UNIVERSIDAD DE GUAYAQUIL

FACULTAD DE CIENCIAS MATEMATICAS Y FISICAS

CARRERA DE INGENIERIA EN NETWORKING Y TELECOMUNICACIONES

"DISEÑO Y SIMULACION DE UN PROTOTIPO DE RED DEFINIDA POR SOFTWARE (SDN) USANDO EL PROTOCOLO OPENFLOW"

PROYECTO DE TITULACIÓN

Previa a la obtención del Título de:

INGENIERO EN NETWORKING Y TELECOMUNICACIONES

AUTOR: VALENCIA ECHEVERRIA CHRISTIAN DE JESÚS

TUTOR: MG. JOSÉ COELLAR SOLÓRZANO

GUAYAQUIL – ECUADOR

2015

REPOSITORIO NACIONAL EN CIENCIAS Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS

TÍTULO: "DISEÑO Y SIMULACION DE UN PROTOTIPO DE RED DEFINIDA POR SOFTWARE (SDN) USANDO EL PROTOCOLO OPENFLOW.**"**

	REVISORES:
INSTITUCIÓN: **Universidad de Guayaquil**	**FACULTAD:** **Ciencias Matemáticas y Físicas**
CARRERA: **Ingeniería en Networking y Telecomunicaciones**	
FECHA DE PUBLICACIÓN:	**N° DE PÁGS.: 99**
ÁREA TEMÁTICA: Tecnológico Redes	
PALABRAS CLAVES: *redes software, SDN, Openflow.*	

RESUMEN: Actualmente, se está viviendo una época de revolución a nivel de las redes de telecomunicaciones, ya que con la introducción de nuevos tipos de redes de velocidades inalcanzables hace unos pocos años se han creado nuevos usos para las redes de datos, y éstas también han evolucionado para estar a la par de los avances tecnológicos. Uno de los más recientes avances son las redes definidas por software (SDN por sus siglas en inglés), las cuales utilizan un software específico que controla al hardware para la administración y mantenimiento de la red, cambiando el paradigma actual donde el hardware es la parte central en las redes de datos. Este estudio tiene como objetivo un prototipo de una red definida por software, a través del uso de herramientas de virtualización conocidas, en la cual se pueda mostrar los elementos centrales en este tipo de redes, su funcionalidad teórica y su demostración práctica en una red estándar, basándose en información del protocolo Openflow. Así mismo, mediante este estudio se podrá emitir comentarios sobre las ventajas y desventajas de los diversos elementos que conforman una red definida por software y su posible aplicación en una red típica empresarial, como reemplazo u optimización de una red existente, proveyendo a los administradores de redes un estudio actualizado de la tecnología revisada en ese proyecto de titulación.

N° DE REGISTRO(en base de datos):	**N° DE CLASIFICACIÓN: Nº**	
DIRECCIÓN URL (tesis en la web):		
ADJUNTO PDF	**SI** [X]	**NO** [ ]
CONTACTO CON AUTOR: VALENCIA ECHEVERRIA CHRISTIAN DE JESÚS	**Teléfono: 0996084491**	**E-mail: cvalenciaec@gmail.com**
CONTACTO DE LA INSTITUCIÓN **Carrera de Ingeniería en Networking y Telecomunicaciones**	**Nombre:** **Ab. Juan Chávez**	
	Teléfono: **2307729**	

APROBACION DEL TUTOR

En mi calidad de Tutor del trabajo de investigación, "DISEÑO Y SIMULACION DE UN PROTOTIPO DE RED DEFINIDA POR SOFTWARE (SDN) USANDO EL PROTOCOLO OPENFLOW" elaborado por el Sr. VALENCIA ECHEVERRIA CHRISTIAN DE JESÚS, Alumno no titulado de la Carrera de Ingeniería en Networking y Telecomunicaciones, Facultad de Ciencias Matemáticas y Físicas de la Universidad de Guayaquil, previo a la obtención del Título de Ingeniero en Networking y Telecomunicaciones, me permito declarar que luego de haber orientado, estudiado y revisado, la Apruebo en todas sus partes.

Atentamente

Mg. José Coellar Solórzano

TUTOR

DEDICATORIA

Dedico el presente trabajo a mi papá, que no me alcanzó a ver lograr este objetivo, pero estoy seguro que se sentiría orgulloso de haberlo hecho… gracias papá…

AGRADECIMIENTO

A Dios por nunca desamparar a sus hijos cuando peor se ponen las cosas.

A mis padres que me ayudaron tanto para lograr este sueño de ser ingeniero.

A mi esposa y su familia que de una u otra manera me ayudaron con lo que necesitaba en el transcurso de mis estudios.

A mi tutor de tesis, por darme todo el apoyo necesario para culminar este difícil reto.

TRIBUNAL DEL PROYECTO DE TITULACIÓN

Ing. Eduardo Santos Baquerizo, M.Sc.
DECANO DE LA FACULTAD
CIENCIAS MATEMATICAS Y
FISICAS

Ing. Harry Luna Aveiga, M.Sc
DIRECTOR
CARRERA DE INGENIERÍA EN
NETWORKING Y
TELECOMUNICACIONES

Ing. Christian Antón Cedeño
PROFESOR DEL ÁREA -
TRIBUNAL

Msc. Kléber Rendón Buros
PROFESOR DEL ÁREA -
TRIBUNAL

Mg. José Coellar Solórzano
DIRECTOR DEL PROYECTO DE TITULACIÓN

Ab. Juan Chávez A.
SECRETARIO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Titulación, me corresponden exclusivamente; y el patrimonio de la misma a la UNIVERSIDAD DE GUAYAQUIL"

_____

VALENCIA ECHEVERRÍA CHRISTIAN DE JESÚS

.

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS

CARRERA DE INGENIERIA EN NETWORKING Y TELECOMUNICACIONES

"DISEÑO Y SIMULACION DE UN PROTOTIPO DE RED DEFINIDA POR SOFTWARE (SDN) USANDO EL PROTOCOLO OPENFLOW"

Proyecto de Titulación que se presenta como requisito para optar por el

título de INGENIERO EN NETWORKING Y TELECOMUNICACIONES

Autor: VALENCIA ECHEVERRÍA CHRISTIAN DE JESÚS

C.I. 0926626284

Tutor: MG. JOSÉ COELLAR SOLÓRZANO

Guayaquil, diciembre del 2015

CERTIFICADO DE ACEPTACIÓN DEL TUTOR

En mi calidad de Tutor de Proyecto de Titulación, nombrado por el Consejo Directivo de la Facultad de Ciencias Matemáticas y Físicas de la Universidad de Guayaquil.

CERTIFICO:

Que he analizado el Proyecto de Titulación presentado por el estudiante VALENCIA ECHEVERRÍA CHRISTIAN DE JESÚS, como requisito previo para optar por el título de Ingeniero en Networking y Telecomunicaciones cuyo problema es:

DISEÑO Y SIMULACION DE UN PROTOTIPO DE RED DEFINIDA POR SOFTWARE (SDN) USANDO EL PROTOCOLO OPENFLOW

Considero aprobado el trabajo en su totalidad.

