donde existen dos AP además del problema de las actualizaciones de las tablas de flujo mencionado con anterioridad. Se puede apreciar que en este escenario se alcanzó un mayor *throughput* UDP en el prototipo real que en la simulación, debido a que las interfaces inalámbricas que utiliza Mininet-WiFi se encuentran programadas para alcanzar un *throughput* de aproximadamente 6Mbps.

En la Figura 3.20 se puede apreciar que en el escenario de prueba el AP_1 es el que hizo un uso mayor del CPU en comparación al resto de los dispositivos implementados, ya que por el AP_1 cursó la mayor cantidad de tráfico de datos. En cuanto al consumo de memoria RAM el controlador es el dispositivo que hizo un mayor uso de este recurso, debido a que este manejaba todas las conexiones de

los dispositivos OpenFlow (*switch* y AP), lo mencionado anteriormente se muestra en la Figura 3.21.

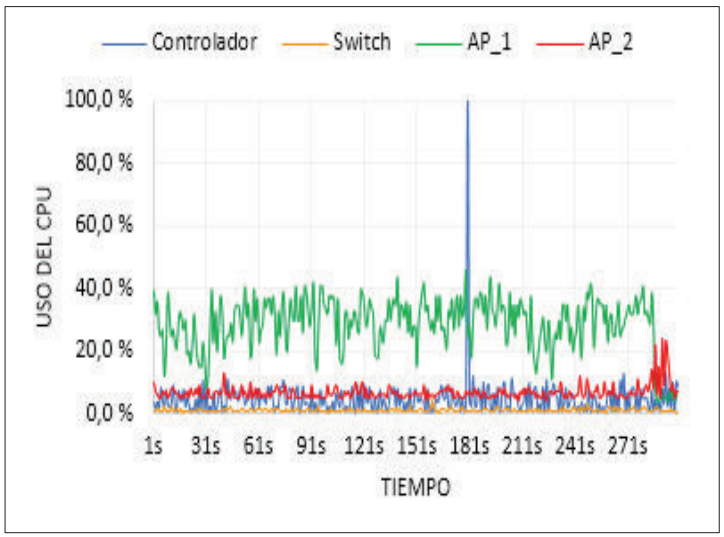

Figura 3.20. CPU de los dispositivos durante tráfico UDP entre dos estaciones inalámbricas

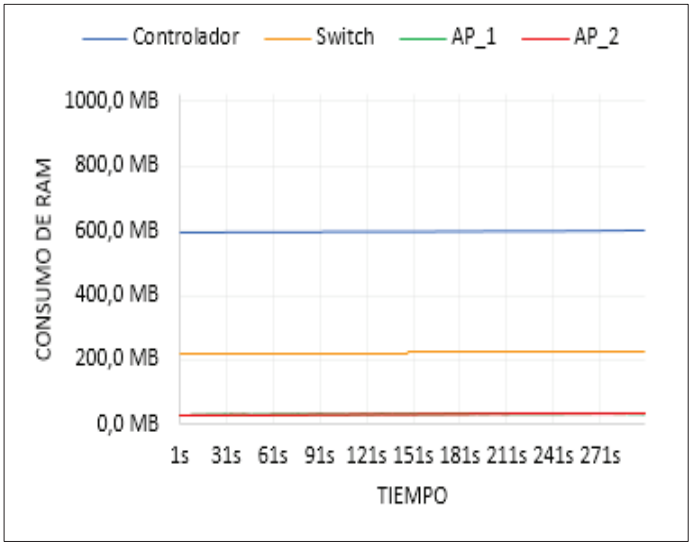

Figura 3.21. RAM de los dispositivos durante tráfico UDP entre dos estaciones inalámbricas

3.2.2.2 Pruebas de throughput UDP entre una estación inalámbrica y una estación cableada

Para las pruebas se utilizó el Comando 3.3 y Comando 3.4, , generando el máximo tráfico soportado por las interfaces de red durante un periodo de 300 segundos es decir cinco minutos, los resultados que se obtuvieron tanto en la simulación como en el prototipo real se pueden apreciar en la Figura 3.22.

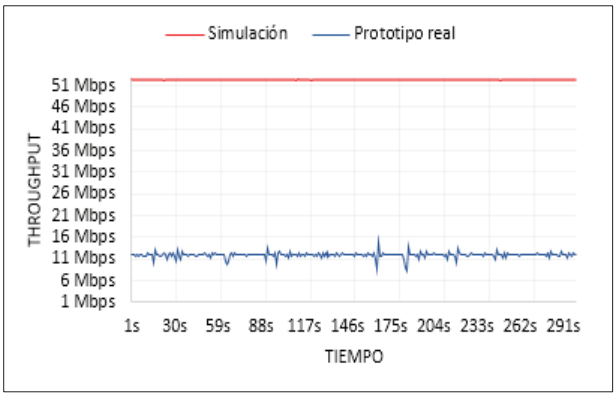

Figura 3.22. *Throughput* UDP entre una estación inalámbrica y una cableada

Analizando la Figura 3.22, se puede observar que en la simulación la generación de tráfico UDP permanece de manera regular con un valor aproximado de 52Mbps a lo largo de todo el periodo de análisis, mientras que en el prototipo real existen variaciones por la utilización de al menos un AP y de problemas relacionados con el funcionamiento de una SDN anteriormente ya mencionados. Además se puede apreciar que en este escenario se alcanzó un mayor *throughput* UDP en la simulación que en el prototipo real, debido a que la interfaz Fast-Ethernet que utiliza Mininet-WiFi se encuentra programada para alcanzar un *throughput* de aproximadamente 52Mbps, el cual se pudo apreciar en la prueba.

En la Figura 3.23 se puede apreciar que en el escenario de prueba el AP_1 es el que hizo un uso mayor del CPU en comparación a los demás dispositivos implementados, ya que por el AP_1 cursó la mayor cantidad de tráfico de datos. En cuanto al consumo de memoria RAM el controlador es el dispositivo que hizo un mayor uso de este recurso, debido a que este manejaba todas las conexiones de los dispositivos OpenFlow (*switch* y AP), tal como se muestra en la Figura 3.24.

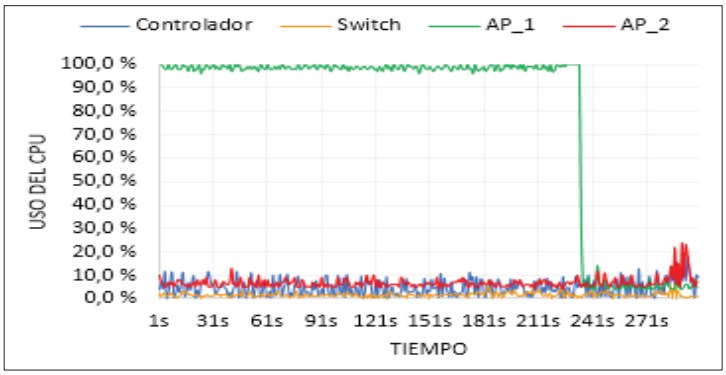

Figura 3.23. CPU de los dispositivos durante tráfico UDP entre una estación cableada y una inalámbrica

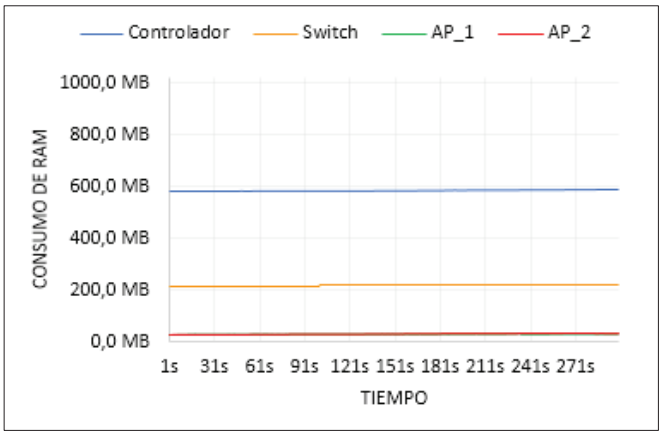

Figura 3.24. RAM de los dispositivos durante tráfico UDP entre una estación cableada y una inalámbrica

3.2.3 PRUEBAS DE JITTER EN EL PROTOTIPO

El software **`iperf`** permite generar y medir el tráfico en una red, el Comando 3.3 y Comando 3.4, utilizados para medir el tráfico UDP adicionalmente permiten medir el *jitter* en una red, en el presente trabajo, utilizando la topología de la Figura 3.12, se han planteado dos escenarios: el primero consiste en realizar pruebas de tráfico entre dos estaciones inalámbricas (Host2 y Host3) y el segundo consiste en realizar pruebas de tráfico entre una estación inalámbrica(Host2) y una cableada(Host1).

