PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

CONSIDERACIONES DE DISEÑO DE UN WIRELESS CONTROLLER CON CÁLCULO DE RUTAS BASADO EN SDN

Tesis para obtener el título profesional de Ingeniero Electrónico

AUTOR:

Julio Fernando Mendoza Huaman

ASESOR:

Gumercindo Bartra Gardini

Lima, Noviembre, 2023

Informe de Similitud

Yo, GUMERCINDO BARTRA GARDINI, docente de la Facultad de CIENCIAS E INGENIERÍA de la Pontificia

Universidad Católica del Perú, asesor de la tesis titulada:

CONSIDERACIONES DE DISEÑO DE UN WIRELESS CONTROLLER CON CÁLCULO DE RUTAS BASADO EN SDN, del autor: JULIO FERNANDO MENDOZA HUAMÁN,

dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 15.%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 02/10/2023.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha: 27 de Octubre de 2023……

Apellidos y nombres del asesor: BARTRA GARDINI, GUMERCINDO	
DNI: 07231977	Firma 
ORCID: https://orcid.org/0000-0002-4307-1285	

Resumen

En el presente trabajo de investigación se presentan las consideraciones de diseño de un Wireless Controller con cálculo de rutas basado en SDN (Software Defined Networking por sus siglas en inglés), un nuevo paradigma de redes que busca centralizar el control de una red y realizar cambios en ella en tiempo real mediante software, separando los planos de control y datos a comparación de las redes tradicionales; ello con miras a una posible implementación en la arquitectura de red, así como en el hardware local de la Pontificia Universidad Católica del Perú (PUCP). Para su desarrollo se ha tenido un enfoque deductivo, entendiendo primero la problemática de la red inalámbrica de la PUCP; luego investigado sobre las redes inalámbricas, algoritmos y protocolos de enrutamiento dinámico junto con el funcionamiento de los Wireless LAN Controllers (WLC); también sobre el paradigma SDN en redes inalámbricas, sobre los parámetros a considerar en el diseño como throughput, latencia, tolerancia a fallas, número de equipos de red conectados y seguridad; así como los entornos de simulación para su implementación, escenarios de prueba y el modelo de solución del mismo. Este proyecto se realiza con el apoyo del laboratorio del Grupo de Investigación de Redes Avanzadas (GIRA) de la PUCP, el cual ha realizado investigaciones anteriores en SDN, y cuenta con recursos de software y hardware que han permitido el desarrollo y pruebas del Wireless Controller a diseñar en la presente tesis, y cuyos resultados han sido validados mediante un gabinete de pruebas para la simulación de tráfico para una red SDN. Dicha contrastación comprobó la escalabilidad, programabilidad y versatilidad de este nuevo paradigma de redes y su beneficio de utilizarse en una red de campus académica como la de la PUCP y sirve como insumo para futuras investigaciones respecto al paradigma de SDN en redes de campus académicas.

Palabras clave: *SDN, Wireless, WLC*

Dedicado a mis padres, hermanos,
abuelos, familiares, profesores y amigos
quienes me han acompañado durante
todos estos años en la universidad.

”To infinity and beyond!”

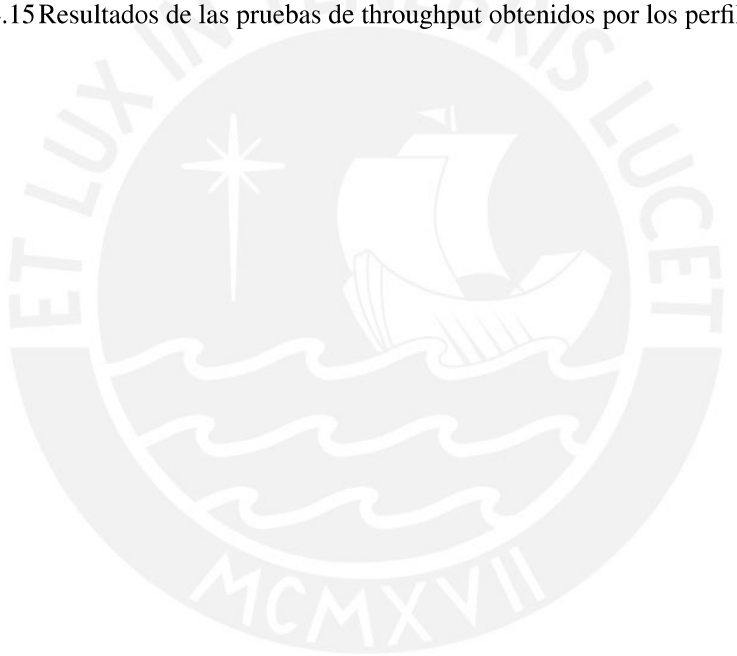
Buzz Lightyear

”Yo puedo vivir del amor..”

Rubén Blades

Índice General

Índice de Figuras



Índice de Tablas

Introducción

Las redes definidas por software (SDN por sus siglas en inglés) son un nuevo paradigma de networking que busca centralizar el control de una red mediante software, logrando variaciones en tiempo real en contraste a las redes tradicionales que son menos flexibles y requieren mayores costes de replicación dado que dependen netamente del hardware existente; y que en el caso de las redes inalámbricas convierte a los Wireless Controllers en cuellos de botella para el tráfico de los dispositivos conectados, afectando negativamente la calidad de servicio a los usuarios.

El presente trabajo consiste en el diseño de un Wireless Controller con cálculo de rutas basado en SDN con miras a una posible implementación en la arquitectura de red, así como en el hardware local de la Pontificia Universidad Católica del Perú (PUCP).

En el primer capítulo, se describe el marco problemático, donde se realiza una comparación entre las redes inalámbricas tradicionales y las redes definidas por software, así como un análisis del crecimiento del tráfico en la PUCP que justifican el desarrollo del presente proyecto. Luego, se plantean los objetivos y alcances del presente trabajo de investigación.

En el segundo capítulo se exponen los fundamentos teóricos para entender el diseño del Wireless Controller. Se explican las redes inalámbricas, sus componentes, diferentes topologías, algoritmos y protocolos de enrutamiento dinámico. Además se detalla el paradigma SDN aplicado a redes inalámbricas, sus componentes, lenguajes de programación utilizados y parámetros importantes para su diseño. En adición, se describen los diferentes ambientes de simulación utilizados para las pruebas de funcionamiento de controladores SDN que servirán para la implementación de la presente tesis.

En el tercer capítulo, se presentan las consideraciones de diseño del Wireless Controller con cálculo de rutas basado en SDN, en el cual se definen requerimientos, metodología, topología, así como las pruebas para determinar los parámetros de red de throughput y latencia que servirán para la validación del controlador.

En el cuarto capítulo, se presentan las pruebas realizadas de generación de tráfico, configuración de los entornos de simulación y los perfiladores que permitirán la medición del

throughput y latencia de la red tanto en redes convencionales como en redes SDN. En adición a ello se presentan los resultados de las pruebas y la respectiva validación del Wireless Controller.

La última parte del trabajo de tesis presenta las conclusiones, las mismas que corresponden a los objetivos del presente trabajo y que fueron logrados satisfactoriamente; así también, las recomendaciones y trabajos futuros que permitirán realizar mejoras al presente trabajo y próximas investigaciones respecto al paradigma de SDN en redes de campus académicas. Asimismo, las referencias bibliográficas respaldan las informaciones empleadas en este documento.

Capítulo 1

Marco problemático

El presente capítulo describe la delimitación del problema, los objetivos, alcances, justificación y la estructura de la presente tesis.

1.1. Delimitación del problema

Durante muchos años la manera en que se han llevado a cabo las implementaciones de redes han dependido del hardware existente, puesto que este era el encargado del cálculo de rutas entre los diferentes elementos de una red. Conforme va aumentando la cantidad de usuarios conectados, estos dispositivos gestores de redes terminan convirtiéndose en potenciales cuellos de botella lo que afecta la calidad de servicio de internet de los dispositivos conectados. Además, estos dispositivos de gestión de red son muy sofisticados y tienen un alto costo en el mercado lo que limita la posibilidad de adquirir nuevos equipos para aliviar el tráfico[1]. Otro punto importante es que las configuraciones de estos toman mucho tiempo y requieren de personal altamente capacitado, llegando a que las soluciones a problemas en la red se den de una manera poco dinámica, afectando negativamente la performance de esta.

En el caso específico de las redes inalámbricas, estas en su estructura requieren de access points (AP) que son administrados por un Wireless LAN Controller (WLC). Un usuario debe conectarse a un AP y este mediante un túnel se conecta al WLC , que es el que se encarga del cálculo de rutas entre los elementos de la red y concentra todas las conexiones de los usuarios de móviles[2]. Al ser concentradores estos representan un potencial cuello de botella, con una replicación costosa y con configuraciones poco automatizables para solución de problemas dentro de esta.

1.1.1. Contexto de la PUCP

Aterrizando al contexto de la Pontificia Universidad Católica del Perú (PUCP), esta cuenta con una topología de red inalámbrica en la cual cada uno de los usuarios inalámbricos se conecta a cada uno de los puntos de acceso distribuidos en el campus PUCP. Estos a su vez se conectan a los switches, los cuales redireccionan todo este tráfico al WLC modelo 3700 de la marca Cisco, ubicado en el edificio de Dirección de Tecnologías de la Información (DTI) de la PUCP, el cual se encarga de gestionar todo el tráfico proveniente de los usuarios conectados a la red inalámbrica. A continuación, en la Figura 1.1 se muestra el esquema de la topología de red inalámbrica del Campus PUCP.

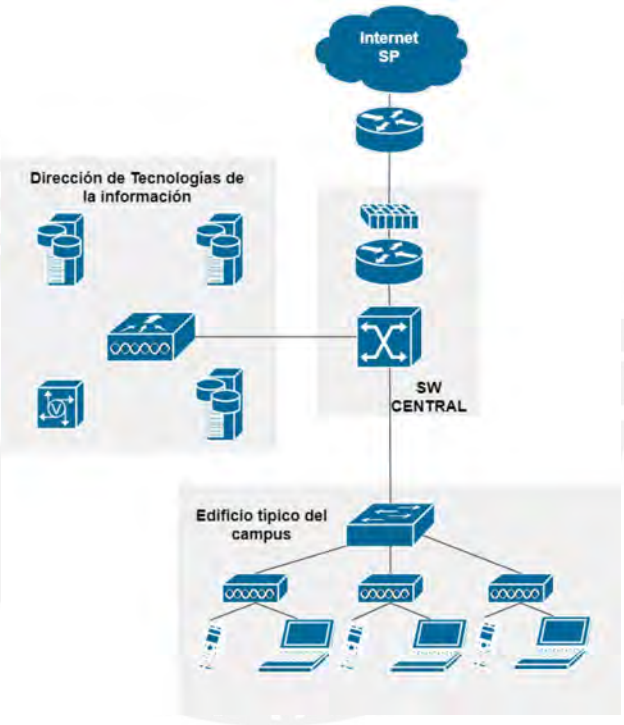


Figura 1.1: Topología de la red inalámbrica del Campus PUCP [3]

En adición a lo anteriormente mencionado, de acuerdo con los datos administrativos del Portal de Transparencia de la PUCP , hasta diciembre del 2019 se contaba con 1189 puntos de acceso inalámbrico dentro del Campus de Pando [4], los cuales están distribuidos en los diferentes espacios interiores como aulas, laboratorios y bibliotecas, así como en espacios exteriores como comedores, jardines y el Tontódromo. La distribución de estos puntos de acceso se muestra a continuación en la Figura 1.2, en la cual se aprecia que el acceso a Internet de manera inalámbrica está asegurado en gran parte del Campus Pando de la PUCP.

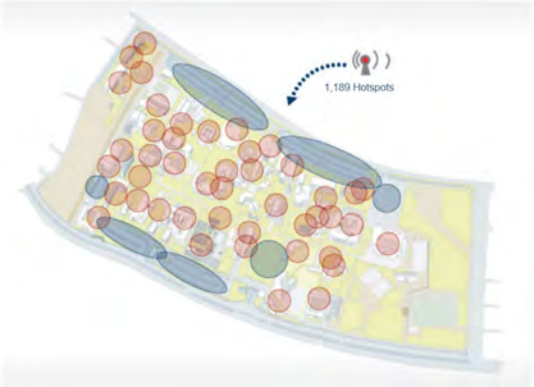


Figura 1.2: Distribución de los puntos de acceso inalámbricos en el Campus PUCP [4]

Así también, en la PUCP, en un contexto pre pandemia, se tuvo un crecimiento del tráfico hacia Internet, el cual pasó de 900 Mbps en el 2015 hasta casi 3 Gbps en el 2019 [5]. Esto se debe principalmente a la cantidad de usuarios y dispositivos conectados a la red de la universidad, siendo en su mayoría conectados a través de la red Wireless, siendo crítico establecer nuevas formas de desarrollar nuevos Wireless Controllers con otras tecnologías menos costosas y de fácil replicación.

Tabla 1.1: Crecimiento del tráfico en la PUCP [5]

Año	Capacidad
1er Año (2015)	900 Mbps
2do Año (2016)	1215 Mbps
3er Año (2017)	1640.25 Mbps
4to Año (2018)	2214.34 Mbps
5to Año (2019)	2989.36 Mbps

Frente a esta problemática se sugiere establecer una solución simplificada y robusta como es el caso de las tecnologías basadas en Redes definidas por Software (SDN por sus siglas en inglés). Por ello, el presente trabajo de tesis que se engloba en este nuevo paradigma de redes permitirá lograr la separación de los planos de control y datos en la gestión de una red Wireless con un Wireless Controller para cálculo de rutas, facilitando la administración y mantenimiento de esta, ya que como es definida por software es automatizable y permite variaciones en tiempo real de las diferentes políticas aplicadas en esta, y al ser de código abierto, requiere menos costes de replicación[6].