Presentado por:

VALENCIA ECHEVERRIA CHRISTIAN DE JESÚS
Cédula de ciudadanía N° 0926626284

Tutor: Mg. José Coellar Solórzano

Guayaquil, diciembre del 2015

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS
CARRERA DE INGENIERIA EN NETWORKING Y TELECOMUNICACIONES

Autorización para Publicación de Proyecto de Titulación en Formato Digital

1. Identificación del Proyecto de Titulación

Nombre Alumno: Valencia Echeverría Christian de Jesús	
Dirección: Cdla. Floresta 2 Mz 121 Villa 2	
Teléfono: 0995660522	**E-mail:** cvalenciaec@gmail.com

Facultad: Ciencias Matemáticas y Físicas
Carrera: Ingeniería en Networking y Telecomunicaciones
Título al que opta: Ingeniero en Networking y Telecomunicaciones
Profesor guía: Mg. José Coellar Solórzano

Título del Proyecto de titulación: DISEÑO Y SIMULACION DE UN PROTOTIPO DE RED DEFINIDA POR SOFTWARE (SDN) USANDO EL PROTOCOLO OPENFLOW

Tema del Proyecto de Titulación: Redes, SDN, Openflow, Software

2. Autorización de Publicación de Versión Electrónica del Proyecto de Titulación

A través de este medio autorizo a la Biblioteca de la Universidad de Guayaquil y a la Facultad de Ciencias Matemáticas y Físicas a publicar la versión electrónica de este Proyecto de titulación.

Publicación electrónica:

Inmediata	**X**	Después de 1 año	

Firma Alumno:

3. Forma de envío:

El texto del proyecto de titulación debe ser enviado en formato Word, como archivo .Doc. O .RTF y .Puf para PC. Las imágenes que la acompañen pueden ser: .gif, .jpg o .TIFF.

DVDROM		CDROM	X

INDICE GENERAL

ÍNDICE DE CUADROS

Pag.

ÍNDICE DE GRÁFICOS

Pág.

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS
CARRERA DE INGENIERÍA EN NETWORKING Y TELECOMUNICACIONES

"Diseño y simulación de un prototipo De red definida por software (SDN) usando el protocolo Openflow."

Autor: Christian Valencia E.
Tutor: Ing. José Coellar Solórzano

RESUMEN

Actualmente, se está viviendo una época de revolución a nivel de las redes de telecomunicaciones, ya que con la introducción de nuevos tipos de redes de velocidades inalcanzables hace unos pocos años se han creado nuevos usos para las redes de datos, y éstas también han evolucionado para estar a la par de los avances tecnológicos. Uno de los más recientes avances son las redes definidas por software (SDN por sus siglas en inglés), las cuales utilizan un software específico que controla al hardware para la administración y mantenimiento de la red, cambiando el paradigma actual donde el hardware es la parte central en las redes de datos. Este estudio tiene como objetivo un prototipo de una red definida por software, a través del uso de herramientas de virtualización conocidas, en la cual se pueda mostrar los elementos centrales en este tipo de redes, su funcionalidad teórica y su demostración práctica en una red estándar, basándose en información del protocolo Openflow 1.5. Así mismo, mediante este estudio se podrá emitir comentarios sobre las ventajas y desventajas de los diversos elementos que conforman una red definida por software y su posible aplicación en una red típica empresarial, como reemplazo u optimización de una red existente, proveyendo a los administradores de redes un estudio actualizado de la tecnología revisada en ese proyecto de titulación.

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS
CARRERA DE INGENIERÍA EN NETWORKING Y TELECOMUNICACIONES

Diseño y simulación de un prototipo
De red definida por software (SDN) usando el protocolo Openflow.

Autor: Christian Valencia E.
Tutor: Ing. José Coellar.

Abstract

In present days, there's a revolution on telecommunication industry, due to the introduction of new networks with very high capacity, giving a second live to data networks with new uses as streaming, telecommuting, and these advancements are not getting bypassed by networks technologies. One of the most recent technologies that appeared are Software Defined Networks, which changes completely the way that networks are being watches for nearly 40 years. This project has as goal to produce a prototype of a SDN network that can show their benefits and changes in front of conventional networks, its theory and experimental testing in a prototype using the protocol Openflow. Also, this project will help to expand the mindshare of SDN inside students, academics, etc. and they will be able to understand the uses and benefits of a SDN network and their possible replacement of physical networks over datacenters across the country. Another goal of this project is serve as an updated information about SDN networks to sysadmins that manage large networks around the globe.

INTRODUCCION

El presente Proyecto de Titulación se encuentra divido en cuatro capítulos: el primer capítulo trata sobre el problema que busca resolver este proyecto, en el segundo capítulo se realiza una revisión del fundamento teórico de las SDN, el tercer capítulo corresponde a la implementación del prototipo de una red SDN empleando switches basados en software y el cuarto capítulo se muestra los criterios que deben emplearse para la aceptación de este proyecto.

El primer capítulo del Proyecto formula la base del proyecto, los objetivos y el alcance del mismo, todo esto fundamentado correctamente en la necesidad a cubrir por parte de este proyecto y su estructura general.

El segundo capítulo del Proyecto busca establecer un punto de inicio, donde se determina los antecedentes de las redes SDN desde su concepción y las limitantes iniciales que se presentaban en su génesis, luego las diferentes evoluciones que se dieron a lo largo de los años hasta llegar a los servicios actuales que se proveen a través de estas redes.

En el tercer capítulo se realiza la implementación del prototipo de una red SDN, se mencionan el servidor controlador a usarse en el prototipo, todo esto basado en la metodología escogida para realizar este tipo de proyectos.

En el cuarto capítulo se explica el método del criterio de aceptación del proyecto por el cual se rige este proyecto.

En los anexos encontramos los elementos complementarios que se usaron en el proyecto y se adjunta el CD correspondiente, donde se encuentra el diseño del proyecto y las herramientas usadas en el mismo.

CAPÍTULO I

EL PROBLEMA

PLANTEAMIENTO DEL PROBLEMA

Ubicación del Problema en un Contexto

Actualmente, las redes de datos son participantes pasivos en la transmisión de información y generalmente requiere mucha experiencia para su administración, lo cual impide que el mantenimiento de las mismas sea más dinámico y acorde a las necesidades inmediatas de la empresa.

En este escenario, las redes definidas por software entran con nuevos paradigmas para masificar la administración y mantenimiento de las redes de datos, ya que incluyen conocimientos adquiridos a nivel de software y potenciación de los conceptos ya estudiados a lo largo de la evolución de las redes de datos.

Además, dado que este tipo de redes son relativamente noveles, es necesario apuntar los beneficios y desventajas para desmitificar el uso de este tipo de redes en ambientes de producción, para poder tomar ventaja de todos los avances tecnológicos que se muestran en estas tecnologías.

Es de notar que el comportamiento de las redes de datos no ha variado desde su creación en los años 70 en los laboratorios de ARPA en los Estados Unidos, sólo se han introducido mejoras en los diseños de redes,

por lo que cualquier cambio en los paradigmas establecidos genera inicialmente dudas y recelos sobre su utilidad y futuras aplicaciones.

Situación Conflicto Nudos Críticos

Las redes de computadoras se mantuvieron como espectadores de los grandes cambios de la tecnología informática, siempre se mejoraban el hardware y el software que los controlaba, pero se mantuvieron los mismos esquemas de diseño de red (cliente-servidor, estrella, etc.).