3.2.3.1 Pruebas de jitter entre dos estaciones inalámbricas

Para las pruebas se utilizó el Comando 3.3 y Comando 3.4, generando el máximo tráfico soportado por las interfaces de red durante un periodo de 300 segundos es decir cinco minutos, los resultados que se obtuvieron tanto en la simulación como en el prototipo real se pueden apreciar en la Figura 3.25.

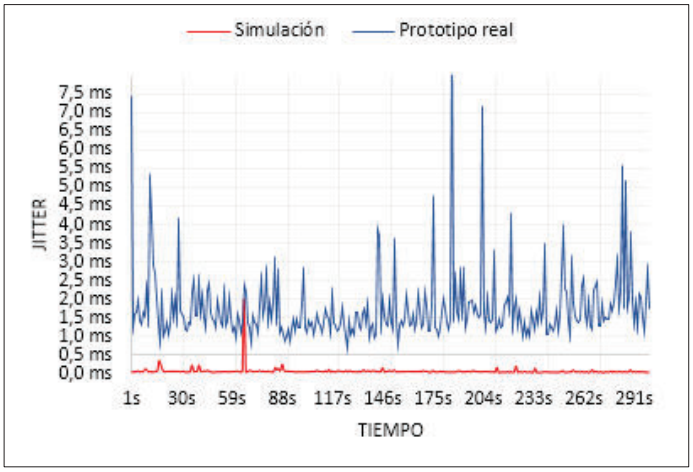

Figura 3.25. *Jitter* entre dos estaciones inalámbricas

Analizando la Figura 3.25, se puede apreciar que en este escenario se presentó mayor cantidad de *jitter* en el prototipo real que en la simulación.

3.2.3.2 Pruebas de jitter entre una estación inalámbrica y una estación cableada

Para las pruebas se utilizó el Comando 3.3 y Comando 3.4, generando el máximo tráfico soportado por las interfaces de red durante un periodo de 300 segundos es decir cinco minutos, los resultados que se obtuvieron tanto en la simulación como en el prototipo real se pueden apreciar en la Figura 3.26.

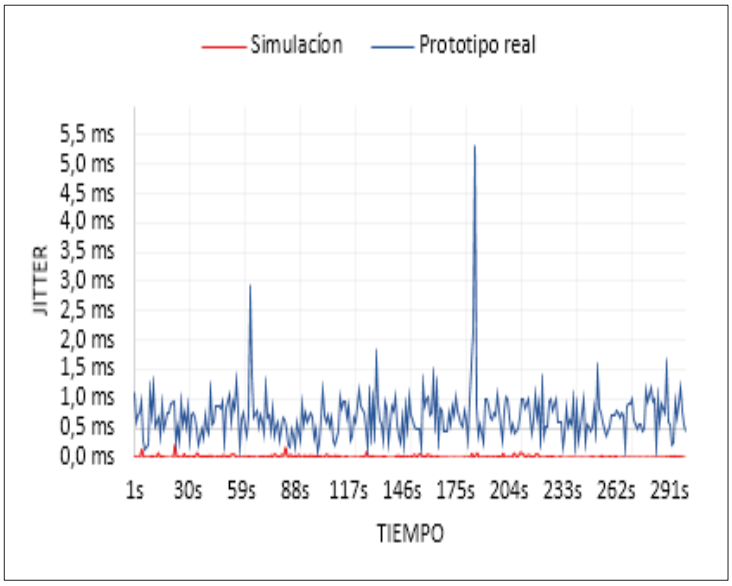

Figura 3.26. *Jitter* entre una estación inalámbrica y una cableada

Analizando la Figura 3.26, se puede apreciar que en este escenario se presentó mayor cantidad de *jitter* en el prototipo real que en la simulación y además una menor cantidad de *jitter* comparándolo con el anterior escenario, dado que intervienen dos AP y una comunicación inalámbrica es más propensa al *jitter* que una comunicación cableada.

3.2.4 RESUMEN DE LOS RESULTADOS OBTENIDOS EN LAS PRUEBAS DE TRÁFICO EN EL PROTOTIPO

A continuación en la Tabla 3.6, se presenta un resumen de los resultados obtenidos en las pruebas realizadas. El resumen de los resultados se encuentra compuesto de los valores promedio obtenidos en cada una de las pruebas de throughput TCP, UDP y jitter. De manera general se puede apreciar que los valores de throughput UDP son más altos que el throughput TCP, tanto en simulación como en el prototipo real debido a que el protocolo UDP, a diferencia del protocolo TCP, no utiliza acuse de recibo lo que permite un flujo de paquetes más continuo, alcanzando así un throughput mayor.

Tabla 3.6. Resultados obtenidos en las pruebas de tráfico

Prueba realizada	Valor promedio alcanzado en la simulación	Valor promedio alcanzado en el prototipo real
Throughput TCP entre dos estaciones inalámbricas	5,50 Mbps	7,61 Mbps
Throughput TCP entre una estación inalámbrica y una estación cableada	51,65 Mbps	13,16 Mbps
Throughput UDP entre dos estaciones inalámbricas	5,59 Mbps	8,46 Mbps
Throughput UDP entre una estación inalámbrica y una estación cableada	52,50 Mbps	11,91 Mbps
Jitter entre dos estaciones inalámbricas	0,049 ms	1,753 ms
Jitter entre una estación inalámbrica y una estación cableada	0,021 ms	0,700 ms

3.3 PRUEBAS DE LA APLICACIÓN DE FIREWALL EN EL PROTOTIPO

Con el objetivo de probar la funcionalidad de la aplicación de firewall, de la sección 2.6.4, sobre el prototipo implementado, se utilizarán los servicios o servidores que se especifican en la Tabla 3.7. Para iniciar tanto la aplicación de firewall, como la interfaz gráfica de usuario, se ejecuta el Comando 2.41 y el Comando 2.6, respectivamente.

Tabla 3.7. Servicios o servidores de prueba implementados en la SDN

Equipo	Dirección IP	Dirección MAC	Servidor o servicio implementado	Puerto TCP/UDP utilizado
Host1	192.168.100.65	18:A9:05:E4:86:0C	HTTP	80
Host2	192.168.100.1	B8:EE:65:16-01:78	FTP	20 y 21
Host3	192.168.100.34	C4:17:FE:20:21:B6	SSH	22

3.3.1 PRUEBA DE BLOQUEO DE TRÁFICO EN TODA LA SDN

Para habilitar o bloquear todo el tráfico en la aplicación de firewall sobre la SDN, se debe utilizar el control del tipo interruptor localizado en la interfaz gráfica. Además para probar que el tráfico está bloqueado se debe insertar una regla para aceptar el tráfico especificando en este caso paquetes del tipo ICMP, como se puede observar en la Figura 3.27.