1.2. Objetivos de la tesis

1.2.1. Objetivo General

- Diseñar un Wireless Controller basado en SDN para cálculo de rutas

1.2.2. Objetivos Específicos

- Entender cómo funcionan los Wireless Controller convencionales
  - Investigar sobre los algoritmos de enrutamiento dinámico utilizados en redes inalámbricas
- Comprender el paradigma de SDN aplicado a redes Wireless
  - Investigar sobre los protocolos y funcionalidades de SDN en redes inalámbricas
- Definir los criterios de diseño del Wireless Controller basado en SDN y parámetros de mejora
  - Definir escenarios de prueba del Wireless Controller a diseñar
- Implementar el Wireless Controller basado en SDN con un emulador de Redes
  - Realizar el cálculo de rutas usando servidores HPC (High Performance Computing) de la marca Hitachi
- Validar la implementación del Wireless Controller basado en SDN
  - Diseñar un gabinete de pruebas para la simulación de tráfico para una red SDN

1.3. Alcance del proyecto de tesis

El presente proyecto de tesis plantea diseñar un Wireless Controller basado en SDN para el cálculo de rutas el cual será validado en laboratorio en conjunto con el controlador PUCPLight desarrollado por el Grupo de Investigación de Redes Avanzadas (GIRA) de la PUCP [5], el cual se basa en FloodLight, y tomará como referencia el tráfico simulado por un emulador de redes, ya sea Virtual Network Research Testbed (VNRT) o MiniNet, los switches SDN/OpenFlow físicos marca Pica 8 y los Access Points de marca TP-Link con software OVS (Linux Embebido OpenWRT), los cuales son equipos brindados por el GIRA. Así también, para el cálculo de rutas se utilizarán servidores HPC de la marca Hitachi los cuales serán facilitados por la PUCP.

1.4. Justificación del proyecto de tesis

El presente proyecto de tesis nace de la necesidad de aplicar nuevas tecnologías, como es el caso de SDN, a problemas reales y de impacto en la sociedad como lo es la problemática presentada en el primer ítem de esta sección y que puede solucionarse con este nuevo paradigma de redes el cual permite soluciones con control centralizado de la red mediante software con variaciones en tiempo real de las políticas de la red; así como menores costes de replicación dado que son de código abierto y que permiten ser mejorados y escalados para las diferentes aplicaciones que pudieran surgir y que sean relacionadas con gestión de redes inalámbricas mediante software, ello en comparación a las redes inalámbricas convencionales.

Así también, este trabajo busca ser un aporte a las diferentes investigaciones realizadas por el GIRA en el contexto del desarrollo de SDN con miras a una posible implementación del Wireless Controller con cálculo de rutas diseñado en la presente tesis en la arquitectura de red y en el hardware local de la PUCP.

1.5. Estructura de la tesis

La presente tesis está compuesta por 4 capítulos que a continuación se presentan:

- En el capítulo 1, se presenta la problemática, objetivos, alcance y justificación de la presente tesis.
- En el capítulo 2, se documentan los conceptos que se utilizarán para las consideraciones de diseño de un Wireless Controller con cálculo de rutas basado en SDN tales como algoritmos de enrutamiento en redes inalámbricas convencionales y el paradigma SDN.
- En el capítulo 3, el diseño del Wireless Controller con cálculo de rutas basado en SDN.
- En el capítulo 4, las pruebas de concepto del Wireless Controller con cálculo de rutas basado en SDN, su validación de funcionamiento y la conclusión de la tesis.

Capítulo 2

Fundamentos teóricos

A continuación se presentan los conceptos más importantes para el diseño del Wireless Controller con cálculo de rutas basado en SDN. Estos conceptos se explicarán deductivamente, comenzando desde los conceptos de redes inalámbricas hacia el paradigma de SDN, SDN en redes inalámbricas explicando consideraciones para la implementación del Wireless Controller y ambientes de simulación de controladores SDN.

2.1. Redes inalámbricas

Para llegar a entender las consideraciones de diseño necesarias en un Wireless Controller es necesario primero ahondar en cómo funcionan las redes inalámbricas. El medio por el cual se comunican los componentes de esta red es el aire, a través de ondas electromagnéticas en lo que se conoce como estándar IEEE 802.11 o comúnmente llamado *WiFi* y cuyo uso se ha extendido con el desarrollo de los dispositivos móviles como celulares o laptops. Al conectarse los dispositivos de este tipo se agrupan en redes denominadas *Wireless Local Area Network* (WLAN) tal como se tiene en viviendas y universidades [7].

2.1.1. Componentes de las redes inalámbricas

Dentro de los componentes de una WLAN, y así como se aprecia en la Figura 2.1 se tienen los siguientes:

- **Wireless LAN Controller (WLC):** Dispositivo de control y gestión de la red inalámbrica que cumple una función similar a la de los routers de las redes cableadas pero con el tráfico de la red inalámbrica. Se detallará más de este en una próxima sección del presente capítulo.

- **Access point (AP):** Elemento repetidor de la señal de *WiFi*, al cual se conectan los endpoints de la red y que está conectado al WLC de manera cableada mediante switches para que los usuarios puedan acceder a Internet. Existen modelos de APs que pueden cumplir la función de WLCs para redes pequeñas.
- **Endpoint:** Dispositivo final el cual se conecta a los APs y genera tráfico en la red inalámbrica.

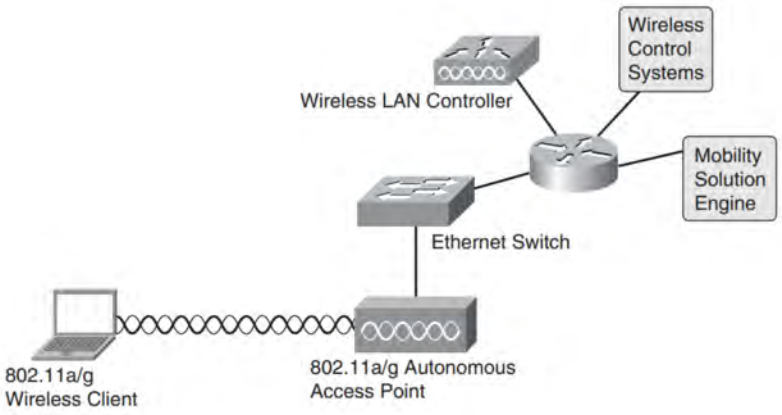


Figura 2.1: Elementos de una WLAN controlada [2]

Además de los componentes anteriormente comentados, también existen casos de redes inalámbricas que no cuentan con elementos como WLCs o APs y en la cual cada *Endpoint* se conecta entre sí con los otros elementos de la red. Este tipo particular de red se conoce como redes *ad hoc*[7].

2.1.2. Tipos de topologías de las redes inalámbricas

Los elementos comentados en la sección anterior pueden ser distribuidos en diferentes tipos de topologías dependiendo el caso de uso.

2.1.2.1. MANET

La topología MANET, es un tipo de red *ad hoc* móvil en la cual los dispositivos son movibles y autoconfigurables, y que además cambian dinámicamente sus enlaces, lo que representa el mayor reto para esta topología dado que se debe mantener la información actualizada en cada uno de los nodos en movimiento. Debido a este particularidad permite que sean desplegadas en momentos de desastre, así como en situaciones que requieran dispositivos conectados en movimiento[8].

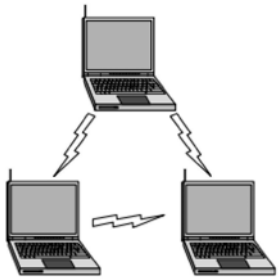


Figura 2.2: Red *ad hoc* [7]

2.1.2.2. Mesh

La topología Mesh, es un tipo de red que combina elementos *ad hoc* y de las redes inalámbricas convencionales. Así pues, elementos de este tipo de red pueden actuar como APs o como intermediarios con otro tipo de redes similares, creando diferentes nodos (tal como se aprecia en la Figura 2.3) y que tienen como principal tarea el correcto funcionamiento de la red el enrutamiento de paquetes [8].

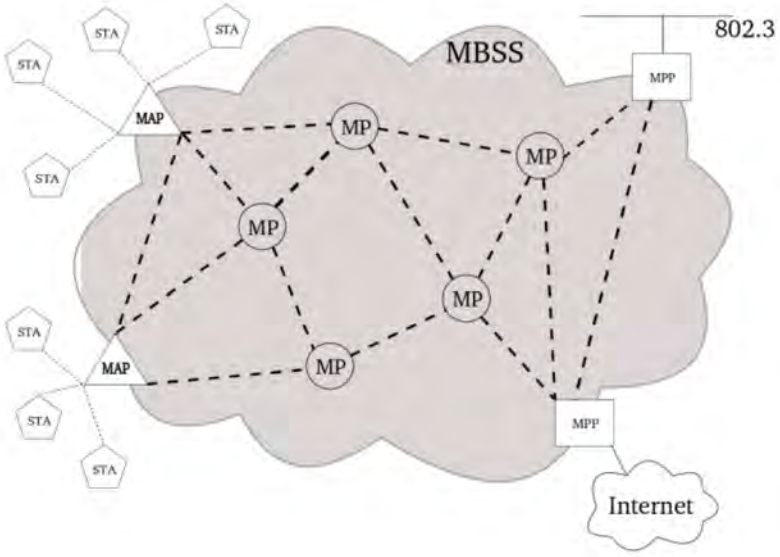


Figura 2.3: Red mesh [8]

2.1.2.3. Infraestructura

La topología del modo infraestructura es como la que se muestra en la Figura 2.1, en la cual los Endpoints se conectan directamente a los APs y estos a su vez a switches y elementos de gestión como los WLC. Así pues, a diferencia de las redes MANET y Mesh, los dispositivos finales se conectan entre ellos a través de los APs y no directamente entre ellos como el caso de MANET y

Mesh.

2.1.3. Algoritmos de enrutamiento dinámico utilizados en redes inalámbricas

Como una labor crucial dentro de las redes inalámbricas es el enrutamiento de paquetes, es necesario entender los algoritmos utilizados para realizar un eficiente enrutamiento dinámico de paquetes en una red.

2.1.3.1. Algoritmo de Dijkstra

Uno de los algoritmos más utilizados es el algoritmo de Dijkstra, el cual fue desarrollado a mediados del siglo pasado por el profesor Edsger Dijkstra el cual permite solucionar el problema de hallar la ruta más corta entre dos grafos separados por segmentos, cada uno de un valor en distancia, tal como se aprecia en la Figura 2.4. Este descubrimiento ha permitido el progreso del campo de redes de computadoras y las optimizaciones a este han permitido mejorar su performance, como es el caso de [9] donde realizan mejoras al algoritmo considerando parámetros como consumo de ancho de banda y el tipo de topología aplicado específicamente para SDN.

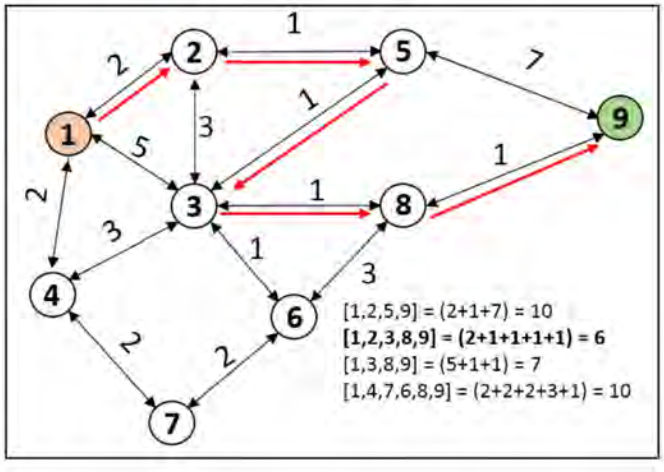


Figura 2.4: Algoritmo de Dijkstra [9]

2.1.4. Protocolos de enrutamiento dinámico utilizados en redes inalámbricas

Existen diferentes tipos de protocolos de enrutamiento dinámico utilizados en redes inalámbricas y cuyo entendimiento compete al desarollo de la presente tesis. Estos debido a su componente dinámico están en constante actualización y se adaptan a cambios en la topología de red para enrutar paquetes eficientemente.

2.1.4.1. Tipos

Los protocolos de enrutamiento se pueden dividir en los siguientes grupos [8]:

- **Proactivos:** Constantemente transmiten paquetes para el descubrimiento de nuevos nodos y rutas dentro de la red en caso sean necesarias. Debido a ello el throughput disminuye producto del aumento de tráfico por el envío de paquetes de descubrimiento, y se tiene una lenta respuesta ante modificaciones en la topología [8]. Algunos ejemplos son los siguientes:
  - **BATMAN:** Aplicado a redes mesh en el cual ningún nodo tiene conocimiento total de la topología de red y solo cuentan con la mejor dirección para llegar a otro nodo, por lo que para que se pueda transmitir a través de la red cada nodo va compartiendo la mejor dirección conforme vayan pasando por él.
  - **Babel:** Similar a BATMAN. Además incluye algoritmos de prevención de bucles sin fin y rápida convergencia.
  - **OLSR:** Similar a BATMAN, solo que en este caso cada nodo comparte su conocimiento de la red por lo que al final cada nodo tiene un conocimiento completo de la topología de red.
- **Reactivos:** A diferencia de los protocolos proactivos, los reactivos solo buscan rutas cuando se presentan modificaciones en la red y se transmiten paquetes por todos los puertos hasta encontrar otra ruta. Ello toma tiempo de ejecución lo que también puede llegar a generar saturación en la red por el sobreenvío de paquetes de descubrimiento [8]. Algunos ejemplos son los siguientes:
  - **AODV:**Debido al tipo reactivo, la primera comunicación presenta una gran latencia, pero con menor consumo de ancho de banda y CPU. Además, cada nodo solo conoce el estado de los nodos vecinos, como el caso de BATMAN y Babel.
  - **DSR:** Similar a AODV, solo que cada nodo cuenta con el conocimiento de las rutas completas, mas no de la topología completa.
- **Híbridos:** Combina las capacidades de los dos tipos anteriormente mencionados. Algunos ejemplos son los siguientes:
  - **IEEE 802.11s:** Funciona a nivel de capa 2, con capacidades similares a AODV, y que además puede contar con cualidades de los protocolos del tipo reactivo, pudiendo incluso ejecutarse de forma simultánea.

2.1.4.2. Comparativa

Así pues, cada uno de los tipos de protocolos de enrutamiento anteriormente comentado tiene sus particularidades, siendo los mejores los protocolos aquellos que consumen menos recursos de sistema como es el caso de AODV o que comparten el conocimiento de los diferentes nodos de una red inalámbrica como BATMAN. Así pues, para fines del desarollo de la presente tesis se considerará uno de cada tipo para realizar las pruebas correspondientes: BATMAN, AODV e IEEE 802.11s que son los que mejores apreciaciones tienen de acuerdo a [8].

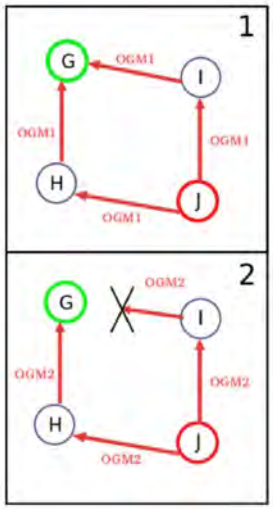


Figura 2.5: Protocolo de enrutamiento BATMAN [8]

2.1.5. Wireless LAN Controllers (WLC)

A continuación se brindan detalles sobre los Wireless LAN Controllers tradicionales, así como una comparativa de los diferentes modelos de este dispositivo.