Este mismo esquema se replicó en los ambientes empresariales por muchos años, creando redes donde los Departamentos de Infraestructura y Redes requerían de colaboradores con conocimientos vastos de hardware y software de red, esto aún sin contar que la dificultad de la administración de estas redes se incrementa exponencialmente de acuerdo al tamaño de las mismas, requiriendo incluso mayor personal humano para su administración.

Además, en el plano empresarial, los costos de las implementaciones de IT, ya sean éstas nuevos proyectos o mejoras a lo existente actualmente, son muy seguidos por los departamentos financieros de las mismas; es común encontrar empresas con una inversión no adecuada en TI y aún así

requieren de servicios mucho más valorados que los implementados, creando problemas a los encargados de las áreas tecnológicas.

En las empresas que tienen como fuerte las telecomunicaciones, la inversión en TI suele ser más generosa que en los otros sectores de la industria, pero esto va unido también a que, generalmente, estas empresas implementan las nuevas tecnologías que van apareciendo en el campo, lo cual son generalmente apuestas a productos no maduros en su totalidad y que requieren inversión de dinero y tiempo para solventar inconvenientes que aparecen con este tipo de productos noveles.

CAUSAS Y CONSECUENCIAS DEL PROBLEMA

CUADRO # 1 Causas y consecuencias referentes al problema

Causas	**Consecuencias**
Falta de presupuesto para mantener la infraestructura de red actual.	Reduce la posibilidad de mejoras en la red de la empresa.
Altos costos del hardware propietario administrable.	Extender el uso del hardware actual más allá de su vida útil.
Falta de espacio físico para nuevos equipos de red.	Incrementa costos destinados para proyectos de TI.
Altos costos del personal humano administrativo de la red.	Incrementa la inversión mensual en los recursos humanos de TI.
Falta de interés en las nuevas tecnologías que implican a las redes.	Riesgo de quedar vulnerables a ataques informáticos conocidos.

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Delimitación del Problema

Campo: Tecnologías de la Información.

Área: Infraestructura de Red.

Aspecto: Software

Tema: Diseño y simulación de un prototipo de red definida por software (SDN) usando el protocolo Openflow.

Formulación del Problema

¿Incrementaría el uso de las redes definidas por software en las redes corporativas locales, si se realiza un diseño con simulación, mostrando las características y beneficios de esta tecnología?

Evaluación del Problema

Examinando la problemática obtenemos los siguientes aspectos generales a evaluar:

Delimitado: Este problema se presenta en los departamentos de Infraestructura de las empresas, que poseen redes corporativas locales que necesitan mayor agilidad para poder proveer servicios de red a sus usuarios.

Relevante: Importante, ya que de extenderse su utilidad en las redes corporativas, modificaría totalmente la manera de administrar redes de datos, generando cambios sustanciales en todos los ámbitos importantes de la empresa, pasando por el aspecto económico hasta el humano.

Factible: Dada las herramientas actuales que provee los servicios de virtualización, es posible implementar diseños y simulaciones de estas

tecnologías, que podrían ser extrapolables a ambientes de producción con una correcta planificación y ejecución del proyecto.

Original: Dada la novedad de este tipo de tecnologías en el ambiente local, es loable decir que es un proyecto original y actualizado en su tipo.

Variables: Las variables de este proyecto pueden ser:

Económicas: el costo del proyecto depende del tamaño de la implementación, en el caso de un prototipo los costos son bajos mientras que llevar a un ambiente de producción puede requerir gran inversión económica.

Físicas: Es posible adaptar este tipo de tecnologías al espacio físico disponible para TI, e incluso reusar los elementos físicos compartidos entre una red tradicional y una red definida por software.

Identifica los productos esperados: Resulta útil la implementación de un prototipo de este tipo de redes, el cual puede proveer recomendaciones y conclusiones al público destinado sobre estas tecnologías.

OBJETIVOS

OBJETIVO GENERAL

- Diseñar y simular una red definida por software orientada para una evaluación actualizada de los beneficios y desventajas de este tipo de redes.

.

OBJETIVOS ESPECÍFICOS

- Crear un diseño sostenible de una red definida por software, usando las herramientas de red que componen Openflow.
- Demostrar con una simulación la parte operativa de una red definida por software, basado en el diseño creado.
- Describir las tecnologías utilizadas en el diseño y simulación para su futuro uso en el ambiente académico.

ALCANCES DEL PROBLEMA

El presente proyecto “Diseño y simulación de un prototipo de red definida por software (SDN) usando el protocolo Openflow”, contará de los siguientes alcances:

- Un diseño de red virtual de la red SDN prototipo en la herramienta gráfica MiniEdit.
- Pruebas de conectividad el funcionamiento de la red SDN prototipo.

- Revisión del flujo de paquetes de datos que forman parte de Openflow en la herramienta gráfica Wireshark.

JUSTIFICACIÓN E IMPORTANCIA

Implementar este prototipo de una red definida por software nos ayudará a difundir los conceptos básicos de este tipo de redes en los ambientes académicos y empresariales, donde se conoce poco o nada de este tipo de redes; además promover las ventajas que proporcionan estas redes a las redes corporativas y a sus usuarios finales.

Cabe recalcar que es importante realizar este tipo de proyectos sobre nuevas tecnologías, ya que éstas se encuentran en constante avance y cuando se realizan estudios generalmente quedan obsoletos al corto tiempo, ya que la tecnología evoluciona rápidamente y los tiempos de generación de documentación científica fiable no está acorde al ritmo del desarrollo de la misma.

METODOLOGIA DEL PROYECTO

La metodología a usar para este proyecto es la llamada “PPDIOO (Prepare, Plan, Design, Implement, Operate and Optimize)”. Esta metodología es la más indicada para este tipo de proyectos, donde se realizan diseños originales de red para pruebas de concepto de productos noveles. En el capítulo 3 del presente documento encontraremos más detalles del uso de esta metodología para este proyecto.

CAPÍTULO II

MARCO TEORICO

ANTECEDENTES DEL ESTUDIO

Estudios iniciales: Redes activas

Las redes de datos de computadoras han evolucionado mucho en los últimos 20 años, los cuales han permitido grandes avances en los campos tecnológicos donde han sido aplicadas, un ejemplo de ello, es que, cuando la red Internet tomó vuelo, a mediados de los 90's, las aplicaciones para lo cual se usaba ya habían superado el consumo de datos de las primeras aplicaciones de trasferencia de datos y correo electrónico que usaban los científicos; lo cual creó expectativa por parte de los mismos para crear o innovar nuevos protocolos de red para soportar las nuevas fuentes de consumos de datos. Este proceso de creación de nuevos protocolos de red era avanzado en laboratorios de pruebas e implementado en simulaciones de redes grandes, pero su envío a la IETF generalmente demoraba mucho en revisarse e implementarse.

Debido a esto, investigadores empezaron a intentar "controlar" la red por medio de la programación; esto no es posible en la redes de datos convencionales, por lo que se definió un nuevo concepto llamado *redes activas*, lo cual manifestaba que la red física es un recurso más, que puede ser manipulado a nivel de CPU, RAM, paquetes, almacenamiento, etc.; todo esto mediante una interface especial que aplica los "controles" a la red de datos.