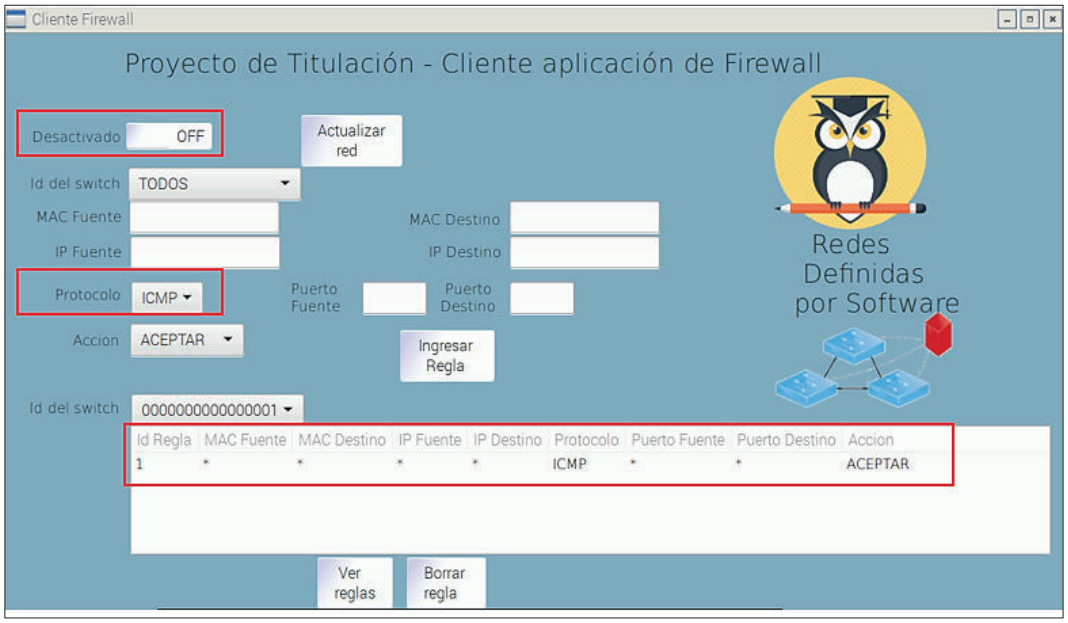

Figura 3.27. Bloqueo de todo el tráfico en toda la red

Posteriormente se puede probar la conectividad de la red con el comando ping. Como se muestra en la Figura 3.28,donde se puede apreciar que el tráfico incluido el tráfico de paquetes ICMP se encuentra bloqueado, es decir los paquetes ICMP no pueden llegar a su destino (*Destination Host Unreacheable*).

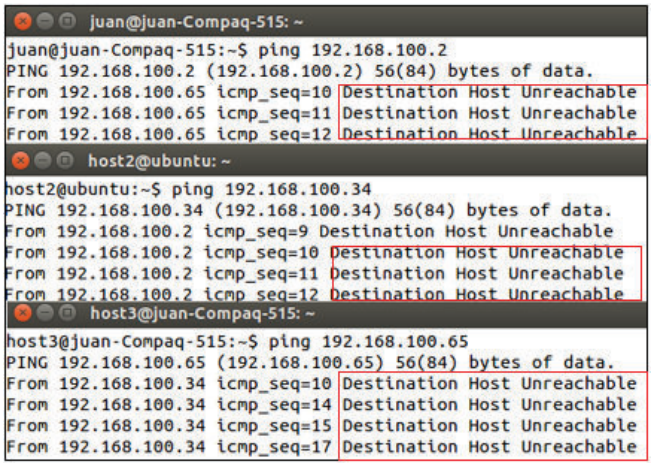

Figura 3.28. Paquetes ICMP bloqueados en toda la red

Luego para habilitar todo el tráfico en la aplicación de firewall, se debe accionar el control de interruptor, tal como se muestra en la Figura 3.29, finalmente en la Figura 3.30 se puede observar que todos los hosts de la red poseen conectividad.