2.1.5.1. Funcionamiento

De acuerdo a la documentación existente de Cisco Systems Inc. [2], un Wireless Controller tiene como función el gestionar y controlar los componentes de una infraestructura de red inalámbrica como APs, SSIDs, usuarios y políticas de seguridad centralizadas. Estos dispositivos se ubican en lugares donde se cuentan con un gran número de APs como el caso específico de la PUCP, que de acuerdo a lo explicado en el capítulo anterior del presente trabajo, cuenta con más de 1000 APs.

2.1.5.2. Comparativa

A continuación se realizará una comparación de modelos de WLC de diferentes marcas como Cisco, Aruba y Ruckus [10][11][12] que cuentan con similar número de APs que pueden gestionar:

Tabla 2.1: Comparación de WLC de diferentes marcas [10][11][12]

Capacidades	Cisco Catalyst 9800-40	Aruba 7240XM	Ruckus SZ144
Máximo n° de APs soportados	2000	2048	2048
Máximo n° de clientes	32000	32768	40000
Máximo Throughput	40 Gbps	40 Gbps	40 Gbps

De la Tabla 2.1 se puede apreciar que para modelos que soportan similar cantidad de APs, la marca Ruckus presenta mejores prestaciones en cantidad de usuarios y cantidad de APs soportados.

2.2. Software Defined Networking (SDN)

La siguiente sección aborda lo relacionado a Software Defined Networking (SDN), su evolución desde las redes tradicionales, así como un análisis de los controladores SDN.

2.2.1. Evolución de las redes hacia SDN

Para comprender este nuevo paradigma de redes es necesario entender su evolución pues implica un punto de quiebre para el cambio de concepción de las redes de computadoras.

Este proceso resumido en la Figura 2.6 comienza desde los inicios de los dispositivos de red, donde los componentes eran enormes mainframes los cuales cumplían con tareas de enlace, enrutamiento, calidad de servicio (QoS), listas de control de acceso (ACLs) y control de la red, siendo la capa física separada pues vendría a ser el medio alámbrico o inalámbrico por el cual se transmitían los datos. Después de ello, nacen poco a poco nuevos componentes de hardware como el switch, router, dispositivos de gestión de QoS y ACLs, y los mainframes dejan de ejecutar todas las tareas dentro de una red; lo que lleva a que para inicios del presente siglo el componente de control de la red pase a ser únicamente dependiente del software, y con el desarrollo de tecnologías de virtualización lleva al surgimiento del nuevo paradigma de SDN[13].

Figura 2.6: Evolución de las redes hacia SDN a lo largo de los años [13]

Este nuevo paradigma traslada las funciones de transmisión de datos (plano de datos) a los dispositivos de hardware y pasa el control centralizado de una red a los controladores SDN (plano de control). De esta manera se distribuye de una mejor manera las tareas de los componentes de red puesto que los dispositivos de hardware se centran netamente en transmisión de paquetes, dejando las labores de enrutamiento, seguridad y manejo de políticas al software[14].

2.2.1.1. Software Defined Radio (SDR)

Un precedente directo del SDN es el radio definido por software (SDR por sus siglas en inglés). Lo particular de esta tecnología es que permite configurar la frecuencia de transmisión, potencia, conectarse con Digital Signal Processors (DSPs) para el modulamiento de la señal, multiplexación, direccionamiento de la señal mediante la configuración de antenas y manejo de tramas con el protocolo MAC mediante software, con hardware más sofisiticado pero que implica menor replicación dado que puede configurarse dependiendo la aplicación [15].

2.2.2. Controladores SDN

Los controladores SDN son sistemas de software los cuales permiten gestionar el estado de los elementos y topología de una red, cálculo de rutas y que cuentan con una REST API la cual expone los servicios del controlador a las aplicaciones ejecutadas dentro de la red[5].

2.2.2.1. REST API

Parte importante de una red SDN son las REST APIs (Representational State Transfer Application Programming Interfaces), las cuales utilizan el modelo cliente/servidor y se comunican mediante el protocolo HTTP [5]. Los métodos HTTP que usan para comunicarse son los siguientes:

- **POST:** Método usado para la creación o la publicación de datos en la aplicación.
- **GET:** Método usado para obtener datos de la aplicación.
- **PUT:** Método usado para configurar parámetros dentro de una aplicación.
- **DELETE:** Método usado para borrar datos de una aplicación.

La interacción con la aplicación se da mediante un objeto REST el cual se manipula a través de una URL que varía dependiendo la acción a realizar con la aplicación y la REST API utilizada. Un detalle a considerar es que los formatos de datos para su interacción son del tipo XML, JSON o YAML, y que se pueden manejar mediante lenguajes de programación orientados a objetos como Python y Java.

2.2.2.2. Arquitectura

Así como se muestra en la Figura 2.7, dentro de la arquitectura de un controlador SDN idealizado se cuenta con los siguientes componentes[13]:

- **Capa de aplicación (Northbound API):** Parte de la infraestructura en la cual se encuentran las aplicaciones a ejecutarse en la red SDN. Para ello mediante REST APIs basadas en diferentes lenguajes de programación como Java y Python se establece una comunicación mediante llamadas a API de parte del controlador SDN ubicado en la capa de control. Dentro de estas aplicaciones se encuentran el cálculo de rutas, descubrimiento de nuevos dispositivos en la red, así como definición de cambios en las políticas de la red.
- **Capa de control (Modules):** Parte central de la infraestructura donde se encuentra el controlador el cual está conectado mediante APIs (interface Northbound) con las aplicaciones a ejecutarse en la red SDN y con los componentes de hardware de la infraestructura de red mediante el protocolo OpenFlow (interface Southbound).
- **Capa de infraestructura (Southbound API):** Parte de la infraestructura en donde se encuentran los componentes de hardware como switches y APs que cuentan con el protocolo OpenFlow, un protocolo que permite la comunicación entre el plano de control y el hardware (plano de datos) para aplicación de políticas de red dadas por la aplicaciones ejecutadas. A su vez, los dispositivos de hardware son los que se conectan a los Endpoints o usuarios finales de la red.

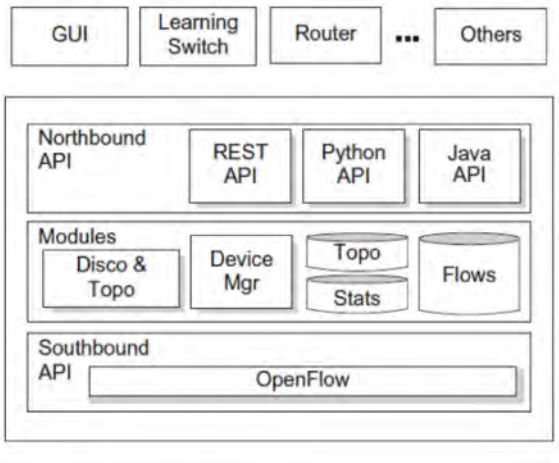


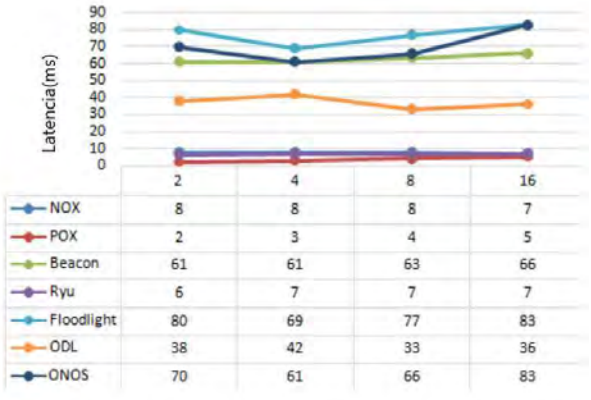
Figura 2.7: Arquitectura de un controlador SDN idealizado [13]

2.2.2.3. Comparativa

Para la elección de un controlador SDN se tienen en cuenta los siguientes parámetros:

- **Throughput:** Métrica que mide la cantidad de respuestas de un controlador en un intervalo de tiempo.
- **Latencia:** Métrica que mide el retardo de respuesta y transmisión de paquetes en una red.

Dados estos parámetros se encontraron estudios de pruebas utilizando Cbench (Controller Bench Marker) para medir estos parámetros en diferentes tipos de controladores [16] obteniéndose los siguientes resultados:



	2	4	8	16
NOX	8	8	8	7
POX	2	3	4	5
Beacon	61	61	63	66
Ryu	6	7	7	7
Floodlight	80	69	77	83
ODL	38	42	33	36
ONOS	70	61	66	83

Figura 2.8: Latencia en diferentes controladores SDN [16]

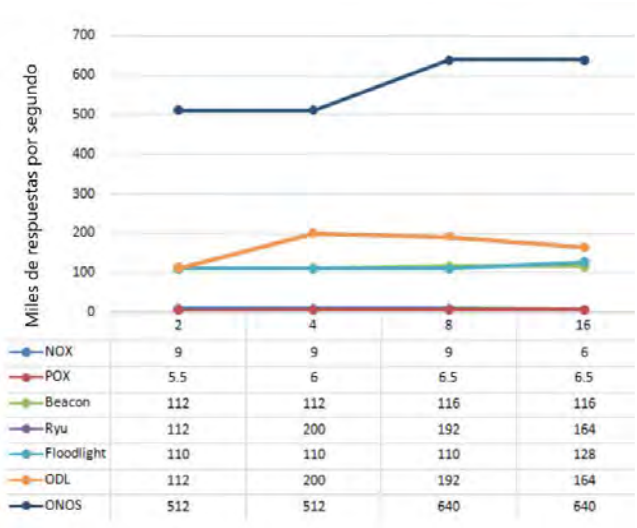


Figura 2.9: Throughput en diferentes controladores SDN [16]

A partir de las gráficas de resultados anteriores de la Figura 2.8 y la Figura 2.9 se tiene que los controladores NOX, POX y Ryu tienen los niveles más bajos de latencia; y que el controlador ONOS tiene la mayor tasa de throughput, seguido por ODL (OpenDayLight) y Ryu.

Así también, la Tabla 2.2 muestra mayores detalles de los diferentes controladores SDN. Dentro de estas características destacan las siguientes el año de lanzamiento, si es multi-thread, las APIs que utiliza, así como la versión de OpenFlow(OF) que soporta, el lenguaje de programación que utiliza como Java y Python (aunque también existen controladores como el NOX que se programa en C++) y la calidad de la documentación.

Tabla 2.2: Comparación de Controladores SDN [16]

	POX	**Beacon**	**Ryu**	**Floodlight**	**ODL**	**ONOS**
First Release	2011	2010	2012	2012	2013	2014
License	Apache 2.0	GPL 2.0	Apache 2.0	Apache 2.0	EPL 1.0	Apache 2.0
Multi-Thread	No	Yes	Yes	Yes	Yes	Yes
Platform	Linux MAC Windows	Linux MAC Windows	Linux MAC Windows	Linux MAC Windows	Linux MAC Windows	Linux MAC Windows
APIs	OVSDB OF REST	OVSDB OF REST	OVSDB, OF, REST NETCONF OFCONFIG	OVSDB, OF REST	OVSDB, OF,REST, YANG, NETCONF, PCEP, BGPSNMP	OVSDB, OF REST, NETCONF
OF Version	1	1	1.0 to 1.5	1.0 to 1.3	1.0 to 1.3	1.0 to 1.3
Documentation	Poor	Poor	Medium	Very Good	Very good	Medium
Language	Python	Java	Python	Java	Java	Java
Legacy Network	No	No	No	No	Yes	Yes
Category	Single	Single	Single	Single	Distributed	Distributed
GUI	Poor	Poor	Medium	Medium	Very Good	Very Good
Regular Updating	Poor	Poor	Medium	Medium	Very Good	Very Good
Modularity	Poor	Medium	Medium	Medium	Very Good	Very Good

De la Tabla 2.2 se tiene que el controlador Ryu soporta la gran mayoría de versión de OpenFlow (de la 1.0 a la 1.5), está basado en Python, un lenguaje de programación orientado a objetos que cuenta con gran cantidad de blibliotecas para el desarrollo de aplicaciones, por lo que sería la primera opción a considerar para el desarrollo del presente proyecto de tesis.

Así también, otro controlador que cumple con los requerimientos es OpenDayLight (ODL) el cual cuenta con una gran soporte a diferentes tipos de APIs, posee buena documentación y está basado en Java, un lenguaje de programación robusto para el desarollo de aplicaciones. Asimismo cuenta con modo distribuido lo que permitiría que se desplieguen múltiples controladores SDN en una red a diferencia del controlador Ryu.

2.3. Redes inalámbricas basadas en SDN

En la presente sección se detallará sobre las redes inalámbricas basadas en SDN, su arquitectura, componentes, así como parámetros de consideración para la implementación del Wireless Controller.

2.3.1. Arquitectura inalámbrica basada en SDN

La arquitectura de las redes inalámbricas basadas en SDN es similar al caso de la arquitectura de una red SDN cableada con la diferencia de que considera componentes compatibles con el protocolo WiFi (IEEE 802.11). Así pues, estos están compuestos por APs los cuales permiten la conexión de los endpoints con los demás componentes de la red. Estos access points se conectan alámbricamente con los switches OpenFlow, los cuales permiten conectarse con el controlador SDN [14]. Además existen tipos de Access Points SDN los cuales cuentan con protocolo OpenFlow, integrando un switch OpenFlow y un AP.

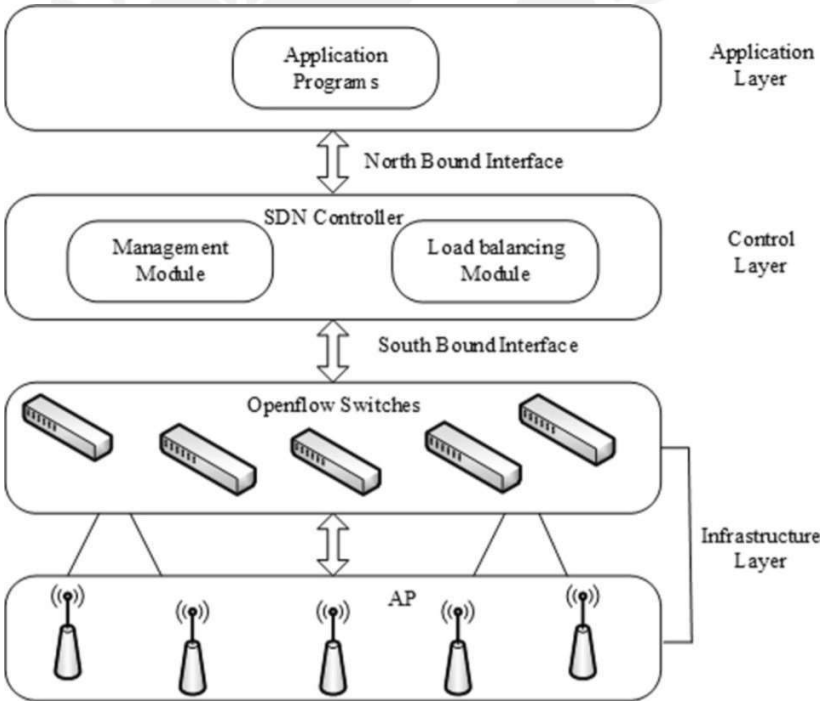


Figura 2.10: Arquitectura inalámbrica basada en SDN [14]

2.3.2. Wireless Controllers basados en SDN

De la Figura 2.10 se aprencian los módulos de gestión y balanceo de carga de la capa de control, que se ejecutan en la capa de aplicación y cuyas funciones son las siguientes:

- **Módulo de gestión:** Encargado de la gestión de APs y terminales, así como de descubrir y configurar nuevos APs.