Este tipo de redes se manifestaron en dos formas:

- El modelo cápsula
- El modelo enrutador/commutador programable

Cada forma tenía sus ventajas, mientras el modelo cápsula tenía como objetivo enviar junto a los paquetes, modificándolos, las órdenes a ser ejecutadas en los enrutadores/conmutadores remotos, el modelo enrutador/conmutador programable buscaba ejecutar las ordenes directamente en estos equipos, sin modificar los paquetes.

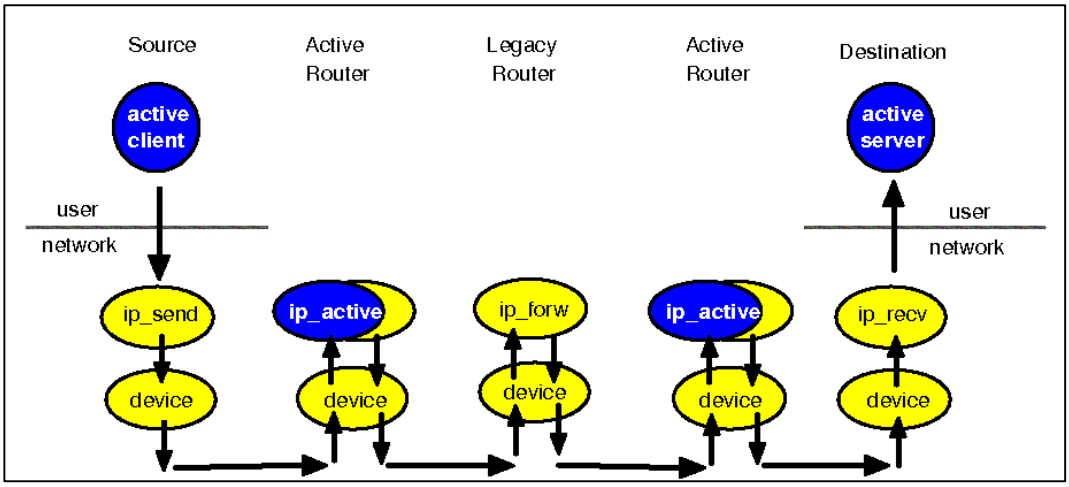


GRÁFICO # 1 Procesamiento en los nodos de una red activa.
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Estas tempranas maneras de intentar controlar la red, estuvieron motivadas en parte, por dos corrientes que todavía se ven ahora, el “empuje tecnológico” y la “tracción del usuario”, esto es, las necesidades de recursos en una red de datos, requerida por los elementos de red “usuarios” de la misma, como cortafuegos, proxis, Gateway, etc., y estos habilitados

por la tecnología necesaria para alcanzar esos objetivos de computación requeridos en la red.

Las redes activas contribuyeron con 3 aspectos importantes a lo que hoy conocemos como SDN:

- Introducir la programación a las redes
- Virtualización de redes
- Una arquitectura estandarizada para manejar las redes programables.

2.- Separación de planos: Datos, Control y Gestión

Todo dispositivo que maneja paquetes de datos de redes, posee tres planos, los cuales se diferencian por las funciones que realizan cada uno:

- Plano de Datos
- Plano de Control
- Plano de Gestión

Plano de Datos

Este plano es la sección de los dispositivos de red que se encarga de receptar y enviar los paquetes a través de los medios conectados al mismo, así como tomar decisiones simples sobre el enrutamiento de paquetes, siempre y cuando la información que recibe en los puertos sea completa y confiable.

Plano de Control

El plano de control se encarga de mantener la tabla de enrutamiento actualizada para el plano de datos, ya que las decisiones complejas de enrutamiento deben realizarse en el plano de control, y es donde se procesan los diferentes protocolos de red que son recibidos a través del plano de datos, es decir, es donde ocurre la “inteligencia” del dispositivo de red.

Plano de Gestión

Es el plano encargado directamente de la gestión del dispositivo de red, por lo que el monitoreo y la configuración de los planos de control y datos se realiza en este nivel, generalmente sólo los administradores de red tienen acceso a este plano a través de una red específica de gestión.

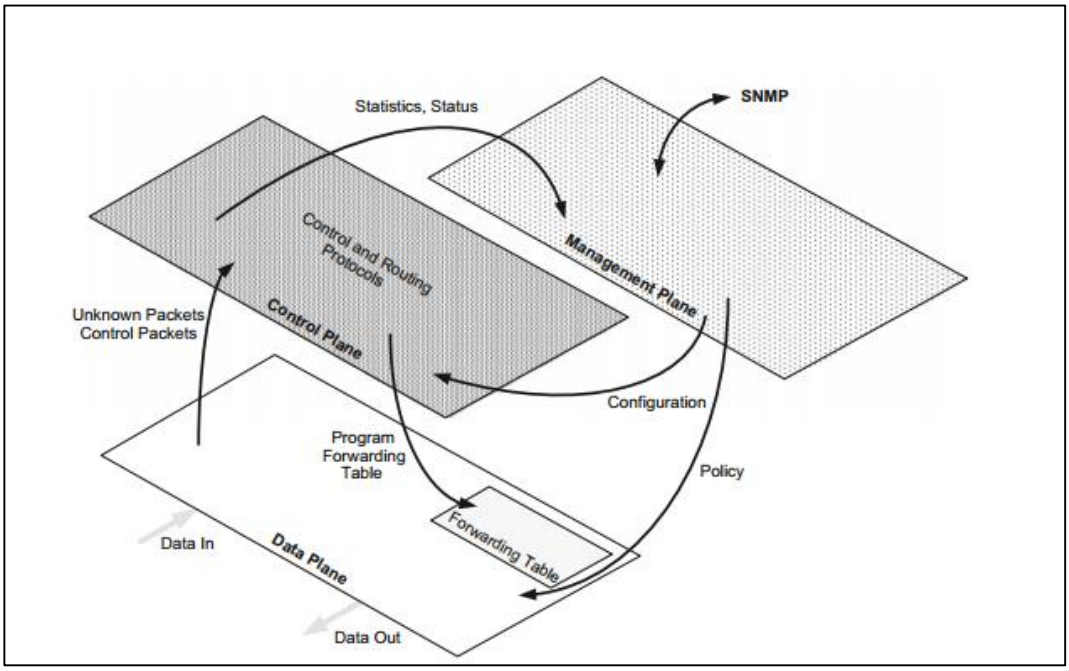


GRÁFICO # 2 Interacción de los planos en un dispositivo de red

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Esta separación de los planos impulsó la creación de 2 nuevas vías de investigación:

- Una interfaz libre entre los planos de control y datos, como la interfaz ForCES (Forwarding and Control Element Separation) y la interfaz Netlink implementada en el kernel de Linux.
- Control lógico central de la red, mostrado por las plataformas RCP (Routing Control Platform) y SoftRouter, y el protocolo PCE (Path Computation Element)

Este modelo de tratamiento de los paquetes en una red de datos, separando por funciones los procesos internos realizados en los dispositivos de red, lleva siendo aplicado más de una década, por lo que es posible que la gran mayoría de los dispositivos use este modelo.