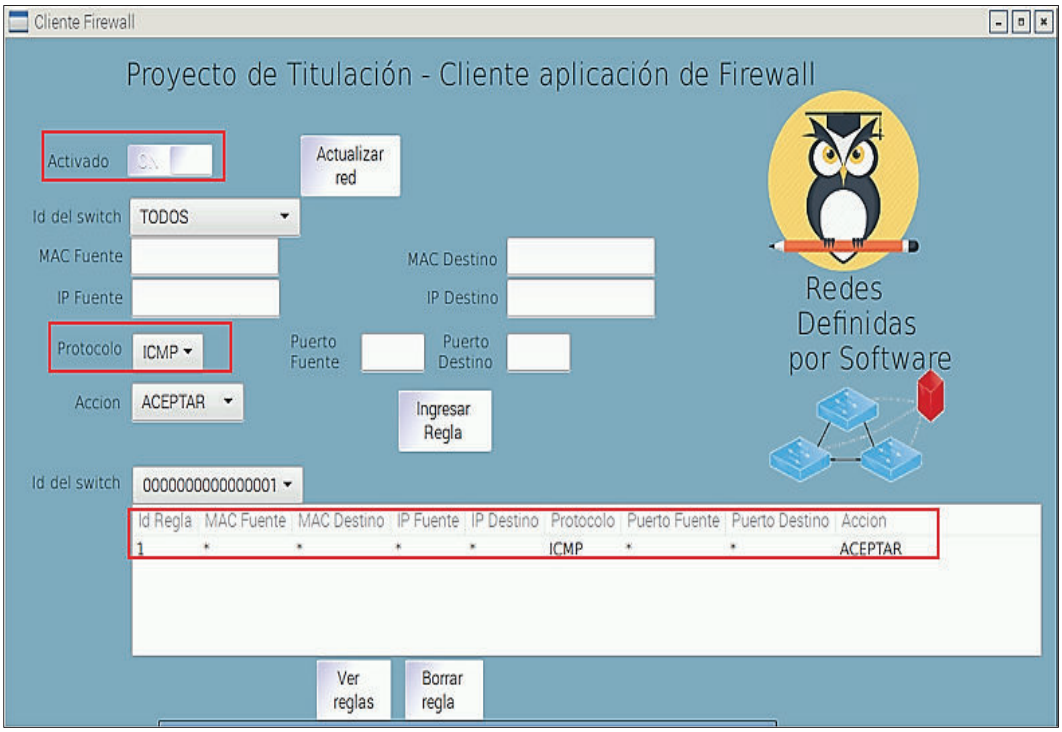

Figura 3.29. Desbloqueo de todo el tráfico en la red

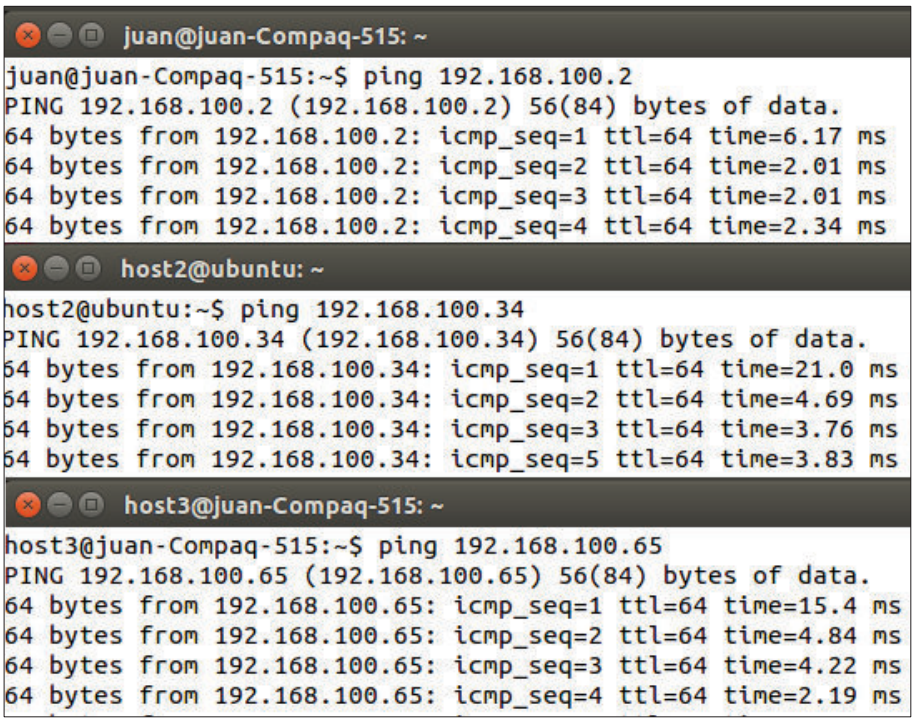

Figura 3.30. Prueba de conectividad exitosa en toda la red

3.3.2 PRUEBA DE INSERCIÓN DE REGLAS PARA ACEPTAR O RECHAZAR EL TRÁFICO HACIA EL SERVIDOR HTTP

Como primer punto se deberá ingresar las reglas a la aplicación del firewall, como se muestra en la Tabla 3.8, donde se decide que equipos tendrán acceso o tendrán restricción al servidor HTTP, cuya dirección es la IP 192.168.100.65. En la Figura 3.31, se puede apreciar el ingreso de las reglas del firewall en la interfaz gráfica

Tabla 3.8. Reglas de tráfico del tipo HTTP a ingresar en la aplicación de firewall

IP Fuente	IP Destino	Protocolo	# Puerto Fuente	# Puerto Destino	Acción
192.168.100.2	192.168.100.65	TCP	*	80	ACEPTAR
192.168.100.65	192.168.100.2	TCP	80	*	ACEPTAR

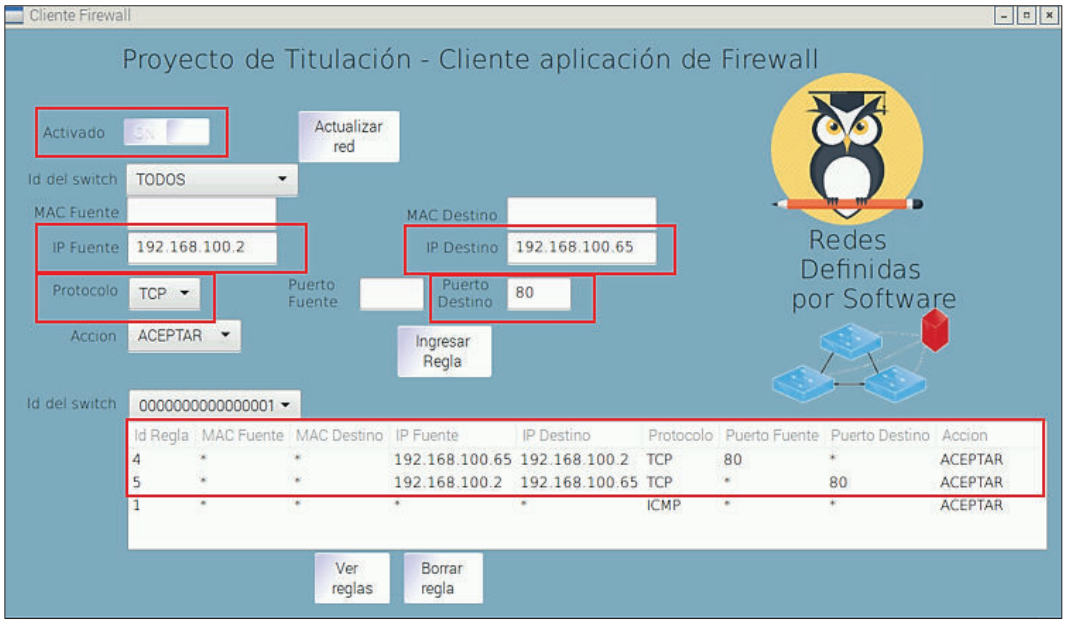

Figura 3.31. Ingreso de reglas HTTP en la interfaz gráfica

Mediante la aplicación de firewall se proveerá el acceso al servidor HTTP, al Host2 con dirección IP 192.168.100.2 y se bloqueará el acceso a el resto de hosts de la red, como por ejemplo, el Host3 con dirección IP 192.168.100.34, como se muestra en la Figura 3.32 y Figura 3.33.

Figura 3.32. Acceso exitoso al servidor web de prueba por parte del Host2

Figura 3.33. Acceso bloqueado al servidor web de prueba por parte del Host3

3.3.3 PRUEBA DE INSERCIÓN DE REGLAS PARA ACEPTAR O RECHAZAR EL TRÁFICO HACIA EL SERVIDOR FTP

Como primer punto se deberán ingresar las reglas a la aplicación del firewall, como se muestra en la Tabla 3.9, donde se aprecia que equipos tendrán acceso o tendrán restricción de acceder al servidor FTP, cuya dirección es la IP 192.168.100.2. En la Figura 3.34 , se muestra el ingreso de las reglas del firewall en la interfaz gráfica

Tabla 3.9. Reglas de tráfico del tipo FTP a ingresar en la aplicación de firewall

IP Fuente	IP Destino	Protocolo	# Puerto Fuente	# Puerto Destino	Acción
192.168.100.65	192.168.100.2	TCP	*	20	ACEPTAR
192.168.100.2	192.168.100.65	TCP	20	*	ACEPTAR
192.168.100.65	192.168.100.2	TCP	*	21	ACEPTAR
192.168.100.2	192.168.100.65	TCP	21	*	ACEPTAR

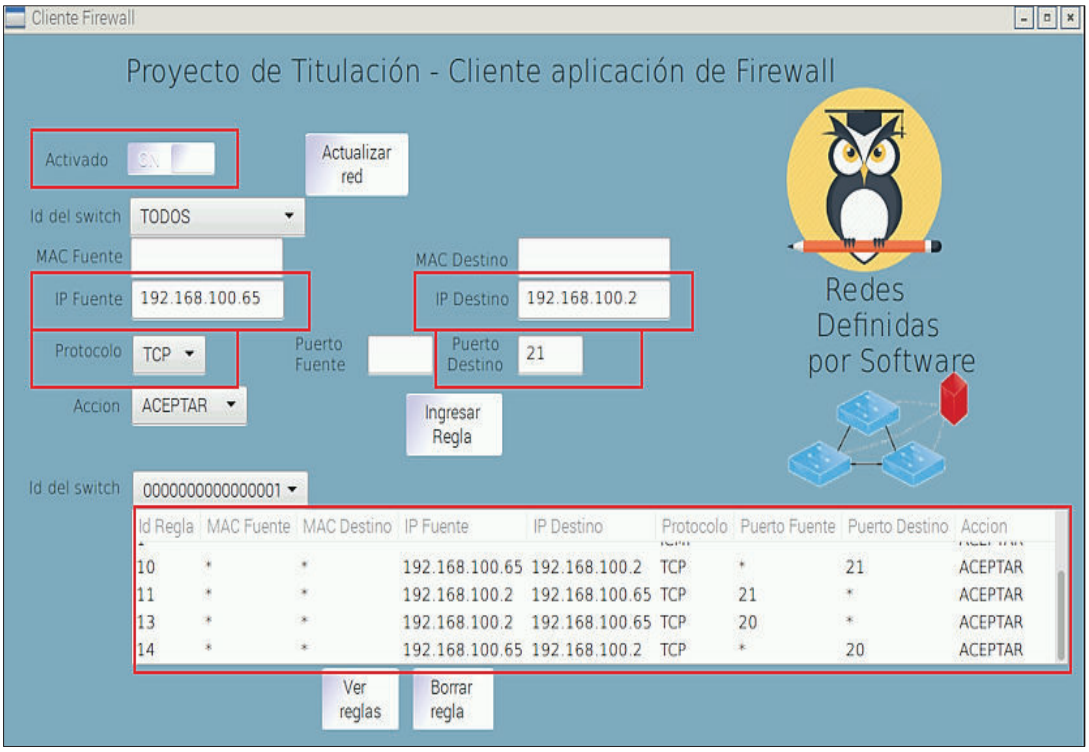

Figura 3.34. Ingreso de reglas FTP en la interfaz gráfica

Mediante la aplicación de firewall se proveerá el acceso al servidor FTP, al Host1 con dirección IP 192.168.100.65 y se bloqueará el acceso a el resto de hosts de la red, como por ejemplo, el acceso al Host3 con dirección IP 192.168.100.34, como se muestra en la Figura 3.35 y Figura 3.36.