- **Módulo de balanceo de carga:** Este componente se despliega en la capa de aplicación y mediante un algoritmo asigna pesos a cada AP de la red inalámbrica de acuerdo a su tráfico, de manera que este permita una adecuada comunicación entre los componentes de la red sin saturarla.

2.3.2.1. Parámetros de consideración

De acuerdo a [17], los parámetros de consideración para una adecuada implementación de un Wireless Controller son los siguientes:

- **Latencia:** En adición a lo comentado anteriormente respecto a los controladores SDN, la demora en la transmisión de paquetes a través de la interface Southbound afecta a la calidad de servicio de una red inalámbrica.

- **Estabilidad del plano de control:** Está relacionado a componentes de throughput y se relaciona al rápido cambio del plano de control ante anomalías en la red SDN.

2.3.3. Consideraciones para la implementación del Wireless Controller

Luego de la revisión de los fundamentos teóricos necesarios para el desarrollo de la tesis se presentan las consideraciones para la implementación del Wireless Controller.

2.3.3.1. Diagrama de bloques del sistema

En la Figura 2.11 se muestra el diagrama de bloques del sistema en el cual se muestra la arquitectura del sistema a desarrollar para la implementación del Wireless Controller con cálculo de rutas. Así pues, se tendrá en la capa de control a los controladores Ryu y OpenDayLight tal como se comentó en la sección de comparación de los diferentes tipos de controladores SDN. En lo que respecta a la capa de aplicación se tendrá a la aplicación del Wireless Controller con cálculo de rutas, el cual será como una aplicación la cual mediante llamadas de API podrá realizar el enrutamiento dentro de la red. En la capa de infraestructura se contará con Access Points Legacy que se conectan con los usuarios de la red inalámbrica, y que a su vez se comunican alámbricamente con switches SDN y estos a su vez mediante el protocolo OpenFlow con el controlador SDN. Cada componente será desplegado en una propia máquina virtual y se

contará con switches SDN y APs físicos o simulados tal como se comentará a detalle en la siguiente sección.

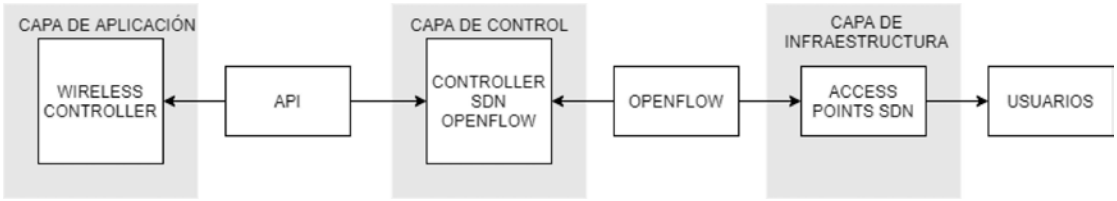


Figura 2.11: Diagrama de bloques del sistema (Elaboración propia)

Así también, se considerará uno de los protocolos de enrutamiento anteriormente mencionados para evaluar las variaciones existentes si es que se implementa con protocolos proactivos, reactivos o híbridos.

2.4. Ambientes de simulación para Controladores SDN

A continuación se listan los ambientes de simulación para redes SDN y que permitirán el desarrollo de la presente tesis.

2.4.1. MiniNet y MiniEdit

MiniNet es un simulador de redes SDN de código abierto el cual permite la prueba de controladores SDN tanto locales como remotos con componentes que uno puede definir en su interfaz gráfica llamada MiniEdit en la cual se pueden agregar switches OpenFlow virtuales, routers legacy y host que simulan el tráfico en una red. Un detalle a considerar de este simulador es que está basado en Python por lo que el desarrollo es más rápido por el fácil uso de este lenguaje de programación, y es ejecutado exitosamente en MAC y Linux [18].

Así también, puede ejecutarse en entornos virtuales con herramientas de virtualización como VirtualBox o VMware, en los cuales se pueden desplegar múltiples máquinas virtuales y personalizar para que en una de ellas se encuentre el controlador SDN a utilizar y en otra se encuentre la máquina virtual con el MiniNet integrado.

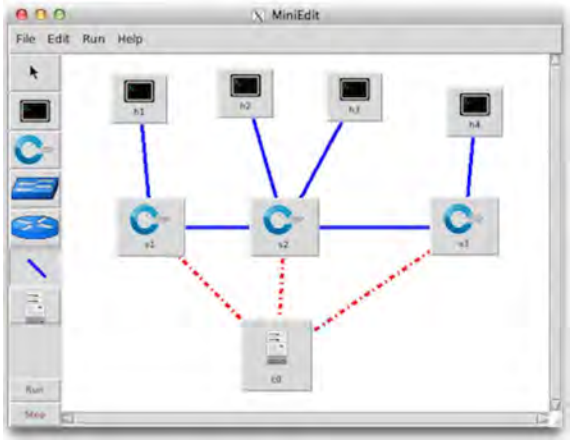


Figura 2.12: Interfaz de MiniNet y MiniEdit [18]

2.4.1.1. MiniNet-WiFi

MiniNet WiFi es un entorno de simulación para redes SDN inalámbricas, basado en MiniNet en el cual se pueden realizar pruebas con diferentes protocolos de enrutamiento dinámico para redes inalámbricas y además se puede añadir APs virtualizados a diferencia de su predecesor, el MiniNet. También cuenta con funcionalidades que permiten emular los atributos de estaciones móviles con base a su posición y movimiento relativo de los access points[19]. Así también cuenta con las mismas capacidades de emulación de SDN que MiniNet y la opción de visualizar las topologías inalámbricas como se aprecia en la Figura 2.13 , siendo una opción más completa para el desarrollo de la presente tesis.

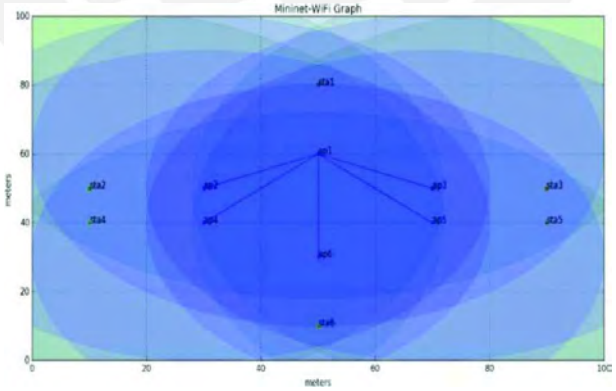


Figura 2.13: Visualización de topología de red en Mininet-WiFi [20]

2.4.2. Virtual Network Research Testbed (VNRT)

VNRT es un emulador de redes desarrollado por AINET Solutions S.A.C. con el que cuenta el Grupo de Investigación en Redes Avanzadas (GIRA) de la PUCP. Es un entorno de pruebas de alta

fidelidad y alta capacidad que sirve para probar el comportamiento de nuevas tecnologías. Este permite el despliegue de máquinas virtuales, componentes de red SDN, en diferentes topologías y permite la conexión con controladores SDN externos mediante su interfaz gráfica. Así también, dado que se encuentra instalado en el laboratorio, se tendrá acceso a este de manera remota.

Dado que el proyecto está asociado a GIRA, se utilizará VNRT como ambiente de pruebas para el diseño del Wireless Controller a diseñar. y en conjunto con los ambientes de simulación anteriormente comentados permitirán medir los parámetros anteriormente definidos y que determinarán el éxito de la implementación del controlador. Además, se ha asignado la siguiente topología en el VNRT (2.14), la cual será utilizada para la realización de pruebas preliminares y familiarización con el emulador.

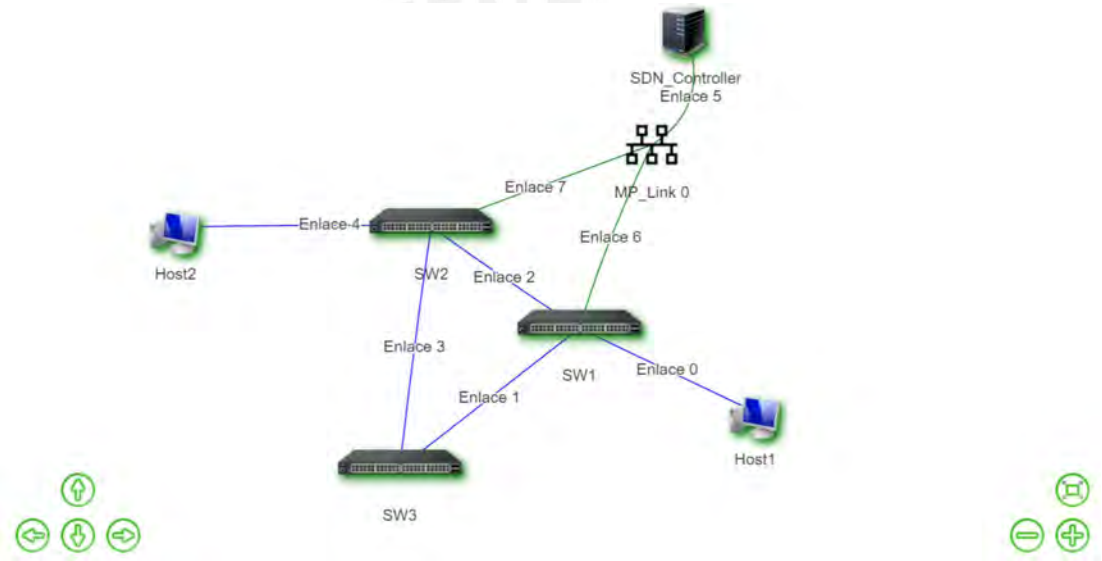


Figura 2.14: Topología para pruebas iniciales en VNRT [21]

Un detalle más a considerar es que el VNRT en el laboratorio de GIRA tiene acceso a APs legacy de marca TP-Link con software OVS (Linux Embebido OpenWRT), lo que los convierte en APs para SDN. Asímismo, el laboratorio cuenta con switches SDN/OpenFlow físicos marca Pica 8.

Capítulo 3

Diseño del Wireless Controller con cálculo de rutas basado en SDN

El presente capítulo describe el diseño del Wireless Controller con cálculo de rutas basado en SDN. En este se explicarán los requerimientos y criterios a considerar en el diseño del mismo, la topología a utilizar para su validación, así como las pruebas con los entornos de simulación y emulación de redes SDN expuestos en el capítulo anterior.

3.1. Requerimientos de diseño del Wireless Controller

En la presente sección de describirá en detalle los requerimientos para el diseño del Wireless Controller en base a lo revisado en los capítulos de Marco problemático y Fundamentos teóricos de la tesis. Además, se explicará a mayor detalle los recursos con los que se cuenta en el laboratorio GIRA, tanto de software como de hardware, que servirán para el desarrollo de pruebas y validación del mismo.

3.1.1. Requerimientos

De acuerdo a lo explicado en los capítulos 1 y 2 de la presente tesis, el propósito de la misma es brindar una solución al problema de la infraestructura Wireless de la PUCP la cual cuenta con un Wireless LAN Controller enmarcado en el paradigma convencional de redes, por lo cual es crucial que el Wireless Controller a diseñar cumpla con las condiciones actuales a las que está sometido el WLC del campus. Es por ello que se consideran los siguientes requerimientos:

- **Throughput:** De lo revisado en el capítulo 1 [5], concerniente al tráfico hacia internet, este se encuentra alrededor de los 40Gbps.

- **Latencia:** Las latencias deben ser mínimas por lo que se considera un valor menor a 20 ms.
- **Dispositivos de red conectados:** De los datos revisados en los portales de transparencia de la PUCP [4], la PUCP cuenta con más de 1000 Access Points en su infraestructura inalámbrica del Campus Pando por lo que el Wireless Controller a diseñar debe soportar al menos esa cantidad de dispositivos de red.

En adición a lo listado anteriormente, se agrega el tema de que el diseño debe considerar control SDN tanto centralizado como distribuido, tal como se explicó en el capítulo 2 y que llevó a la elección del controlador Ryu y OpenDayLight. La explicación respecto al cómo es que funcionan estos tipos de control SDN se explicarán en la siguiente sección de este capítulo. Asimismo, este aspecto está relacionado con la tolerancia a fallas que debe contar el diseño del Wireless Controller, ello para que pueda ser utilizado en una red de campus académica de alta disponibilidad como es el caso de la red inalámbrica de la PUCP.

Otro detalle a considerar y cuya relevancia es determinante es el factor de seguridad que la implementación va a contar. Así pues, se considerará la inclusión de un bloque de seguridad básico que permitirá un filtrado básico de paquetes, ello para que luego sea mejorado por futuras integraciones, dado que el propósito de la tesis no está centrado en temas de seguridad.

Los requerimientos comentados se describen de forma resumida en la Tabla 3.1.

Tabla 3.1: Tabla de requerimientos de diseño del Wireless Controller (Elaboración propia)

Requerimiento	***Descripción***
Latencia máxima	20 ms
Throughput	40 Gbps
Cantidad de equipos de red conectados(Switches y APs)	1000
Tipo de control SDN	Centralizado y distribuido
Tolerable a fallas	Sí
Tipo de seguridad de la red	Básica

3.1.2. Recursos necesarios

Luego de haber defnido los requerimientos para el diseño del Wireless Controller es necesario analizar los recursos con los que se cuenta para la implementación, pruebas y validación del mismo. La presente tesis se realiza en colaboración con el Grupo de Investigación de Redes Avanzadas (GIRA) de la PUCP el cual dentro de su logística cuenta con un servidor HPC (High Performance Computing) de la marca Hitachi en el cual se encuentra desplegado el emulador Virtual Network Research Testbed (VNRT). El VNRT, como herramienta virtualizada

permitirá el despliegue y el desarrollo de pruebas dado que es posible crear dentro del mismo múltiples máquinas virtuales y conexiones entre las mismas, lo que se asemejaría a un entorno real, permitiendo la validación del Wireless Controller. Las especificaciones técnicas del HPC Hitachi se encuentran detalladas en la Tabla 3.2.