La separación de los planos

Antes de las SDN, la separación de los planos sólo era realizada como una manera de establecer recursos específicos para cada función en el dispositivo de red, y no se concebía separar los planos más allá de unos cuantos centímetros dentro del chasis de los enrutadores/conmutadores.

Con la aparición en el mercado de soluciones de virtualización de sistemas operativos, tal como VMware, los centros de cómputos pudieron explotar el máximo de capacidad disponible en procesamiento y almacenamiento de

información, incrementando así mismo la cantidad de datos trasladada dentro y fuera del mismo. Los equipos de red pronto se vieron congestionados con la cantidad de información que tenían que procesar e intercambiar con otros equipos, creando estrés en los planos de control y datos. Este tipo de problemas generalmente se solucionan con la adquisición de más equipos con mayor capacidad de procesamiento de datos de red, lo cual no es una opción financiera viable en los centros de cómputos que manejar cantidades ingentes de información y para los cuales, la tasa de crecimiento de información trasladada a través de sus redes crece exponencialmente cada año.

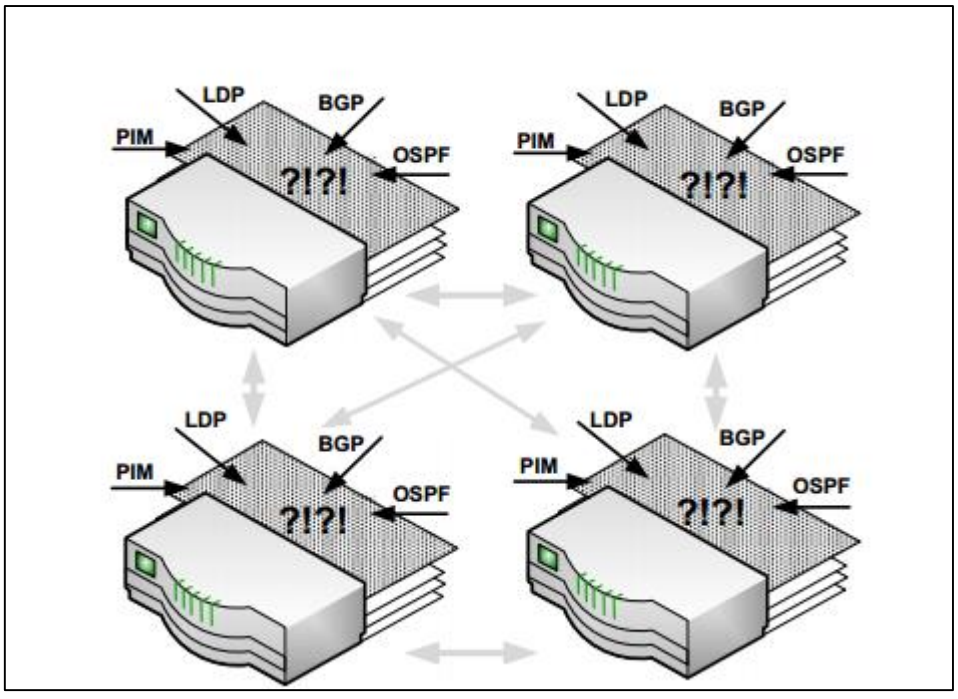


GRAFICO # 3 Tratamiento de protocolos distribuidos entre dispositivos de red.
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Además, la inserción o remoción de elementos de red en este campo, genera interrupción mínima de los servicios prestados en la misma, así como añade complejidad al manejo de la red, algo tan evitado en los centros de cómputos.

Las redes SDN llegan a proponer un nuevo uso de la separación de los planos en los dispositivos de red; no es necesario mantener los planos en el mismo chasis para tener el mismo procesamiento de datos y se obtienen algunas ventajas al separar completamente los planos, las cuales explicamos a continuación:

Escalabilidad: Los planos, al estar separados, pueden ser dimensionados de manera independiente para poder soportar una mayor cantidad de datos a lo largo del tiempo, se eliminan los límites que podrían imponer un plano sobre el otro al momento de tratar de escalar los dispositivos de red.

Evolución: Cada plano maneja tecnologías que evolucionan a velocidades diferentes, en este caso, a mayor evolución de la tecnología, requería reemplazar los dispositivos de red sólo para actualizar los elementos del plano requerido, desperdiciando tiempo de vida útil en los equipos del centro de cómputo.

Costo: los costos de mantener los planos en el mismo espacio físico dentro del chasis aumentan constantemente, ya que el costo del procesamiento de una computadora es menor al costo del procesamiento del plano de control, hablando en costos monetarios.

Innovación: cambios radicales pueden ocurrir en las tecnologías de los planos, pero si se encuentran unidos, es difícil implementar estos cambios sin afectar al otro, por lo que la separación de los planos ha conllevado a la creación de nuevas mejoras independientes para cada plano.

Estabilidad: al manejarse de manera independiente, se puede alcanzar cierto grado de estabilidad a nivel de los planos de datos y control, donde se conocen configuraciones estables de las mismas que se pueden usar como línea base en caso de problemas que se presenten en los dispositivos de red.

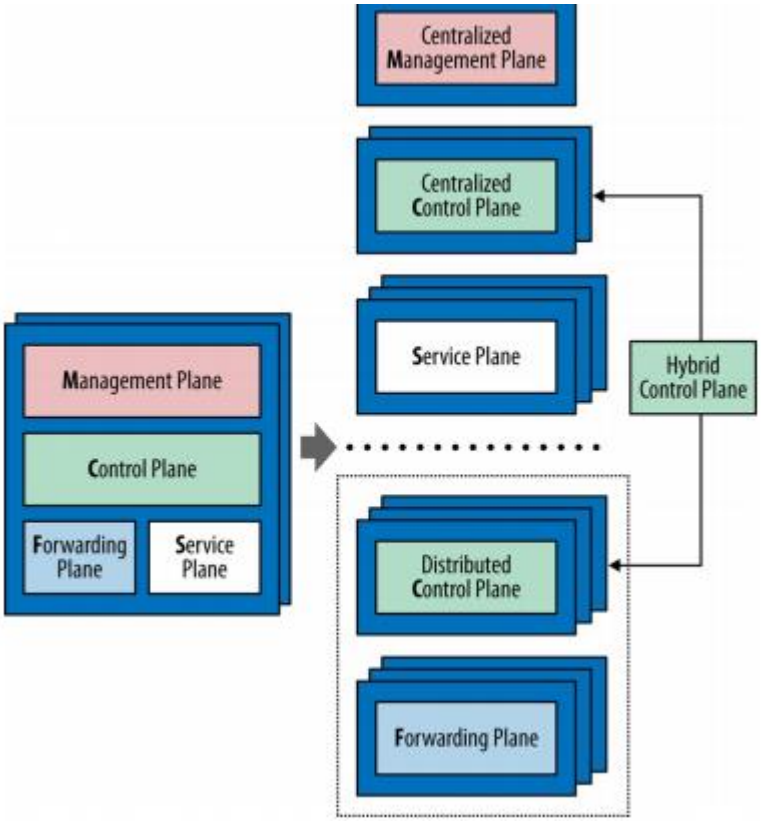


GRÁFICO # 4 Separación de los planos y su flujo
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Limitaciones actuales de las redes de datos

Actualmente, las redes de datos presentan varios desafíos, creados por los avances de la tecnología en los campos de la computación, como el cloud computing, virtualización de servidores, etc.; lo cual evidenció que se necesitaba una nueva manera de clasificar el tráfico de datos, por lo que se identificaron varias limitantes:

Complejidad: las redes actuales han crecido tanto que se han creado innumerables protocolos adaptados para situaciones específicas, lo cual ha causado que se incremente la complejidad al manejar una red de datos: con la inclusión de un nuevo dispositivo, se debe configurar y modificar las configuraciones de la gran parte de los dispositivos de red, creando una excesiva cantidad de sobre trabajo para este tipo de tareas.