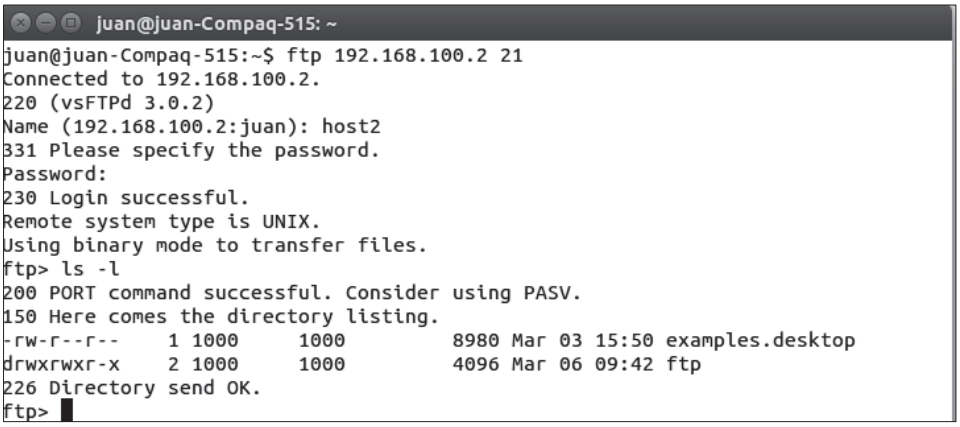

Figura 3.35. Acceso exitoso al servidor FTP de prueba por parte del Host1

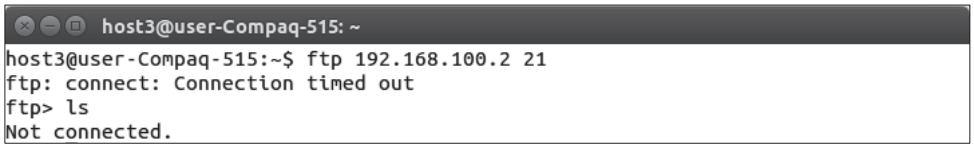

Figura 3.36. Acceso bloqueado al servidor FTP de prueba por parte del Host3

3.3.4 PRUEBA DE INSERCIÓN DE REGLAS PARA ACEPTAR O RECHAZAR EL TRÁFICO HACIA EL SERVICIO SSH

Como primer punto se deberá ingresar las reglas a la aplicación del firewall, como se muestra en la Tabla 3.10, donde se aprecia que equipos tendrán acceso o tendrán restricción de acceder al servicio SSH, cuya dirección es la IP 192.168.100.34. En la Figura 3.37, se puede observar el ingreso de las reglas del firewall en la interfaz gráfica.

Tabla 3.10. Reglas de tráfico del tipo SSH a ingresar en la aplicación de firewall

IP Fuente	IP Destino	Protocolo	# Puerto Fuente	# Puerto Destino	Acción
192.168.100.65	192.168.100.34	TCP	*	22	ACEPTAR
192.168.100.34	192.168.100.65	TCP	22	*	ACEPTAR

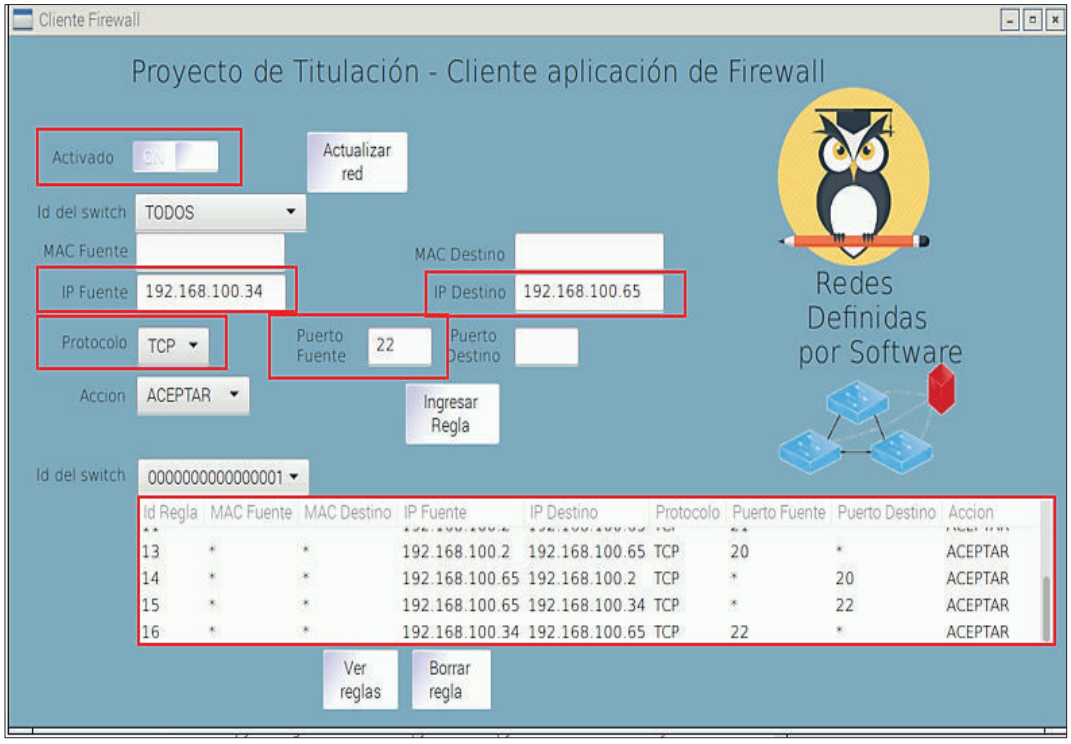

Figura 3.37. Ingreso de reglas SSH en la interfaz gráfica

Mediante la aplicación de firewall se proveerá el acceso al servicio SSH, al Host1 con dirección IP 192.168.100.65 y se bloqueará el acceso al resto de hosts de la red, como por ejemplo, el acceso al Host2 con dirección IP 192.168.100.2, como se muestra en la Figura 3.38 y Figura 3.39.