Tabla 3.2: Tabla de especificaciones técnicas del HPC Hitachi [22]

Especificación	***Descripción***
Máximo número de máquinas virtuales	1000
Memoria RAM	256 GiB
Almacenamiento	10 TiB
Máxima Capacidad de la tarjeta de Red	8 Gbps

En adición a lo anteriormente comentado, el laboratorio de GIRA cuenta con equipos de red como switches SDN/OpenFlow físicos marca Pica 8 y los Access Points de marca TP-Link modelo TL-WDR3600 (N600) con software OVS (Linux Embebido OpenWRT versión 18.06.4), los que permitirán las pruebas respectivas y que serán explicadas más adelante en este capítulo. Así también, estos dispositivos pueden conectarse de forma cableada con el HPC Hitachi e interactuar con las máquinas virtuales creadas en el emulador VNRT, facilitando las pruebas a realizar en el mismo.

La lista completa de los recursos a utilizar para el diseño del Wireless Controller se encuentra detallada en la Tabla 3.3.

Tabla 3.3: Tabla de recursos necesarios para el desarrollo del Wireless Controller (Elaboración propia)

Recurso	***Cantidad***
HPC Hitachi	1
Switches SDN/OpenFlow Pica 8	1
Access Points TP-Link con Linux Embebido OpenWRT	1

Un detalle a considerar es el hecho de que debido al contexto actual las pruebas con el Access Point TP-Link con Linux Embebido OpenWRT serán realizadas en ambientes locales, diferentes al laboratorio de GIRA.

3.2. Consideraciones de diseño del Wireless Controller

Después de haberse definido los requerimientos de diseño de Wireless Controller en la sección anterior se procede al diseño del mismo, para lo cual se tiene en cuenta las arquitecturas de los controladores a utilizar, siendo en este caso, y de acuerdo a lo explicado en el capítulo 2, los controladores Ryu y OpenDayLight, los cuales soportan control SDN centralizado y distribuido respectivamente. Así también, se ha tomado como referencia investigaciones anteriores [23][24] en las cuales se ha considerado tópicos como escalabilidad y tolerancia a fallas lo que también se ve plasmado en la lista de requerimientos anterior.

3.2.1. Arquitectura de los controladores a utilizar para el diseño del Wireless Controller

A continuación se explicarán las arquitecturas de cada uno de los controladores a utilizar para el diseño del Wireless Controller:

3.2.1.1. Arquitectura del controlador Ryu

En adición a los explicado en el capítulo 2 respecto a los controladores SDN Ryu, los cuales están basados en el lenguaje de programación Python y dentro de sus capacidades se encuentra el soporte del protocolo OpenFlow de la versión 1.0 a la 1.5 [16]; se añade que estos cuentan con una diversa cantidad de módulos dentro de su arquitectura tal como se aprecia en la Figura 3.1.

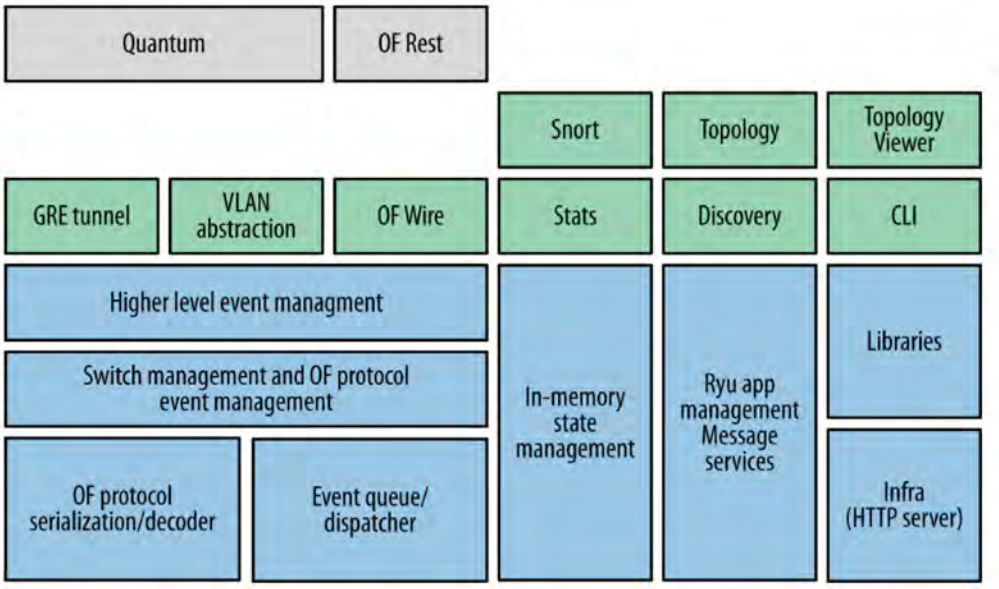


Figura 3.1: Arquitectura del controlador Ryu [25]

Dentro de los módulos de la arquitectura del controlador Ryu, se utilizarán los siguientes[25][26]:

- **Discovery:** Este módulo será el encargado de descubrir nuevas rutas en caso algún dispositivo de red intermedio falle. Además, en caso nuevos usuarios se conecten a la red, este los descubrirá y añadirá a la topología de red inalámbrica existente.
- **CLI:** Este módulo permite ejecutar comandos en los diferentes dispositivos de red para así visualizar y realizar las configuraciones pertinentes.
- **Topology:** Módulo a cargo de estar al tanto de la topología de red y de los nodos existentes en la misma.
- **Topology Viewer:** Módulo que permite visualizar la topología, el cual es de mucha ayuda para tener un esquema general de la red.
- **Ryu app management:** Módulo que permite la ejecución de aplicaciones dentro del controlador, así como su ejecución entre los dispositivos de red conectados.

3.2.1.2. Arquitectura del controlador OpenDayLight

Respecto a los controladores SDN OpenDayLight, los cuales están basados en el lenguaje de programación Java y dentro de sus capacidades se encuentra en que soportan el protocolo OpenFlow de la versión 1.0 a la 1.3 [16]; se añade que estos cuentan con una diversa cantidad de módulos dentro de su arquitectura tal como se aprecia en la Figura 3.2.

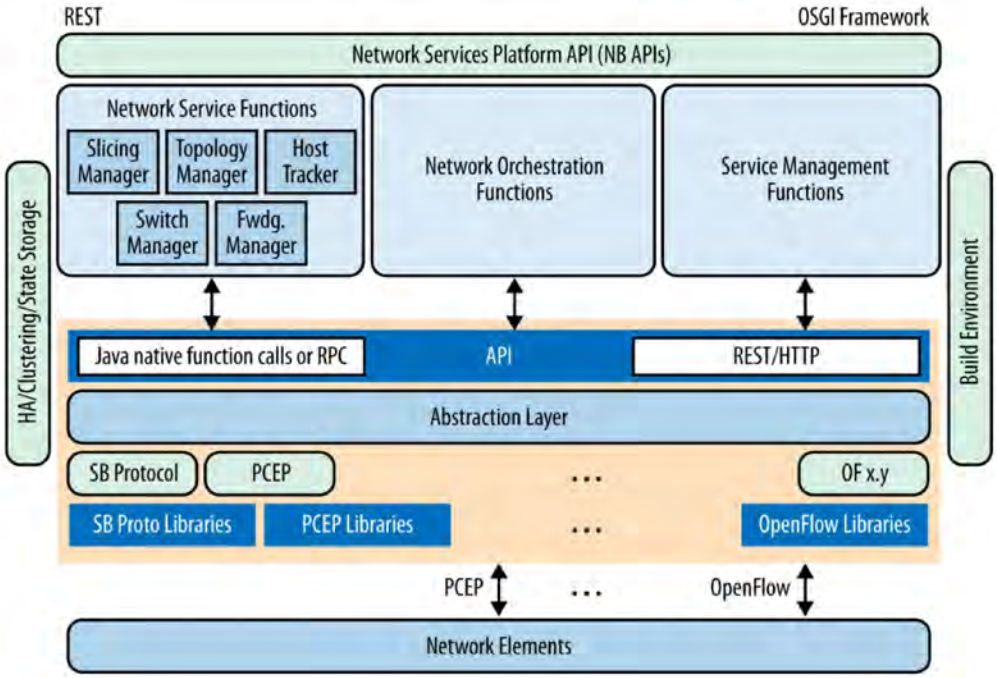


Figura 3.2: Arquitectura del controlador OpenDayLight [25]

Dentro de los módulos de la arquitectura del controlador OpenDayLight, se utilizarán aquellos que se encuentran dentro del bloque de *Network Service Functions* los cuales son los siguientes [25][27]:

- **Slicing Manager:** Este módulo será el encargado de gestionar las diferentes divisiones existentes dentro de una red.
- **Host Tracker:** Este módulo está a cargo del seguimiento y descubrimiento de nuevos hosts dentro de la red.
- **Topology Manager:** Módulo a cargo de estar al tanto de la topología de red y de los nodos existentes en la misma, similar al caso de Ryu.
- **Switch Manager:** Módulo que permite la gestión de switches dentro de la red.
- **Forwarding Manager:** Módulo que permite la adecuada transmisión de paquetes dentro de la red.

3.2.2. Diseño de la topología

En la presente subsección se ofrecen detalles del diseño de la topología que permitirá la validación del Wireless Controller, tales como la consideración de una arquitectura de

controladores SDN distribuidos para el caso del controlador OpenDayLight el cual cuenta con esta característica.

3.2.2.1. Arquitectura de controladores SDN distribuidos

En esta sección se detallará sobre la arquitectura de controladores SDN distribuidos, los cuales buscan ser tolerantes a fallas en una red. La idea de este tipo de arquitectura es de tener un plano de control descentralizado, en el cual cada uno de los controladores conectados está a cargo de un determinado dominio en el cual se tienen diferentes dispositivos de red como switches y APs. Además, la comunicación entre estos es realizada mediante un bus de comunicación AMQP de manera que las políticas estén homologadas entre los distintos dominios de una red. Incluso hay casos en los cuales se cuenta con un controlador maestro que gestiona los demás controladores a cargo de dominio [28]. Así como se puede apreciar en la Figura 3.3, la consideración sería tener controladores diferenciados para cada dominio de la red.

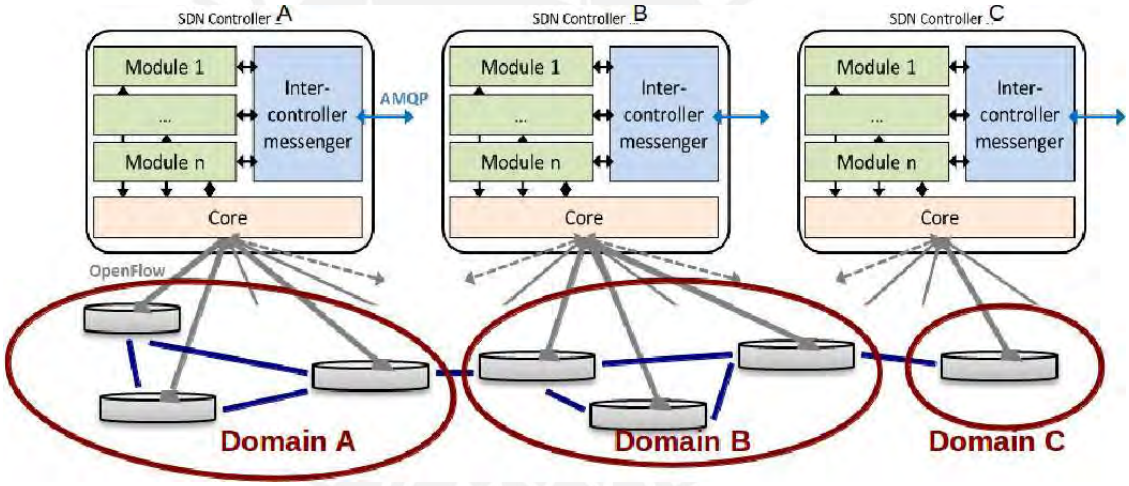


Figura 3.3: Arquitectura de Controladores SDN distribuidos [28]

3.2.2.2. Topología diseñada

Para el diseño de la topología que permitirá las pruebas y validación del Wireless Controller se tomó como referencia la topología de la Figura 3.4 usada para pruebas del controlador PUCPLight [24], el cual usó un entorno de pruebas con similares especificaciones técnicas al HPC Hitachi, el cual fue un exoGENI.

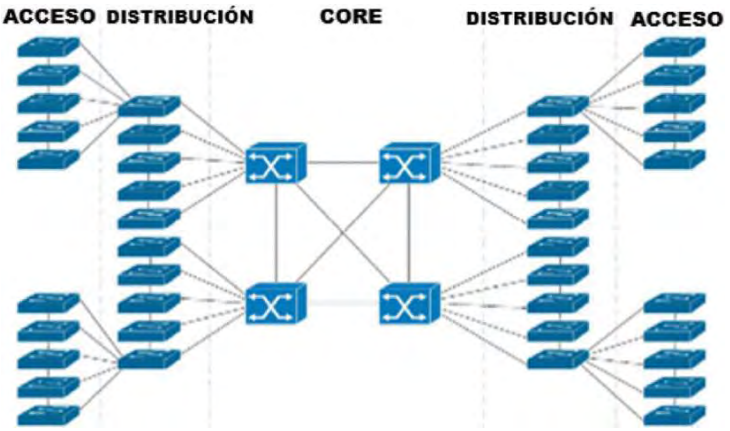


Figura 3.4: Topología usada para pruebas del controlador PUCPLight [24]

Para el caso de la topología diseñada (Figura 3.5), se realizó una actualización a la topología presentada en el capítulo 2 y se tuvo en consideración los requerimientos expuestos al inicio del presente capítulo. Así pues, la presente topología presenta dominios tanto en el lado derecho e izquierdo, los cuales en el caso de un control distribuido con controladores OpenDayLight permitirán que se empleen 2 controladores. Además de ello cuenta con un switch concentrador central el cual a su vez tendrá la función de Firewall y realizará un filtrado básico de los paquetes que entren y salgan hacia internet. Un detalle a considerar es que en la descripción de la figura se incluyen un número indeterminado n de dispositivos conectados, valor que irá variando de acuerdo a las pruebas de esfuerzo realizadas y que se comentará más adelante.