Los departamentos de redes de las empresas prefieren mantener las redes de forma estáticas (es decir, no añadir nuevos equipos sólo en caso de ser necesario) para evitar al máximo esta complejidad y por ende, evitar también la interrupción del servicio que viene con estas adiciones.

Este comportamiento de las redes no va acorde al comportamiento en el lado de los servidores, donde las tecnologías como virtualización han proveído de una gran flexibilidad al momento de la creación de nuevos recursos computacionales, así como optimizar al máximo el recurso físico disponible, creando una disparidad entre la flexibilidad de los servidores con la estaticidad de las redes de datos, lo que generalmente causa inconvenientes cuando se intentan integrar ambas en un espacio único como lo es un centro de datos.

Políticas de redes inconsistentes: para configurar una política de red, como una ACL o QoS, en una red estática grande toma mucho esfuerzo humano ir dispositivo en dispositivo introduciendo las configuraciones necesarias, incluyendo que esto también está sujeto al error humano que puede ocurrir al realizar tareas repetitivas tantas veces como sea necesario, esto exacerbado por la complejidad de las redes actuales, vuelve tarea difícil para los administradores de red introducir cambios, por pequeños que sean, a la configuración lógica de su red de datos.

Escalabilidad limitada: en redes pequeñas, la escalabilidad no es bien estimada ya que no surgen las necesidades necesarias para esto, pero en la redes modernas, donde la flexibilidad es algo de todos los días, y con el advenimiento de los usuarios móviles, se crea la necesidad de estar preparados para la inclusión de nuevos usuarios y equipos a la red, adaptarlos a la red actual y asegurarse que tengan accesos a todos los recursos proveídos por ella; las redes actuales no proveen este tipo de escalabilidad, ya que el uso futuro de una red es generalmente impredecible.

Dependencia de proveedores: al momento de planificar una red, se toma en cuenta los proveedores que se espera tener para la misma, en el caso de una red dinámica, los cambios en esta red están sujetos a los ciclos de vida de los productos adquiridos con un proveedor específico, por lo que se

depende del proveedor para implementar nuevas configuraciones o soluciones para problemas diarios de la red, u optimizaciones necesarias para la misma.

Nuevos comportamientos en las redes de datos

Con la aparición de nuevos usos de la tecnología, se añaden nuevas maneras de tratar el tráfico de las redes, adaptándose a las necesidades de las empresas:

Cambio en los patrones de tráfico: tiempo atrás, los datos que viajaban a través de las redes siempre fueron considerados generales, es decir, no existía clasificación en los mismos por servicios, sólo se implementaban seguridades a nivel de VLAN para agrupar las redes por usuarios y no por aplicaciones. En la actualidad, existe una mucha mayor importancia en la clasificación de los datos viajeros por servicios o aplicación, ya que las prioridades en la disponibilidad de los datos se rigen por el tipo de servicio que proveen y su importancia para el negocio.

Cuando las aplicaciones de red eran de tipo cliente-servidor, la información se obtenía de una sola fuente y sólo iba en una vía, donde el cliente consumía lo que el servidor enviaba; actualmente la información es solicitada por muchos usuarios al mismo tiempo, y así mismo, la información es servida por muchos servidores, creando muchas vías de

comunicación entre usuarios finales y servidores, que muy difícilmente pueden ser manejada con las herramientas actuales para la administración de las redes de datos.

Generalización de los servicios de información: con la explosión de uso de teléfonos inteligentes, existen muchos más equipos que se conectan a la red administrada, por lo que es necesario proveer servicios rápidos y efectivos a estos nuevos dispositivos sin afectar a los usuarios actuales de la red.

Servicios en la nube: con la nube, se agregan nuevas variables a los servicios prestados en una red, ya que pueden ser usados en nubes públicas, privadas o híbridas, donde la seguridad de la información es muy importante y la auditoría de la red es una medida rutinaria; este tipo de servicios requieren escalabilidad dinámica para poder crecer dependiendo de las necesidades.

FUNDAMENTACION TEORICA

SDN O REDES DEFINIDAS POR SOFTWARE

Según Marschke, Doyle y Moyer (2015), afirman dos conceptos para SDN: "SDN es una arquitectura de capa 2/capa 3 en la cual un controlador centralizado controla el enrutamiento de un conjunto de conmutadores distribuidos" (pag.13) y "SDN es un área conceptual de trabajo donde las redes son tratadas como abstractas y son controladas por programación, con mínima intervención en los elementos de red físicos"

Estos dos conceptos representan la descripción general, pero como toda tecnología, posee una arquitectura definida general que permite su estudio y aplicación de manera organizada.

Arquitectura de las SDN

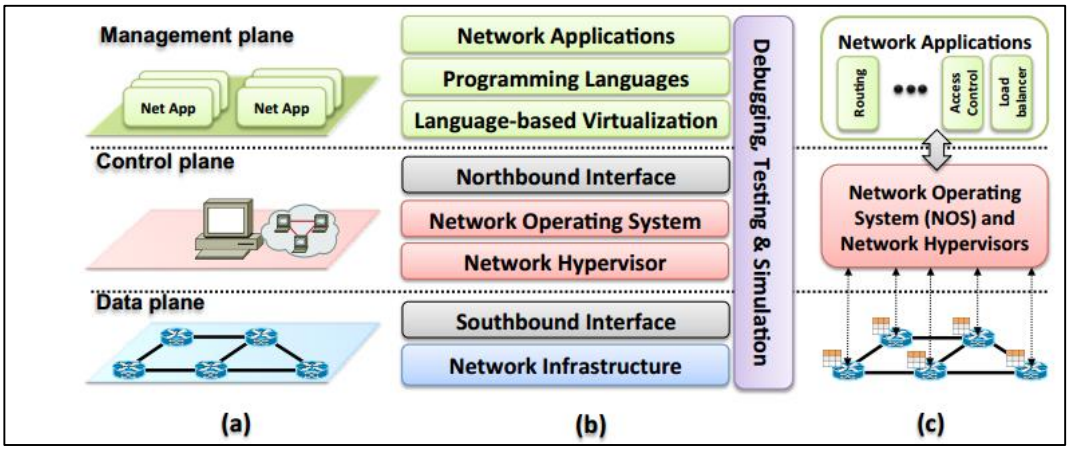


GRÁFICO # 5 Arquitectura de una SDN en planos (a), capas (b) y diseño de red (c).
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Las SDN se pueden definir que poseen varias capas, donde existen los mismos elementos de red que se encuentran en las redes, estas capas son las siguientes:

- Capa de Infraestructura: como las redes de datos básicas, una infraestructura de SDN está compuesta de los mimos elementos, la diferencia está en que la “inteligencia” que poseían estos elementos en las redes básicas, ha sido removida de los mismos para que sean ahora sólo “direccionadores” de las órdenes, es decir, no pueden tomar decisiones por su cuenta.