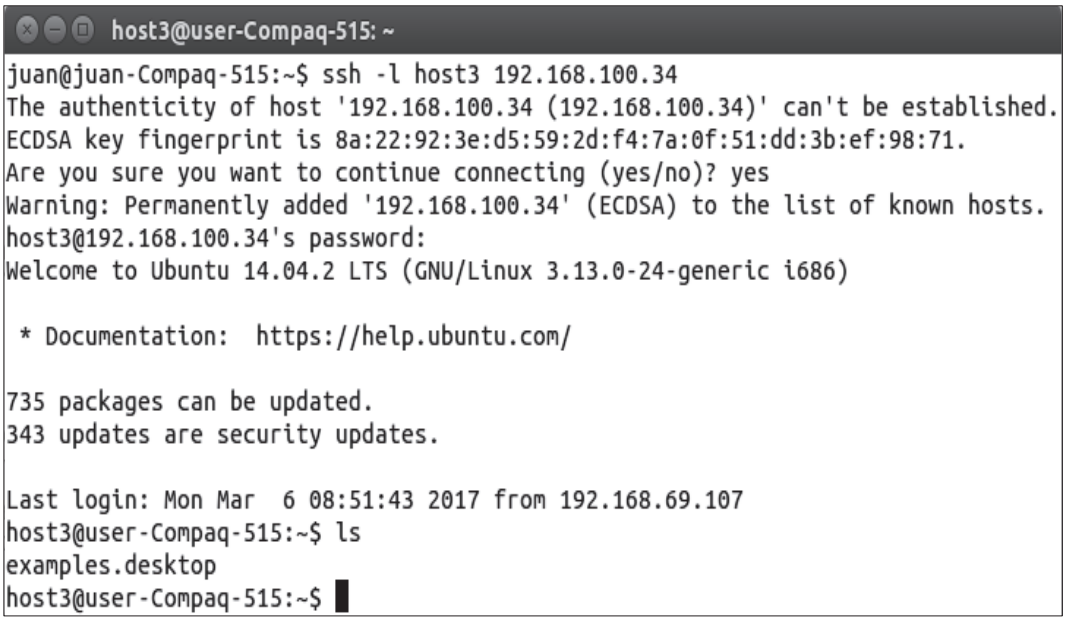

Figura 3.38. Acceso exitoso al servidor SSH de prueba por parte del Host1

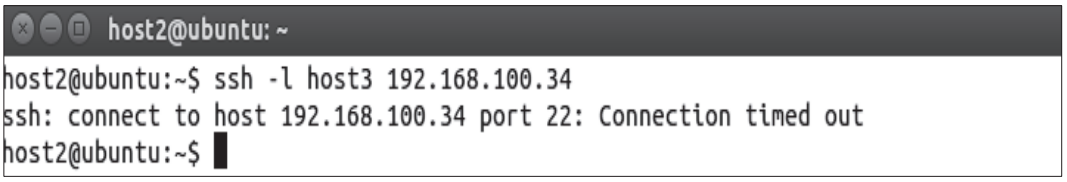

Figura 3.39. Acceso bloqueado al servidor SSH de prueba por parte del Host2

3.4 COSTO DEL PROTOTIPO

A partir de los pasos que se ha propuesto en el presente trabajo, se puede establecer los costos de los dos dispositivos básicos implementados, los cuales son un *switch* OpenFlow y un AP OpenFlow, adicionalmente se presenta algunas alternativas de *switches* OpenFlow disponibles en el mercado, cabe indicar que al ser opciones comerciales fabricadas por grandes marcas, presentan mejores prestaciones que los dispositivos implementados de este trabajo. El objetivo de presentar estas opciones es conocer cuánto costaría adquirir uno de estos equipos sabiendo que los mismos serán usados para experimentación. En muchos de los casos para personas interesadas en practicar con esta tecnología o estudiantes que quieran probar sus aplicaciones y desarrollar soluciones basadas en OpenFlow, adquirir uno de estos equipos está fuera de su alcance económico.

A continuación en la Tabla 3.11, se presenta una comparación de precios de los dispositivos implementados y opciones de *switches* OpenFlow disponibles en el mercado.

Tabla 3.11. Comparación de precios de dispositivos OpenFlow [47], [48], [49]

Marca	Modelo	Puertos	Interfaz WiFi	Precio
Switch OpenFlow implementado	*	4	No	$ 140,00
AP OpenFlow implementado	*	3	Si	$ 145,00
Switch Pica8	P-3297 48-PORT 1GBE SWITCH	48	No	$ 3.960,00
Switch EDGE	AS4600	48	No	$ 3.000,00
Switch NEC	PF5240	48	No	$ 1.996,00

Luego de observar la comparación de precios, se puede destacar que los dispositivos OpenFlow desarrollados representan una buena opción para estudiantes o investigadores que requieran experimentar con las SDN.

A continuación en la Tabla 3.12, se presenta de manera resumida los costos referenciales de todos los elementos del prototipo a implementar en el presente trabajo.

Los computadores portátiles o laptops únicamente representan estaciones utilizadas para realizar pruebas en el prototipo, sin embargo pueden utilizarse otros equipos, de distinto precio, con el requisito de que posean conectividad de red cableada e inalámbrica.

En cuanto a los accesorios de cada placa Raspberry Pi, estos comprenden: case, disipadores, fuente de poder y tarjeta microSD clase 10.

Finalmente es importante señalar que el monitor LCD, mouse USB y teclado USB tienen únicamente la función de facilitar la configuración de la Raspberry Pi para que actúe como controlador.

Tabla 3.12. Costos del prototipo a implementar

Equipo	Precio
2 Raspberry Pi Modelo B+ (incluidos accesorios)	$ 100,00
Raspberry Pi 2 Modelo B (incluidos accesorios)	$ 80,00
Raspberry Pi 3 Modelo B (incluidos accesorios)	$ 100,00
5 adaptadores USB - LAN Ethernet Realtek	$ 60,00
2 adaptadores USB - WiFi DLink	$ 22,00
7 Patchcords Categoría 5e	$ 75,00
Mouse USB XTECH	$ 5,00
Teclado USB XTECH	$ 10,00
Monitor LCD LG	$ 30,00
Switch convencional 5 puertos NEXXT	$ 15,00
Laptop Toshiba Satellite	$ 500,00
2 Laptop Compaq 515	$ 700,00
TOTAL	**$1.697,00**

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y recomendaciones obtenidas luego de la realización del presente trabajo.

4.1 CONCLUSIONES

- En el presente trabajo se implementó un prototipo de SDN, el cual se compone de varios elementos básicos de una SDN, como son: *switch*, *access points* y controlador, los cuales fueron implementados utilizando herramientas open-source, tales como placas de bajo costo Raspberry Pi y distribuciones de software libre basadas en LINUX Raspbian OS y OpenWrt. El prototipo de SDN implementado es compatible con dispositivos inalámbricos (*access points*), es decir brinda cierto grado de movilidad a las estaciones. Para implementar la compatibilidad del protocolo OpenFlow, se utilizó el *switch* basado en software Open vSwitch y el controlador basado en software Ryu. Además, se utilizó la herramienta de emulación para SDN Mininet-WiFi, obteniendo así resultados de *throughput* y *jitter* para luego compararlos con los resultados alcanzados con el prototipo real. Finalmente, en el controlador se implementó una aplicación de firewall sobre la SDN y se realizaron pruebas de su funcionamiento.
- Se consiguió una solución de bajo costo que comparado con otras soluciones es una solución bastante económica. Se demostró que se pueden obtener dispositivos SDN compatibles con OpenFlow, tales como *switch,* controlador y *access point*, mismos que pueden ser usados para experimentación a nivel de laboratorio, en una red cableada y con conectividad WiFi, para así tratar de fomentar el uso de esta tecnología de red.
- Se comprobó que el *switch* virtual basado en software Open vSwitch es compatible con las placas Raspberry Pi tanto para implementar un *switch* como un *access point* gracias a la utilización de dos distribuciones basadas en LINUX compatibles con Open vSwitch las cuales son Raspbian OS y OpenWrt.
- El controlador Ryu brinda la posibilidad de desarrollar una infinidad de aplicaciones que pueden ser usadas en una SDN, el desarrollador debe tener al

menos un conocimiento básico del lenguaje de programación Python, sin embargo la documentación disponible en Internet sobre las clases para poder utilizar las distintas funcionalidades del protocolo OpenFlow son explicadas de manera general y sin profundizar en demasía, por lo cual la comunidad de desarrolladores de Ryu es de gran importancia ya que brinda ayuda a los interesados en empezar a desarrollar aplicaciones utilizando dicho controlador.