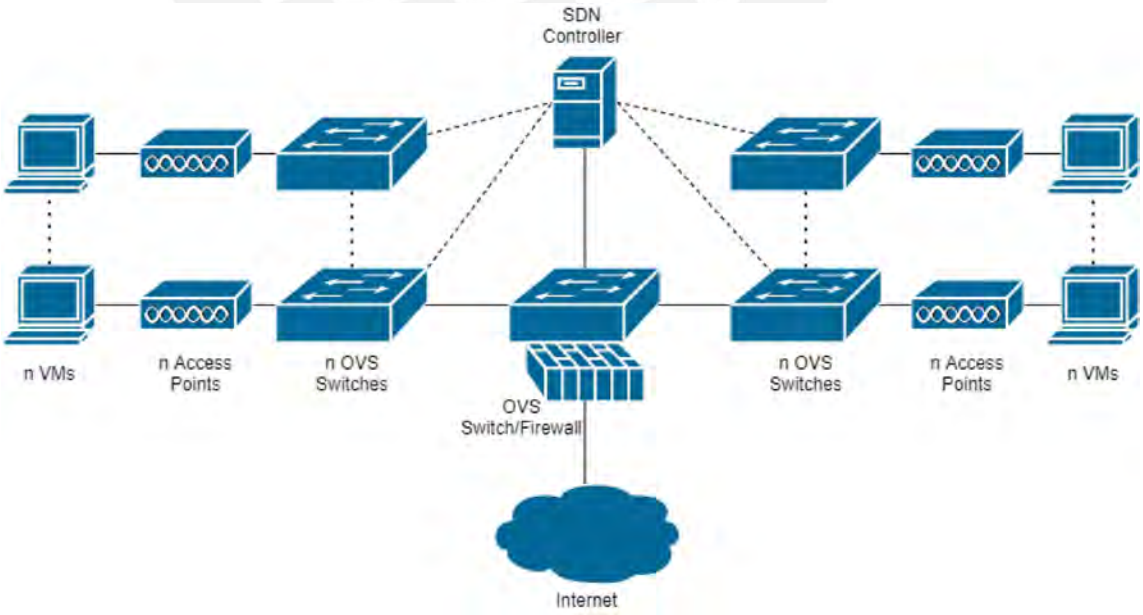


Figura 3.5: Nueva topología del sistema (Elaboración propia)

3.3. Escenarios de pruebas

En adición a lo comentado en el capítulo 2, dentro de los escenarios de pruebas a realizar como parte de la validación del presente proyecto se tendrán los siguientes:

- **Pruebas de Throughput:** Se probará la solución desarrollada con test similares a los de Cbench e Iperf[16] en donde se manda gran cantidad de paquetes para así conocer el throughput del sistema SDN, a fin de contrastar si las prestaciones están de acuerdo con los WLC tradicionales.
- **Pruebas de latencia:** Se probará la solución desarrollada con test similares a los de Cbench e Iperf[16] en donde se manda un único paquete a la red del dispositivo, ello para conocer el tiempo de respuesta.
- **Pruebas con número variado de switches, APs y usuarios conectados:** Se probará la solución desarrollada con número variado de switches, APs y usuarios a fin de contrastar si las prestaciones están de acuerdo con los WLC tradicionales. Cabe resaltar que los APs que se utilizarán se emplearán en escenarios distintos al laboratorio GIRA debido al contexto actual.

Para complementar a lo explicado en los escenarios de prueba, se realizarán simulaciones de tráfico de un dominio hacia otro, tal como si fuera un escenario real dentro del campus PUCP. Para ello en uno de los nodos de la derecha se generará tráfico artificial mediante scripts que fluirá tanto para el nodo de la izquierda como hacia internet. De esta manera se obtendrán parámetros como throughput y latencia, que a su vez tendrán resultados diferentes conforme se varíe la cantidad de dispositivos de red y usuarios conectados a la misma. Cabe resaltar que todo es posible gracias a que dentro del VNRT alojado en el HPC Hitachi es posible generar nuevas máquinas virtuales de manera dinámica, sin necesidad de hardware adicional.

3.4. Pruebas de los entornos de simulación y emulación

En la presente sección se comentan las pruebas realizadas a los entornos de simulación y emulación explicados en el capítulo 2.

3.4.1. Pruebas con el VNRT

Para las pruebas preliminares se empleó el VNRT del laboratorio de GIRA en el cual se desplegó una versión básica de la topología presentada en una sección anterior.

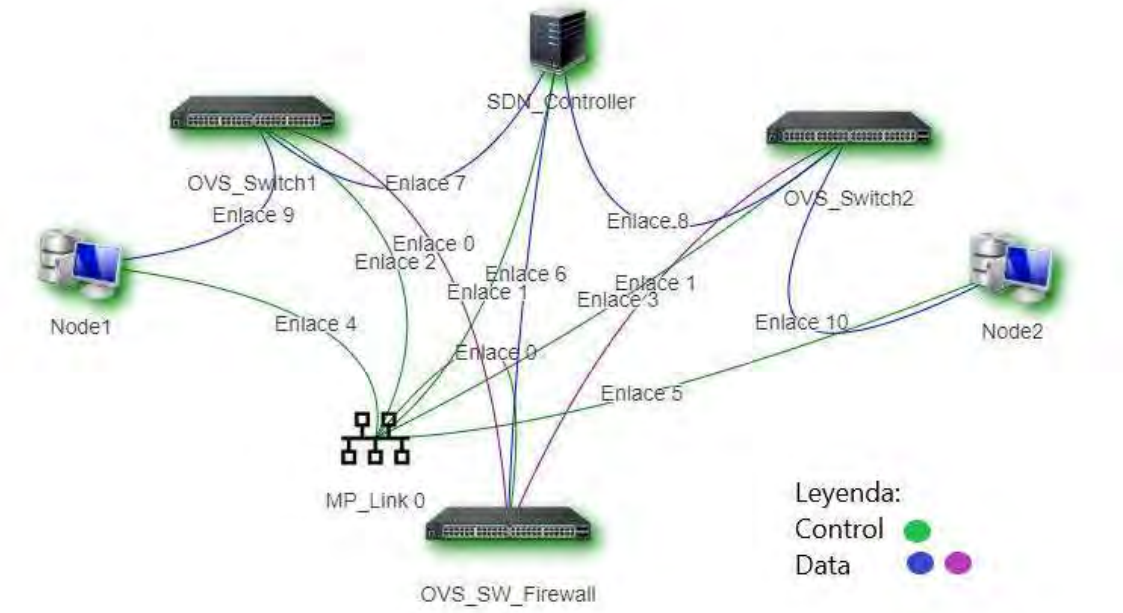


Figura 3.6: Topología del sistema en VNRT (Elaboración propia)

A continuación se lista información de cada una de las máquinas virtuales creadas en la anterior topología en el VNRT, la cual se encuentra detallada en la Tabla 3.4.

Tabla 3.4: Tabla de especificaciones de cada máquina virtual del VNRT (Elaboración propia)

Máquina Virtual	***# CPUs***	***Memoria RAM***	***Almacenamiento***
Controlador SDN	2	512 MiB	10 GiB
OVS Switches	2	512 MiB	4 GiB
Nodos	2	512 MiB	2 GiB

Asimismo, se han realizado pruebas ejecutando el controlador Ryu en conjunto con el simulador Mininet en la máquina central principal del VNRT, logrando visualizar la interfaz web (Figura 3.7) en donde se puede apreciar la topología generada [26]

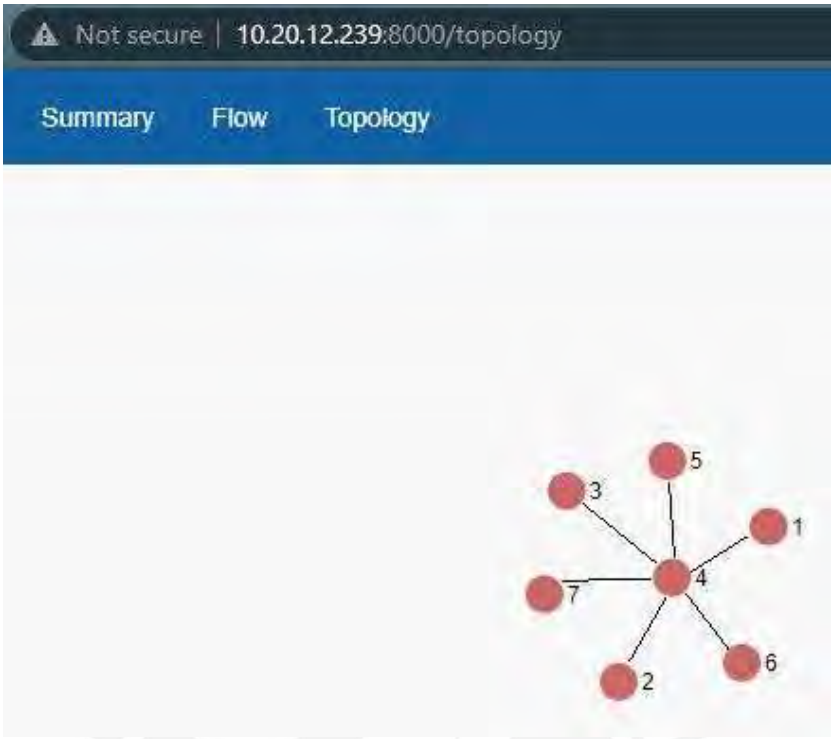


Figura 3.7: Interfaz gráfica del controlador Ryu (Elaboración propia)

Capítulo 4

Pruebas y resultados

En el presente capítulo se presentan las pruebas realizadas del diseño del Wireless Controller con cálculo de rutas basado en SDN, la medición de las métricas de red de throughput y latencia definidas en los capítulos anteriores; así como su respectiva validación.

4.1. Pruebas realizadas

Tal como se describe en el capítulo 3 de la presente tesis, para las pruebas de validación el diseño del Wireless Controller basado en SDN, se incluirá un bloque de generación de tráfico además de la configuración de red; así como una pertinente configuración de los perfiladores a utilizar los cuales medirán las métricas de red deseadas: throughput y latencia. Los detalles de los mismos se muestran en la presente sección.

4.1.1. Configuración de las redes a utilizar

4.1.1.1. Configuración de la solución en el VNRT

Dado que las pruebas se realizan de manera simulada, es necesario primero configurar el VNRT de manera que exista una adecuada comunicación entre las diferentes máquinas virtuales que emulan al controlador SDN, a los switches y los hosts tal como fue definido en la Figura 3.6 y que cumple con las funciones deseadas por el mismo.

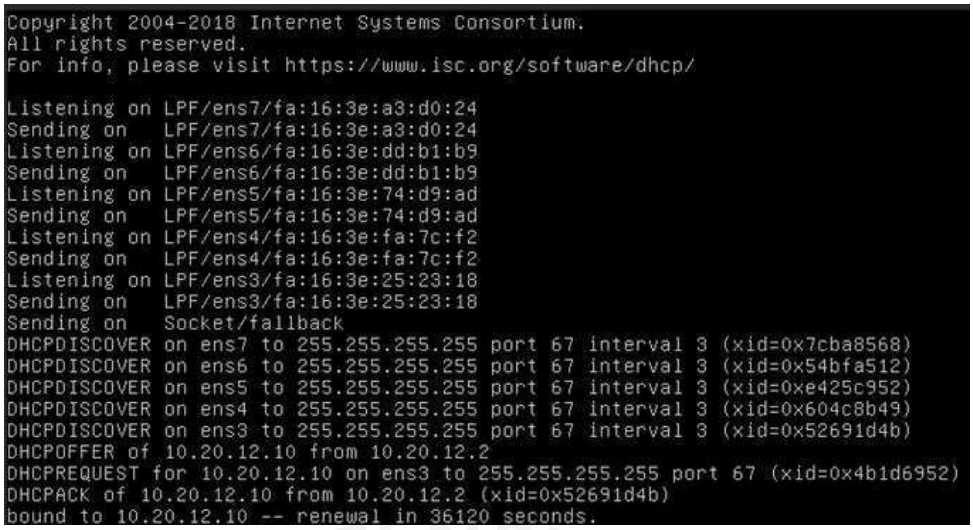


Figura 4.1: Configuración del entorno del VNRT (Elaboración propia)

En la Figura 4.1 se aprecia una de las muchas configuraciones realizadas en el entorno como lo es el designar IPs mediante el protocolo DHCP a cada uno de los componentes de la red desplegada. En adición a ello tuvieron que configurarse los espacios de disco de cada una de las máquinas virtuales, tal como es descrito en la Figura 4.2 a fin de que fuera posible instalarse los programas posteriormente descritos.

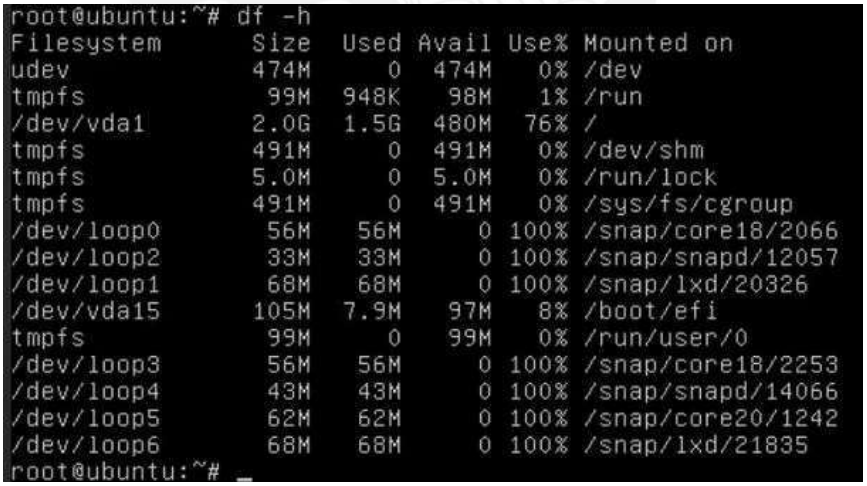


Figura 4.2: Configuración del entorno del VNRT (Elaboración propia)

4.1.1.2. Configuración del access point SDN

En adición a la configuración de red, se tuvo permiso de acceder a las instalaciones del laboratorio de GIRA para poder manipular y configurar los AP TP-LINK TL-WDR3600(N600) (Figura 4.3) que soportan SDN y cuyo sistema operativo es OpenWRT 18.0.4 (Figura 4.4) el cual

es compatible con el protocolo OpenFlow y permite la configuración del mismo en una red SDN.