  Esta llamada “inteligencia” es movida del plano de datos al plano de control, el cual en las redes SDN es centralizado, e incluso, se apertura la red al uso casi exclusivo de estándares de redes abiertos y no propietarios, lo cual es importante para asegurar la coexistencia de equipos de diferentes proveedores y fabricantes a nivel de los planos de control y de datos.

  En la arquitectura SDN existes 2 elementos, los controladores y los elementos de redireccionamiento, los cuales son la representación de los planos de control y de datos, respectivamente.

  Los dispositivos de red se convierten en direccionadores de las órdenes realizadas en los controladores, como Openflow, donde se encargan de cumplir la misma función que realizan los dispositivos de red actuales:

  1. Recibir paquetes de la red en el plano de datos.
  2. Encontrar una regla concordante en el plano de control.

3. Aplicar las acciones a ejecutar, de acuerdo a la regla encontrada.
4. Mantener estadísticas y contadores de los paquetes procesados a través de ambos planos.

En los dispositivos de redes que soportan el estándar básico Openflow, con cada versión del software se fueron agregando el soporte para varios protocolos de red comunes usados en las redes de datos (Ethernet, IPv4/IPv6, TCP, UDP, MPLS, etc.), lo cual permite conectar redes SDN a redes normales por medio de protocolos compatibles, sin afectar mayormente al tráfico actual de las redes.

En tanto a las soluciones virtuales, existen gran variedad de switches virtuales que pueden interactuar con redes SDN y Openflow, entre los que contamos:

- Switch Light switch
- Open vSwitch switch
- Openflow Reference switch
- Pica8 switch

La aparición de gran variedad de switches virtuales, se da por la aparición de un concepto cercano a SDN llamado virtualización de redes, la cual usa varios conceptos generales de SDN para su propia implementación, y sus aportes en el campo tecnológico benefician también al avance de las redes SDN.

Es también posible mencionar que el advenimiento de las redes SDN ha causado que aparezcan proveedores de hardware de red genérico, hardware que no posee mayor ingeniería en el plano de control ya que fueron creados exclusivamente para ser controlados en una red SDN, este tipo de hardware es usualmente creado por empresas pequeñas e innovadoras, por mencionar algunas:

  - Big Switch
  - Pica8
  - Plexxi

- Capa de interfaces sur: SDN se maneja de manera programable, y para esto, se requiere la intervención de las APIs, siendo estas interfaces de programación las responsables de comunicar las diferentes planos a través de las herramientas proveídas por la tecnología.

  En el caso de las comunicaciones entre el plano de control y de datos, se llaman a estas interfaces “interfaces sureñas”, dado que son las que comunican los elementos sureños con los centrales, en el caso de la arquitectura SDN, comunica a la infraestructura de red con el hipervisor de red. Es aquí donde aparece el nombre de Openflow como el máximo referente de las interfaces sureñas en las redes SDN, el cual se ha convertido en un estándar de facto para establecer el uso general del mismo.

  Una interfaz sureña es OVSDB, que se desarrolló para ofrecer mayores capacidades a los switches Open vSwitch, como la

creación de varias instancias virtuales en el switch virtual, estableces políticas de QoS y designarlas a cada interface, manejar colas de paquetes y recolectar datos estadísticos de los paquetes que son procesados, por lo que se considera como un complemento a Open vSwitcth.

Otra interfaz es POF, donde su principal diferencia con Openflow es que maneja un estándar de comunicación con los switches de la red SDN en el cual no se necesita ningún tipo de procesamiento adicional de los paquetes en los dispositivos de la red, es decir, los paquetes llegan a éstos listos para ser reenviados a su siguiente destino, lo que no sucede con Openflow, ya que este añade cabeceras que deben ser interpretados por los dispositivos de red; en este caso, POF libera de todo procesamiento a los dispositivos y gobierna totalmente el direccionamiento de los datos que se transportan por la red, dándolo un aspecto agnóstico al ambiente donde se implementa, ya que se puede adaptar a cualquier dispositivo de red que deba controlar.

Otras interfaces sureñas creadas para las redes SDN son:

  - Opflex
  - OpenState
  - ROFL
  - HAL

- Capa de hipervisores de red: en las redes de datos también existen hipervisores, donde se intenta aprovechar toda la capacidad del

hardware de red instalado, y además, se obtienen los beneficios que la virtualización provee, como escalabilidad y alta disponibilidad, en este caso, de las redes de datos.

En las redes SDN, esto se conoce como *slicing*, y cuenta a FlowVisor como su principal componente, partiendo de su concepto básico, que es permitir muchas redes lógicas en la misma instancia de Openflow. Esto permite tener redes muy disparejas coexistir en el mismo ambiente de red, sin limitar a las mismas.

Para el slicing, se consideran 5 variables que se pueden usar: ancho de bando, topología, tráfico, CPU y tablas de redireccionameito, e incluso, se puede usar diferentes controladores para cada slice, cada uno con sus ventajas y desventajas pero sin afectar a las redes vecinas en otros *slices*.

OpenVirteX es otro hipervisor de red, que separa las redes SDN por 3 componentes: topología, direccionamiento y funciones de virtualización, donde su principal característica es no depender del direccionamiento de los equipos de red por debajo, así que permite aislar la topología de red virtual totalmente y así tener total independencia y control de la red.

Autoslice y AutoFlow son hipervisores que separan las redes igualmente, pero su objetivo principal es reducir las acciones operativas de los administradores de la red SDN, automatizando la gran parte de comandos a realizar y monitoreando constantemente los flujos de la red SDN.

En el caso de los hipervisores de red, los grandes fabricantes de software de redes se han inmiscuido con mayor capacidad en este ámbito, por lo que podemos ver varias opciones:

  - NVP de VMware: tecnología adquirida por la compra de Nicira, pionera en la virtualización de redes, provee una solución completa para grandes redes de datos, integrable totalmente con el centro de datos actual, cubriendo todos los aspectos de una red física normal con los beneficios de una red virtualizada.
  - SDN VE de IBM: similar a NVP, posee una capacidad inicial de soportar 16000 redes virtuales y 128000 máquinas virtuales en una instalación simple.

- Capa de controladores: en esta capa, se encuentran los sistemas operativos de red que conocemos como Cisco IOS y JuniperOS, estos sistemas operativos son la base del funcionamiento de las redes actuales, donde conviven cercanamente con el hardware y por ende el plano de datos para el procesamiento de los paquetes. En el caso de las redes SDN, este funcionamiento es proporcionados por NOS, que es quien controla la totalidad de las ordenes a dar a los equipos de la red, el cual puede ser centralizado o distribuido a lo largo de la infraestructura de red, teniendo ambos su características propias:

- Centralizado: un controlador centralizado es una sola entidad que maneja todo el direccionamiento de una red SDN, al ser centralizado, tiene los mismos defectos de un sistema central, como ser no escalable a las necesidades de la red. Estas limitantes son solucionadas a nivel de software con procesos multihilos que permiten realizar múltiples tareas de control de flujo a la vez, y creando aislamiento para las redes dentro del controlador, para en caso de una falla en una red, no se propague a otras; se pueden encontrar varios controladores centralizados en el mercado, como:
    - NOX
    - Beacon
    - Maestro
    - Floodlight
- Distribuido: en este caso, el controlador puede existir en dos formas, como un clúster de aplicaciones distribuidas o como elementos físicos distribuidos; ambas formas requieren conectividad rápida para la convergencia de los datos entre todos los elementos del controlador y también sistemas de protección ante fallas en uno de los elementos que componen el controlador distribuido; ejemplos de controladores distribuidos son:
    - Onix
    - Hyperflow

- HP VAN SDN

Para los controladores distribuidos, hay ciertos parámetros que deben cumplirse para mantener la integridad de la información que pasa por los mismos, ya que al existir varios nodos con la misma información, se debe llevar un registro centralizado de cada operación de escritura que hace cada nodo, para no repetir los mismos cambios en otro nodo.