- La herramienta de emulación de SDN Mininet es utilizada para realizar pruebas de aplicaciones sobre SDN, antes que estas se implementen con dispositivos reales, sin embargo dicha herramienta posee la limitación de emular dispositivos inalámbricos, afortunadamente existe una extensión de la herramienta llamada Mininet-WiFi la cual permite emular dichos dispositivos brindando la posibilidad de añadir a las SDN compatibilidad con redes inalámbricas. Esta herramienta fue utilizada para emular previamente el prototipo de la SDN que se implementó.
- El prototipo fue desarrollado con el fin de que se utilice para experimentación de SDN basadas en el protocolo OpenFlow a nivel de laboratorio o para aplicaciones en redes cableadas y con conectividad WiFi, por lo tanto se debe tener en cuenta que el rendimiento del mismo jamás será el de un prototipo diseñado y construido para trabajar en producción con el protocolo OpenFlow, ya que en el mercado existen opciones comerciales fabricadas por grandes marcas que presentan mejores prestaciones que los dispositivos implementados de este trabajo.
- Se utilizó el software Wireshark para capturar y verificar el intercambio de mensajes OpenFlow entre los dispositivos implementados, es decir tanto entre el *switch* y el controlador SDN, como entre el AP y el controlador SDN.
- Se compararon los resultados de Mininet-WiFi y del prototipo real, en *throughput* y *jitter*, a partir de generar tráfico con el software iperf. Los resultados mostraron que en el prototipo real se presentan variaciones en *throughput* y *jitter* propias de una red donde intervienen nodos inalámbricos, en tanto que en la simulación los resultados fueron bastantes cercanos a los ideales, es decir no se presentaron variaciones en *throughput;* sin embargo en cuanto al *jitter* si se presentaron variaciones dado que es una métrica que

es independiente de que tan constante sea la tasa de datos en una red, puesto que tan solo mide la variación de la latencia.

- La aplicación de f*irewall* para el controlador fue correctamente implementada en el prototipo y su comportamiento fue comprobado a través de pruebas del bloqueo en el acceso a varios servidores implementados, esto fue realizado mediante la inserción de reglas las cuales fueron traducidas en entradas de las tablas de flujos OpenFlow en el *switch* y en los *access points*. Las pruebas que se realizaron fueron el bloqueo de todo el tráfico de la red por defecto y el bloqueo del tráfico por IP y puerto en los servidores HTTP, FTP y SSH, que se implementaron.

4.2 RECOMENDACIONES

- La tarjeta SD utilizada como memoria de las Raspberry Pi se recomienda que sea de clase 10, que al momento de la elaboración del presente trabajo es la que brinda mayor velocidad de lectura y escritura, esto con la finalidad de optimizar al máximo todos los recursos.
- Pese a que existen varias opciones de controladores OpenFlow disponibles, se recomienda el uso de Ryu ya que éste utiliza el lenguaje de programación Python, que es el lenguaje de programación utilizado por la mayoría de aplicaciones desarrolladas para la Raspberry Pi.
- Se recomienda la verificación del soporte de adaptadores USB-WiFi y USB-Ethernet en las distribuciones LINUX utilizadas, ya que para el correcto funcionamiento de las mismas se deben seleccionar e instalar los drivers manualmente, ya que en ocasiones los drivers no se encuentran instalados por defecto.
- Se recomienda que se incentive a desarrollar nuevas e innovadoras aplicaciones propias de las SDN que permitan tener redes más inteligentes que sean más fáciles de configurar y administrar y que además brinden funciones que no hayan sido probadas anteriormente.
- Se recomienda revisar toda la bibliografía disponible sobre las SDN de tal forma que se tenga una idea clara de lo que son, de cómo funcionan, y de cómo se puede empezar a desarrollar aplicaciones para las mismas. Es importante dejar en claro que aun cuando las SDN ofrecen una administración

más fácil en redes de gran tamaño, esto no quiere decir que un administrador de red ya no sea necesario, de hecho se necesitará de personas bien preparadas en diversos lenguajes de programación y en redes de comunicaciones para que las empresas puedan beneficiarse al máximo de estas tecnologías.

REFERENCIAS BIBLIOGRÁFICAS

[1] ONF, «ONF White Paper Software-Defined Networking: The New Norm For Networks,» 13 Abril 2012. [En línea]. Disponible: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. [Último acceso: 10 Septiembre 2016].

[2] The Tech Corner, «The Tech: SDN Architecture,» 25 Diciembre 2012. [En línea]. Disponible: http://www.thetech.in/2012/12/sdn-architecture.html. [Último acceso: 1 Diciembre 2016].

[3] Ryu SDN Framework Community, «Ryu SDN Framework,» 3 Junio 2013. [En línea]. Disponible: https://osrg.github.io/ryu/. [Último acceso: 15 Septiembre 2016].

[4] Project Floodlight, «Project Floodlight,» 2 Enero 2015. [En línea]. Disponible: http://www.projectfloodlight.org/. [Último acceso: 15 Septiembre 2016].

[5] LINUX FOUNDATION, «The OpenDaylight Platform | OpenDaylight,» 13 Abril 2016. [En línea]. Disponible: https://www.opendaylight.org/. [Último acceso: 15 Septiembre 2016].

[6] ONOS, «ONOS,» 4 Diciembre 2015. [En línea]. Disponible: http://onosproject.org/. [Último acceso: 15 Septiembre 2016].

[7] Open Networking Foundation, «SDN Architecture Overview,» 12 Diciembre 2013. [En línea]. Disponible: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf. [Último acceso: 1 Diciembre 2016].

[8] Internet Engineering Task Fork IETF, «Software-Defined Networking (SDN): Layers and Architecture Terminology,» 7 Enero 2015. [En línea]. Disponible: https://tools.ietf.org/html/rfc7426. [Último acceso: 15 Noviembre 2016].

[9] E. Haleplidis, J. H. Salim y K. Pentikousis, «Techniques and Tools for the Management and Operation of NFV and SDN Networks,» 22 Julio 2016. [En

línea]. Disponible: https://www.ietf.org/proceedings/96/slides/slides-96-sdnrg-8.pdf. [Último acceso: 15 Noviembre 2016].

[10] P. Goransson y C. Black, Software Defined Networks - A Comprehensive Approach, Elsevier, 2014.

[11] M. Singh y G. Singh, «The Exposition of Enigma Software Defined Networking».

[12] LINUX Foundation, «LFS265 Software Defined Networking Fundamentals,» Online, 2016.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker y J. Turner, «OpenFlow: Enabling Innovation in Campus Networks,» *SIGCOMM Comput. Commun. Rev.,* vol. 38, pp. 69-74, #mar# 2008.

[14] CNMC blog, «Openflow, un nuevo código de circulación para Internet,» 22 Junio 2012. [En línea]. Disponible: https://blog.cnmc.es/2012/06/22/un-nuevo-codigo-de-circulacion-para-internet/. [Último acceso: 2 Octubre 2016].

[15] The OpenFlow Switch Consortium, « OpenFlow Switch Specification Version 1.0.0,» 2009.

[16] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky y S. Uhlig, «Software-defined networking: A comprehensive survey,» *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[17] The OpenFlow Switch Consortium, « OpenFlow Switch Specification Version 1.1.0,» 2011.

[18] The OpenFlow Switch Consortium, «OpenFlow Switch Specification Version 1.2.0,» 2011.

[19] The OpenFlow Switch Consortium, «OpenFlow Switch Specification Version 1.3.0,» 2012.

[20] The OpenFlow Switch Consortium, «OpenFlow Switch Specification Version 1.4.0,» 2013.

[21] The OpenFlow Switch Consortium, «OpenFlow Switch Specification Version 1.5.0,» 2014.

[22] RASPBERRY PI FOUNDATION, «Raspberry Pi Foundation - About Us,» 17 Mayo 2016. [En línea]. Disponible: https://www.raspberrypi.org/about/. [Último acceso: 30 Septiembre 2016].