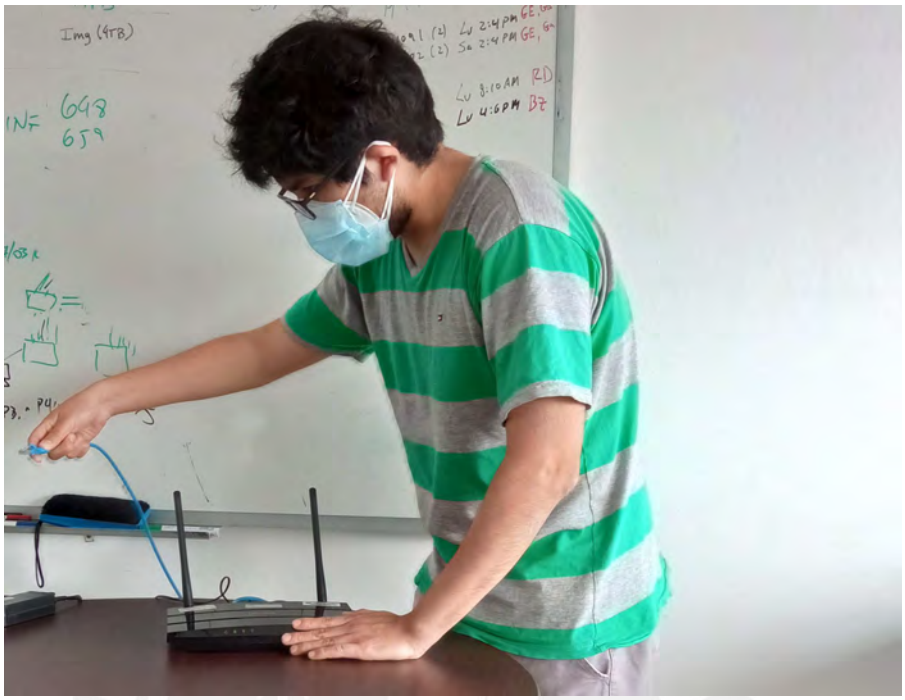


Figura 4.3: Conexión del AP TP-Link TL-WDR3600(N600) (Elaboración propia)

Figura 4.4: Interfaz del OpenWRT 18.0.4 [29]

Dentro de las configuraciones realizadas estuvieron contempladas la configuración de las interfaces para que puedan conectarse con el controlador, así como se describe en la Figura 4.5. Lamentablemente, por tema de accesos no se pudo llegar a integrar con el VNRT desplegado en el HPC Hitachi.

Figura 4.5: Configuración de interfaces en el OpenWRT 18.0.4 (Elaboración propia)

4.1.2. Generación de tráfico

4.1.2.1. Plex Media Server

Para el tema de generación de tráfico se ha considerado el desplegar un Media Server, el cual almacenará archivos de video los cuales serán transmitidos dentro de una red local. Para el caso de las pruebas se utilizará la plataforma Plex la cual es de código abierto y permite la transmisión de video streaming para simulación de tráfico y medición del desempeño de la red. El funcionamiento del mismo se explica claramente en la Figura 4.6 mostrada a continuación.

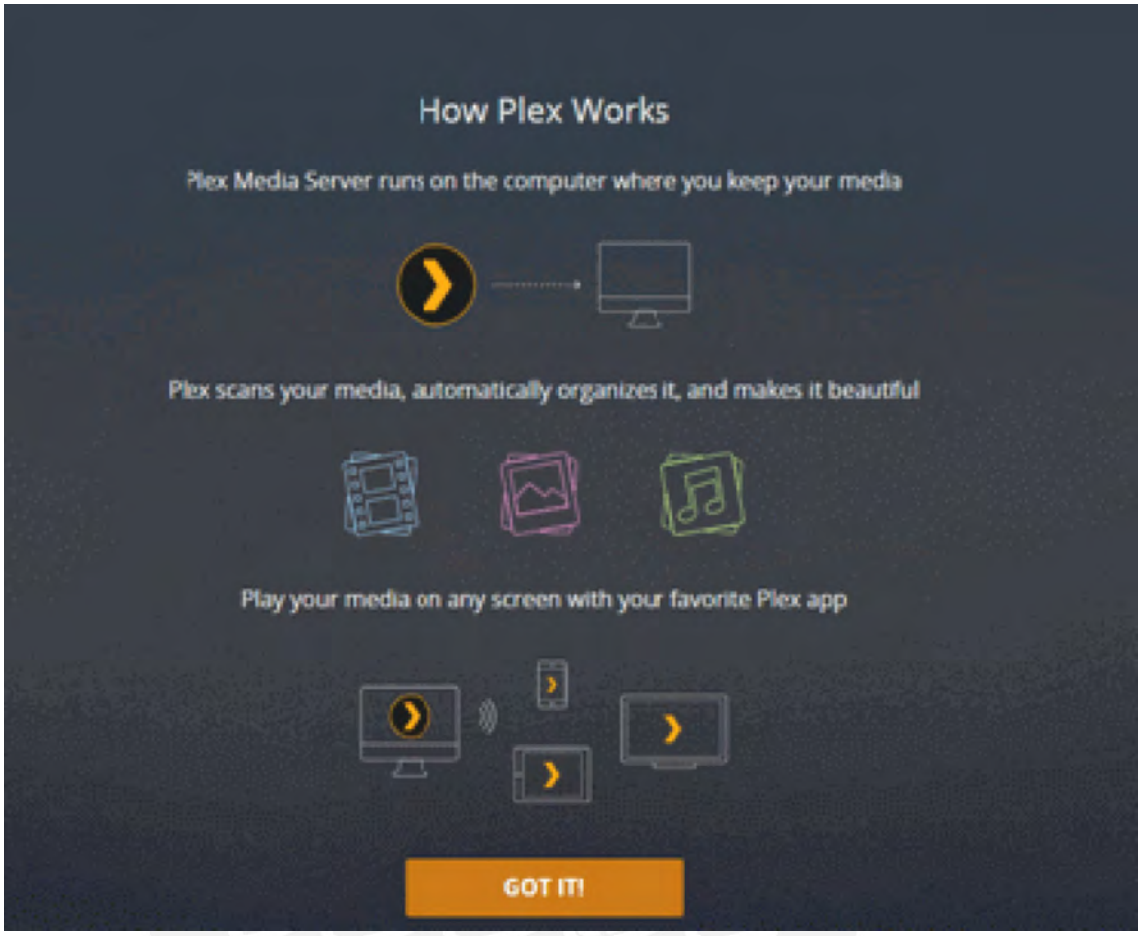


Figura 4.6: Funcionamiento de Plex Media Server [30]

La configuración del Plex Media Server realizada en las máquinas virtuales se define en la Figura 4.7, en donde se desplegó el Media Server y se incluyó archivos multimedia para su reproducción desde otra máquina virtual que funcione como cliente.

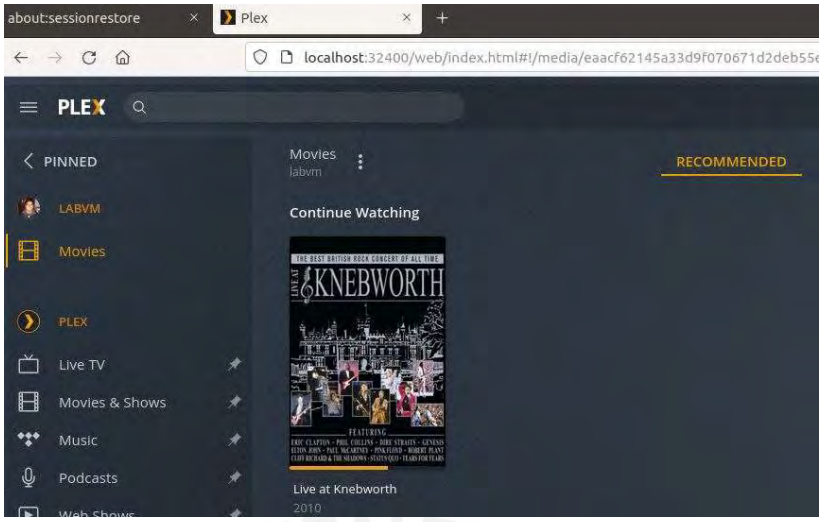


Figura 4.7: Configuración del Plex Media Server (Elaboración propia)

4.1.2.2. ZMQ en Python

En adición a lo comentado anteriormente, la transmisión de datos de video se complementa con la transmisión de data mediante el protocolo ZMQ [31] el cual está implementado en una librería de Python y que en conjunto con otras librerías de manipulación de datos de video como OpenCV (Figura 4.8) permiten que se pueda enviar frames de video desde una máquina virtual cliente hacia un servidor, generado el tráfico deseado. Este flujo se aprecia adecuadamente en la Figura 4.9.

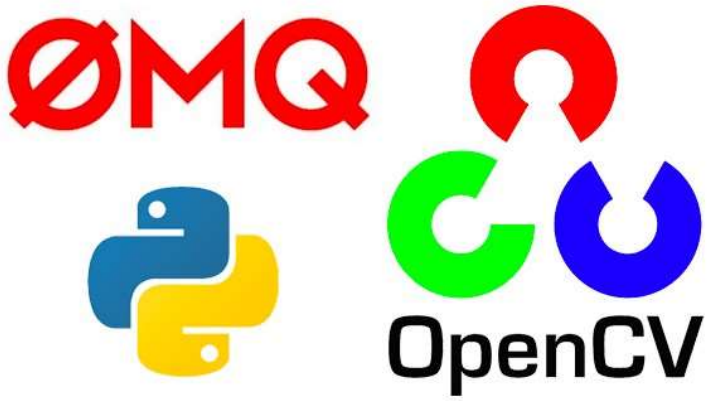


Figura 4.8: ZMQ en Python [31]

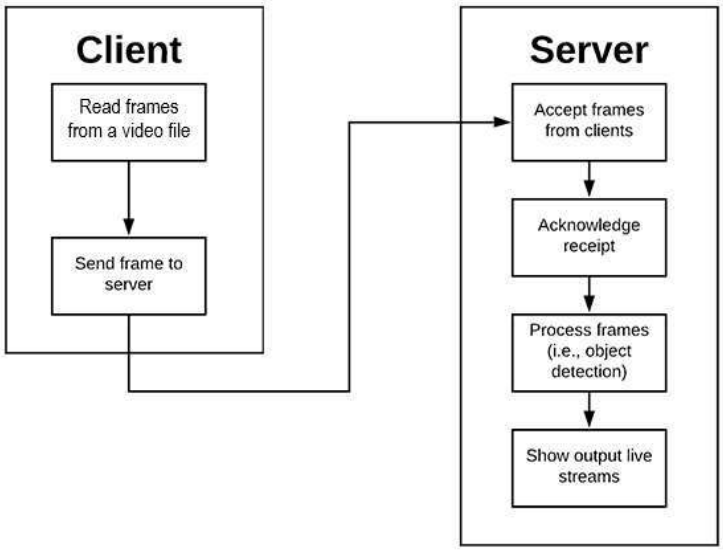


Figura 4.9: Generación de tráfico con ZMQ en Python. Adaptado de [31]

Respecto a la ejecución realizada tanto en el cliente como en el servidor de ZMQ, se procedió a correr los scripts correspondientes a la transmisión de datos desde un punto a otro de la red, obteniéndose correctamente la conexión, tal como se aprecia en las Figuras 4.10 y 4.11 donde se muestra la transmisión y recepción de data de video.

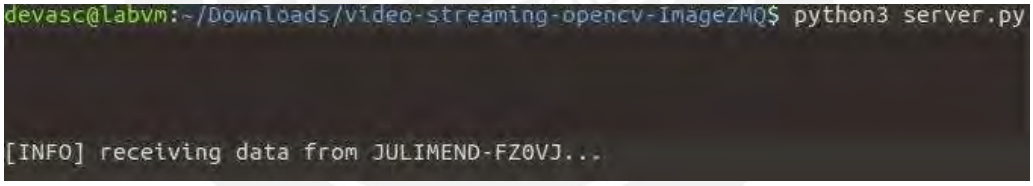


Figura 4.10: Ejecución del server ZMQ en la CLI (Elaboración propia)

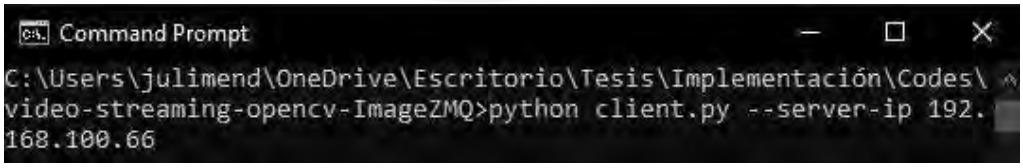


Figura 4.11: Ejecución del cliente ZMQ en la CLI (Elaboración propia)

4.1.3. Pruebas de throughput y latencia

4.1.3.1. Configuración de perfiladores Cbench e iPerf

Tal como se comentó en la sección 3.3 del Capítulo 3 de la presente tesis, se utilizarán perfiladores como Cbench enfocado principalmente en la determinación de métricas de red en

controladores SDN, así como iPerf el cual cuenta con parecidas prestaciones pero para cualquier tipo de red. En adición, de acuerdo a lo revisado en [32], dichos perfiladores deben configurarse en cada una de las máquinas virtuales desplegadas, tanto en un entorno local como del VNRT. Para ello se sigue la distribución presentada en la Figura 4.12 la cual aplica tanto para Cbench como para iPerf.

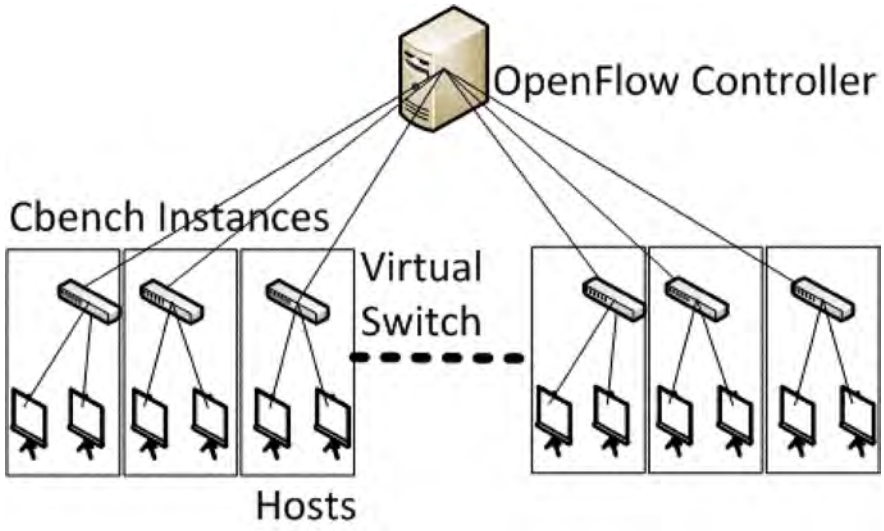


Figura 4.12: Configuración del perfilador Cbench en una red SDN [32]

Con respecto a la instalación de los perfiladores, para el caso de iPerf basta con realizar una actualización a los paquetes del sistema operativo, en este caso Linux, y ejecutar el comando **sudo yum install iperf -y**. Con respecto a Cbench, es necesario clonar los paquetes desde un repositorio de Github, tal como se aprecia en la Figura 4.13 donde se detallan los pasos de instalación necesarios.

```
$ git clone https://github.com/trema/cbench.git
$ cd cbench
$ bundle install --binstubs
```

Figura 4.13: Instalación del perfilador Cbench [33]

4.2. Resultados obtenidos

En la presente sección se listan los resultados obtenidos para cada una de las métricas de red definidas.

4.2.1. Latencia

Luego de ejecutar Plex Media Server se ejecutó el Controlador SDN Ryu y se procedió a ejecutar el perfilador Cbench, obteniendo lo mostrado en la Figura 4.14, los cuales están dentro de los rangos establecidos en la Tabla 3.1 de consideraciones de diseño del Wireless Controller y que concluyen que en la red se tiene una latencia de **10 ms**. Queda pendiente para trabajos futuros el realizar una mayor cantidad de pruebas presenciales con dispositivos inalámbricos a fin de complementar los presentes resultados.

```
devasc@labvm:~/Documents/cbench/bin$ cbench - localhost
cbench: controller benchmarking tool
   running in mode 'latency'
   connecting to controller at localhost:6633
   faking 16 switches offset 1 :: 16 tests each; 1000 ms per test
   with 100000 unique source MACs per switch
   learning destination mac addresses before the test
   starting test with 0 ms delay after features_reply
   ignoring first 1 "warmup" and last 0 "cooldown" loops
   connection delay of 0ms per 1 switch(es)
   debugging info is off
22:32:11.359 16  switches: flows/sec:  10  6  6  6  8  8  8  6  10  6  6  6  8  6  6  6   total = 0.111892 per ms
22:32:12.466 16  switches: flows/sec:  8  8  8  10  8  8  10  8  8  10  8  8  8  8  8  8   total = 0.133212 per ms
```

Figura 4.14: Resultados de las pruebas de latencia obtenidos por los perfiladores (Elaboración propia)

4.2.2. Throughput

Para la medición del throughput se ejecutó el programa de transmisión de data de video a través de ZMQ con Python junto con el Controlador SDN Ryu y se procedió a ejecutar el perfilador iPerf, obteniendo lo mostrado en la Figura 4.15, los cuales son limitados debido a que solo se han ejecutado en un entorno local. Para llevarlo a valores que se encuentren dentro de los rangos establecidos en la Tabla 3.1 de consideraciones de diseño del Wireless Controller es necesario extrapolar los resultados, lo cual se detallará en la siguiente sección. Al igual que en el caso de los resultados de latencia, es necesario complementar los presentes resultados con equipos inalámbricos y un diverso número de dispositivos conectados.

```
devasc@labvm:~/Downloads/video-streaming-opencv-ImageZMQ$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.100.66 port 5001 connected with 192.168.100.49 port 63359 (peer 74.21836.18765-unk)
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 0.0 sec  37.0 Bytes  63.5 Kbits/sec
```

Figura 4.15: Resultados de las pruebas de throughput obtenidos por los perfiladores (Elaboración propia)

4.3. Análisis de resultados

4.3.1. Validación del Wireless Controller diseñado

Luego de obtenidos los resultados presentados en la sección anterior se procede al análisis correspondiente, el cual requiere de una extrapolación del parámetro de throughput, tal como se describe a continuación:

63.5 Kbps x 1000 usuarios conectados x 1000 sesiones conectadas x 0.125= 8 Gbps aprox

Para el calculo anteriormente presente se considera una atenuación de una octava parte que implicaría las limitaciones de hardware de los equipos usados en el laboratorio GIRA. Todo ello lleva a lo presentado en la Tabla 4.1. Cabe resaltar que a la fecha quedan pruebas pendientes por realizar que complementarán el presente estudio.

Tabla 4.1: Tabla de validación del Wireless Controller diseñado (Elaboración propia)

Parámetro	**Perfilador**	**Legacy (referencia)**	**SDN (propuesta)**
Latencia (ms)	Cbench	20	10
Throughput (Gbps)	Iperf	40	8

De la Tabla 4.1 se tiene que se pudo obtener resultados mejores a los requeridos, pero que lamentablemente, debido a limitaciones de hardware, tal como fueron descritos en la Tabla 3.2 no se pudieron llegar a valores elevados de throughput.

Finalmente estos resultados comprueban la escalabilidad, programabilidad y versatilidad de este nuevo paradigma de redes y su beneficio de utilizarse en una red de campus académica como la de la PUCP . Asimismo, queda trabajo pendiente en el ámbito del paradigma SDN y su aplicación en redes inalámbricas.

Conclusiones

- El presente trabajo de tesis permitió el adecuado entendimiento del funcionamiento de los Wireless Controller convencionales, los algoritmos de enrutamiento dinámico utilizados en redes inalámbricas, así como la comprensión del paradigma SDN junto con los protocolos y funcionalidades de estos en las redes inalámbricas.

- Se definieron los criterios de diseño del Wireless Controller basado en SDN con base en los parámetros de los Wireless Controllers convencionales comerciales, así como la data obtenida de la infraestructura de red inalámbrica de la PUCP tales como cantidad de dispositivos de red conectados, throughput, latencia, tolerancia a fallas y seguridad; los cuales están plasmados en la Tabla 3.1.

- La implementación del Wireless Controller basado en SDN siguiendo con la topología de pruebas definida en la Figura 3.6 y desplegada en el VNRT del HPC Hitachi del laboratorio GIRA, en conjunto con perfiladores como Cbench e iPerf permitió medir métricas de latencia y throughput a partir del tráfico de video streaming generado mediante la plataforma de Plex Media Server, así como la transmisión de data de video en una red mediante ZMQ.

- Tal como se aprecia en la sección de Análisis de resultados del capítulo 4 de la presente tesis, se logró el resultado final de la tesis, verificando el objetivo principal de la tesis, así como la validación de los resultados en SDN , así como la escalabilidad, programabilidad y versatilidad de este nuevo paradigma de redes y sus ventajas al utilizarse en una red de campus académica como la de la PUCP.

Recomendaciones y trabajos futuros

- Se recomienda que para futuros desarrollos en el ámbito de Wireless Controllers basados en SDN se realicen una mayor cantidad de pruebas presenciales a fin de afianzar los resultados obtenidos en la presente tesis, que fueron limitados por el contexto de la pandemia del COVID-19.
- En conjunto con las métricas de la red evaluadas en la presente tesis: latencia y throughput, podrían considerarse el medir otro tipo de métricas como *jitter* o pérdida de paquetes, lo que complementaría el desarrollo del Wireless Controller basado en SDN desarrollado, otorgando una mayor visibilidad de lo que sucede en la red inalámbrica, así como la calidad de las conexiones de la misma.
- Si bien para el diseño de la topología se consideró el uso de un switch de firewall, el cual cuenta con configuraciones básicas de seguridad para la red simulada en el VNRT, podría incluirse en trabajos futuros aspectos de seguridad y autenticación, ello con miras a una integración con la red inalámbrica de la PUCP.
- Dado que el Wireless Controller se ejecuta dentro de una máquina virtual, futuras implementaciones podría considerar la posibilidad de desplegarlo en nubes públicas como Amazon Web Services, Microsoft Azure o Google Cloud Platform; así como incluir herramientas de analítica usando inteligencia artificial en lo que en la industria se conoce como ARM (Application Resource Management), consolidando al Wireless Controller desarrollado como un producto competitivo.

Referencias

[1] V. Jain, V. Yatri, Kanchan, and C. Kapoor, "Software defined networking: State-of-the-art," *Journal of High Speed Networks*, vol. 25, pp. 1–40, Enero 2019. doi: 10.3233/JHS-190601.

[2] J. Smith, J. Woodhams, and R. Marg, *Controller-Based Wireless LAN Fundamentals*. Indianapolis: Cisco Press, 2011.

[3] G. Cuba and J. Becerra, "Diseño e implementación de un controlador SDN/openflow para una red de campus académica," *Tesis de Licenciatura, Pontificia Universidad Católica del Perú*, Agosto 2016. Disponible en: `http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/7149`.

[4] PUCP, "Datos administrativos," 2019. [En línea] `https://www.pucp.edu.pe/la-universidad/nuestra-universidad/pucp-cifras/datos-administrativos/`. [Accedido: 4-may-2021].

[5] C. Quispe, "Diseño e implementación de un balanceador de carga para la optimización de los recursos de protección en una red enterprise mediante un banco de firewalls n: 1 controlado vía SDN," *Tesis de Licenciatura, Pontificia Universidad Católica del Perú*, Diciembre 2019. Disponible en: `http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/15579`.

[6] IBM, "SDN versus traditional networking explained," 2021. [En línea] `https://www.ibm.com/services/network/sdn-versus-traditional-networking`. [Accedido: 4-jun-2021].

[7] A. Tanenbaum, N. Feamster, and D. Wetherall, *Computer Networks , Global Edition.*, vol. Sixth Edition, Global Edition. Harlow: Pearson, 2021.

[8] A. Batiste Toryano, "Protocolos de encaminamiento en redes inalámbricas mesh: un estudio teórico y experimental," *Universitat Oberta de Catalunya*, Junio 2011. Disponible

en: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/8164/1/abatistet_TFM_0611.pdf.

[9] A.-h. Abdulaziz, E. Adedokun, and S. Man-Yahya, "Improved extended dijkstra's algorithm for software defined networks," *International Journal of Applied Information Systems*, vol. 12, pp. 22–26, Noviembre 2017. doi: 10.5120/ijais2017451714.

[10] Cisco, "Cisco Catalyst 9800-40 Wireless Controller Data Sheet," 2021. [En línea] https://www.cisco.com/c/en/us/products/collateral/wireless/catalyst-9800-series-wireless-controllers/nb-06-cat9800-wirel-cont-data-sheet-ctp-en.html#Details. [Accedido: 30-jun-2021].

[11] Aruba, "Aruba 7200 Series Mobility Controllers," 2021. [En línea] https://https://www.arubanetworks.com/products/wireless/gateways-and-controllers/7200-series/. [Accedido: 30-jun-2021].

[12] Commscope, "SZ144 — RUCKUS SmartZone 144," 2021. [En línea] https://www.commscope.com/product-type/enterprise-networking/control-management/network-controllers/smartzone-144/. [Accedido: 30-jun-2021].

[13] P. Göransson and C. Black, *Software Defined Networks: A Comprehensive Approach*. Waltham: Elsevier Inc., 2014.

[14] Z. Chen, Z. Luo, X. Duan, and L. Zhang, "Terminal handover in software-defined WLANs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, p. 68, Marzo 2020. doi: 10.1186/s13638-020-01681-w.

[15] D. Comer, *Computer Networks and Internets, Global Edition.*, vol. Sixth Edition, Global Edition. Harlow: Pearson, 2015.

[16] E. Amiri, E. Alizadeh, and M. Rezvani, "Controller selection in software defined networks using best-worst multi-criteria decision-making," *Bulletin of Electrical Engineering and Informatics*, vol. 9, Abril 2020.

[17] A. Dvir, Y. Haddad, and A. Zilberman, "The controller placement problem for wireless SDN," *Wireless Networks*, vol. 25, pp. 4963–4978, Noviembre 2019. doi: 10.1007/s11276-019-02077-5.

[18] Mininet, "Mininet," 2021. [En línea] http://mininet.org/news/. [Accedido: 10-jun-2021].

[19] Mininet-WiFi, "Mininet-WiFi," 2021. [En línea] https://mininet-wifi.github.io/. [Accedido: 30-jun-2021].

[20] A. Chaurasia, S. N. Mishra, and S. Chinara, "Performance evaluation of software-defined wireless networks in it-sdn and mininet-wifi," in *2019 1st International Conference on Advances in Information Technology (ICAIT)*, pp. 315–319, 2019.

[21] A. Merino and C. Quispe, *Manual de usuario del emulador de Redes: VNRT*. AINET Solutions S.A.C, Julio 2018.

[22] Hitachi, "HPC Hitachi - Datasheet," 2018. [En línea] https://www.hitachi.co.jp/Prod/comp/hpc/pdf/sr11000e_ka_z.pdf. [Accedido: 13-oct-2021].

[23] G. Espinoza, "Diseño e implementación de un sistema de control de acceso de usuarios inalámbricos integrado al controlador SDN PUCPLight para una red de campus académica," *Tesis de Licenciatura, Pontificia Universidad Católica del Perú*, Diciembre 2017.

[24] G. Cuba, J. M. Becerra, G. Bartra, and C. Santivanez, "PUCPLight: A SDN/OpenFlow controller for an academic campus network," *2016 IEEE ANDESCON*, pp. 1–4, Octubre 2016.

[25] T. D. Nadeau and K. Gray, *SDN : Softare Defined Networks*. Sebastopol: O'Reilly, 2013.

[26] Ryu, "Ryu resources," 2021. [En línea] https://ryu-sdn.org/resources.html. [Accedido: 01-oct-2021].

[27] OpenDaylight, "Opendaylight documentation," 2021. [En línea] https://docs.opendaylight.org/en/latest/index.html#opendaylight-contributor-guides. [Accedido: 01-oct-2021].

[28] M. Obadia, M. Bouet, J. Leguay, K. Phemius, and L. Iannone, "Failover mechanisms for distributed SDN controllers," *2014 International Conference and Workshop on the Network of the Future (NOF)*, 2014.

[29] L. D. Pinney, "TP-link TL-WDR3600 (n600)," 2021. [En línea] https://openwrt.org/toh/tp-link/tl-wdr3600_v1. [Accedido: 23-nov-2021].

[30] Plex, “Stream movies free online | free TV and movies with plex,” 2021. [En línea] https://www.plex.tv/. [Accedido: 13-nov-2021].

[31] PyImageSearch, “Live video streaming over network with OpenCV and ImageZMQ,” 2019. [En línea] https://www.pyimagesearch.com/2019/04/15/live-video-streaming-over-network-with-opencv-and-imagezmq/. [Accedido: 13-nov-2021].

[32] Z. Khattak, M. Awais, and A. Iqbal, “Performance evaluation of OpenDaylight SDN controller,” vol. 2015, pp. 671–676, Abril 2015.

[33] Trema, “cbench,” 2021. [En línea] https://github.com/trema/cbench. [Accedido: 23-nov-2021].