[23] Core Electronics, «Raspberry Pi Boards Compared,» 8 Mayo 2016. [En línea]. Disponible: http://core-electronics.com.au/tutorials/compare-raspberry-pi-boards.html. [Último acceso: 30 Septiembre 2016].

[24] Element14, «Raspberry Pi 3 Model B Technical Specifications,» 22 Febrero 2016. [En línea]. Disponible: https://www.element14.com/community/docs/DOC-80899/l/raspberry-pi-3-model-b-technical-specifications. [Último acceso: Octubre 5 2016].

[25] LINUX Foundation, «Open vSwitch,» 30 Julio 2009. [En línea]. Disponible: http://openvswitch.org/. [Último acceso: 5 Octubre 2016].

[26] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon y M. Casado, «The Design and Implementation of Open vSwitch,» de *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, 2015.

[27] RYU project team, «RYU SDN Framework - Ryubook 1.0 documentation,» 12 Agosto 2015. [En línea]. Disponible: https://osrg.github.io/ryu-book/en/html/preface.html. [Último acceso: 10 Octubre 2016].

[28] Nippon Telegraph and Telephone Corporation, «Components of Ryu — Ryu 4.8 documentation,» 24 Abril 2013. [En línea]. Disponible: http://ryu.readthedocs.io/en/latest/components.html. [Último acceso: 10 Octubre 2016].

[29] S. Rao, «SDN Series Part Four: Ryu, a Rich-Featured Open Source SDN Controller Supported by NTT Labs,» 23 diciembre 2014. [En línea]. Disponible: https://thenewstack.io/sdn-series-part-iv-ryu-a-rich-featured-open-source-sdn-controller-supported-by-ntt-labs/. [Último acceso: 25 diciembre 2016].

[30] Mininet Team, «Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet,» 21 Diciembre 2012. [En línea]. Disponible: http://mininet.org/. [Último acceso: 15 Octubre 2016].

[31] ramonfontes.com, «Mininet-WiFi – ramonfontes.com,» 8 Agosto 2015. [En línea]. Disponible: http://www.ramonfontes.com/mininet-wifi/. [Último acceso: 11 Octubre 2016].

[32] R. Fontes y C. Rothenberg, «Mininet-WiFi The User Manual Draft Release,» Diciembre 2016. [En línea]. Disponible: https://www.dropbox.com/s/jgkhsdrz736e4q2/mininet-wifi-draft-manual.pdf?dl=0. [Último acceso: 5 Diciembre 2016].

[33] R. R. Fontes y P. N. M. Sampaio, «Visual Network Description: A Customizable GUI for the Creation of Software Defined Network Simulations».

[34] S. Hätönen, «SOFTWARE-DEFINED WI-FI NETWORKS WITH WIRELESS ISOLATION,» 22 agosto 2016. [En línea]. Disponible: https://wiki.helsinki.fi/display/WiFiSDN/Software-Defined+Wi-Fi+Networks+with+Wireless+Isolation. [Último acceso: 10 enero 2017].

[35] OpenWrt - Wireless Freedom, «Downloads - Snapshots,» 20 junio 2015. [En línea]. Disponible: http://downloads.openwrt.org/snapshots/trunk/brcm2708/generic/openwrt-brcm2708 -bcm2708-rpi-ext4-sdcard.img.gz . [Último acceso: 5 diciembre 2016].

[36] Sourceforge, «Download software Win 32 disk imager,» 28 marzo 2016. [En línea]. Disponible: https://sourceforge.net/projects/win32diskimager/. [Último acceso: 10 diciembre 2016].

[37] S. Tatham, «PuTTY - Downloads,» 15 noviembre 2016. [En línea]. Disponible: http://www.chiark.greenend.org.uk/~sgtatham/putty/.. [Último acceso: 15 diciembre 2016].

[38] Flowgrammable, «Message Layer - SDN / OpenFlow / Message Layer | Flowgrammable,» 2 enero 2014. [En línea]. Disponible: http://flowgrammable.org/sdn/openflow/message-layer/#tab_ofp_1_3. [Último acceso: 15 enero 2017].

[39] Wireshark Foundation, «Wireshark,» 17 agosto 2016. [En línea]. Disponible: https://www.wireshark.org. [Último acceso: 18 enero 2017].

[40] Flowgrammable, «SDN / OpenFlow / Message Layer / Hello | Flowgrammable,» 2 enero 2014. [En línea]. Disponible: http://flowgrammable.org/sdn/openflow/message-layer/hello/#Hello_1.3. [Último acceso: 20 enero 2017].

[41] Flowgrammable, «SDN / OpenFlow / Message Layer / FeatureReq-Res - Flowgrammable,» 2 enero 2014. [En línea]. Disponible: http://flowgrammable.org/sdn/openflow/message-layer/feature/#tab_ofp_1_3. [Último acceso: 20 enero 2017].

[42] Flowgrammable, «SDN / OpenFlow / Message Layer / StatsRequest | Flowgrammable,» 2 enero 2014. [En línea]. Disponible: http://flowgrammable.org/sdn/openflow/message-layer/statsrequest/#ofp_1_3. [Último acceso: 21 enero 2017].

[43] Flowgrammable, «SDN / OpenFlow / Message Layer / FlowMod | Flowgrammable,» 2 enero 2014. [En línea]. Disponible: http://flowgrammable.org/sdn/openflow/message-layer/flowmod/#FlowMod_1.3. [Último acceso: 21 enero 2017].

[44] iPerf, «iPerf - The TCP, UDP and SCTP network bandwidth measurement tool,» 19 septiembre 2016. [En línea]. Disponible: https://iperf.fr. [Último acceso: 28 enero 2017].

[45] R. Sukapuram y G. Barua, «PPCU: Proportional Per-packet Consistent Updates for Software Defined Networks - A Technical Report,» *CoRR,* vol. abs/1609.00126, 2016.

[46] K. Zhao, Q. Li y Y. Jiang, «Flow-level consistent update in SDN based on K-prefix covering,» de *Global Communications Conference (GLOBECOM), 2014 IEEE*, 2014.

[47] myriad SUPPLY, «Edge-Core AS4600-54T,» 8 marzo 2015. [En línea]. Disponible: http://www.myriadsupply.com/product/edge-core-as4600-54t/. [Último acceso: 1 febrero 2017].

[48] Netgate, «Pica8 P-3297 TCAM 48 x 1G OpenFlow Switch,» 16 febrero 2014. [En línea]. Disponible: https://store.netgate.com/Pica8/P-3297.aspx. [Último acceso: 3 febrero 2017].

[49] PROVANTAGE, «1-Year Upplt Warranty PF5240-48T4XW,» 26 marzo 2016. [En línea]. Disponible: https://www.provantage.com/nec-q24dn000000007952~1350008547.htm. [Último acceso: 3 febrero 2017].

[50] Ryu SDN Framework Community, «Overview/What's Ryu the Network Operating System,» 12 agosto 2013. [En línea]. Disponible: https://github.com/osrg/ryu/blob/master/doc/source/parameters.rst. [Último acceso: 20 diciembre 2016].

ANEXOS

ANEXO A: Código fuente para la simulación del prototipo.

ANEXO B: Procedimiento de instalación de la distribución Raspbian en Raspberry Pi.

ANEXO C: Código fuente para la configuración de Open vSwitch en Raspbian.

ANEXO D: Código fuente para la configuración de Open vSwitch en OpenWrt.

ANEXO E: Código fuente de la aplicación de *firewall*.

ANEXO F: Código fuente de la interfaz gráfica de usuario de un cliente para la aplicación de *firewall*.

ANEXO G: Valores de *throughput*, *jitter*, uso de CPU y consumo de RAM.

Nota: *Los anexos se encuentran en el CD adjunto.*