Así mismo, también deben poseen características de ser resistentes a fallos, ya que si un nodo del controlador falla, otro debe reemplazarlo para mantener los servicios activos, esto no previene que en caso de una mala configuración en un nodo, se auto aísle el mismo, ya que los controladores distribuidos sólo perciben errores de disponibilidad, mas no de mal funcionamiento.

CUADRO # 2 Tabla de comparación entre controladores y sus diseños.

Component	OpenDaylight	OpenContrail	HP VAN SDN	Onix	Beacon
Base network services	Topology/Stats/Switch Manager, Host Tracker, Shortest Path Forwarding	Routing, Tenant Isolation	Audit Log, Alerts, Topology, Discovery	Discovery, Multi-consistency Storage, Read State, Register for updates	Topology, device manager, and routing
East/Westbound APIs	—	Control Node (XMPP-like control channel)	Sync API	Distribution I/O module	*Not present*
Integration Plug-ins	OpenStack Neutron	CloudStack, OpenStack	OpenStack	—	—
Management Interfaces	GUI/CLI, REST API	GUI/CLI	REST API Shell / GUI Shell	—	Web
Northbound APIs	REST, REST-CONF [200], Java APIs	REST APIs (configu-ration, operational, and analytic)	REST API, GUI Shell	Onix API (general purpose)	API (based on OpenFlow events)
Service abstraction layers	Service Abstraction Layer (SAL)	—	Device Abstraction API	Network Information Base (NIB) Graph with Import/Export Functions	—
Southbound APIs or connectors	OpenFlow, OVSDB, SNMP, PCEP, BGP, NETCONF	—	OpenFlow, L3 Agent, L2 Agent	OpenFlow, OVSDB	OpenFlow

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

- Capa de interfaces norteñas: las interfaces norteñas son el punto de comunicación con el plano de gestión, donde el controlador recibe las órdenes de los administradores de red. En estas interfaces, no existe todavía un estándar libre entre los controladores, cada uno tiene una implementación de estas interfaces diferentes para sus funciones, lo que dificulta la portabilidad de las redes que quieran mezclar controladores desde un mismo punto de gestión. Así mismo, es posible que nunca exista un solo estándar para las interfaces norteñas, ya que, como son el punto de contacto con el exterior, las necesidades de seguridad, políticas de uso o acceso varían mucho de red en red, dependiendo de su uso (una red local de un cibercafé necesita mucha menos seguridad que una red empresarial de un banco, por ejemplo).
- Capa de virtualización de lenguaje: esta capa se presenta como la capa donde ocurre la traducción de las ordenes a darse a la red, para controlar su comportamiento; esto se logra nuevamente con el slicing de la red, creando ambientes aislados para cada tipo de red que pueden ser manejados por diferentes lenguajes de programación especialmente diseñados para crear y convivir con los virtualizadores de lenguaje adaptados al mismo, ejemplos de estos son:
    - Pyretic

- Splendid
- libNetVirt

En el caso de Openflow, usa un controlador tipo NOX para manejar e aislar la infraestructura este nivel, siguiendo la arquitectura del modelo creado por Openflow.

- Lenguajes de programación: Openflow es un ejemplo de un lenguaje de programación de red, hecho a bajo nivel (es decir, interactúa directamente con el hardware); la tendencia es que , para las redes SDN, se mantenga el uso de este tipo de lenguajes de bajo nivel en las comunicaciones con el plano de datos, mientras que en las comunicaciones con el plano de gestión, se pueda mover hacia lenguajes de alto nivel, que sean mucho más entendibles para el ser humano y más fáciles de aprender y aplicar. Las razones por las cueles usar lenguajes de alto nivel son usados en las redes SDN pueden ser:
  - Simplificar tareas al usar lenguajes de alto nivel.
  - Enfocarse en los problemas a resolver y no en los adentros de la tecnología.
  - Usar beneficios conocidos de la programación en alto nivel (modularizacion, reuso, etc.).
  - Expandir el desarrollo de las herramientas de virtualización de red.

Es también un beneficio de usar lenguajes de este tipo, cuando se puede instruir a los desarrolladores a usar sus técnicas de desarrollo previas, como la programación en objetos, para programar redes, algo impensable en una red normal, y distribuirlas en muchos nodos del controlador sin tener experiencia con los dispositivos físicos de red; es decir, se puede programar una red de datos sin conocer el hardware donde será implementado, todo esto siguiendo las directrices de una red SDN.

Ejemplos claros de lenguajes de alto nivel para redes SDN son:

- FatTire
- Flog
- FlowLog
- FML
- HFT
- Procera
- Nettle
- NetCore

Cada uno de estos lenguajes posee características propias y compartidas entre ellos, las cuales detallamos en el siguiente cuadro:

CUADRO # 3 Detalle de los lenguajes de programación usados en SDN

Name	Programming paradigm	Short description/purpose
FatTire [261]	declarative (functional)	Uses regular expressions to allow programmers to describe network paths and respective fault-tolerance requirements.
Flog [257]	declarative (logic), event-driven	Combines ideas of FML and Frenetic, providing an event-driven and forward-chaining logic programming language.
FlowLog [256]	declarative (functional)	Provides a finite-state language to allow different analysis, such as model-checking.
FML [203]	declarative (dataflow, reactive)	High level policy description language (e.g., access control).
Frenetic [204]	declarative (functional)	Language designed to avoid race conditions through well defined high level programming abstractions.
HFT [255]	declarative (logic, functional)	Enables hierarchical policies description with conflict-resolution operators, well suited for decentralized decision makers.
Maple [262]	declarative (functional)	Provides a highly-efficient multi-core scheduler that can scale efficiently to controllers with 40+ cores.
Merlin [263]	declarative (logic)	Provides mechanisms for delegating management of sub-policies to tenants without violating global constraints.
nlog [112]	declarative (functional)	Provides mechanisms for data log queries over a number of tables. Produces immutable tuples for reliable detection and propagation of updates.
NetCore [205]	declarative (functional)	High level programming language that provides means for expressing packet-forwarding policies in a high level.
NetKAT [226]	declarative (functional)	It is based on Kleene algebra for reasoning about network structure and supported by solid foundation on equational theory.
Nettle [223]	declarative (functional, reactive)	Based on functional reactive programming principles in order to allow programmers to deal with streams instead of events.
Procera [224]	declarative (functional, reactive)	Incorporates a set of high level abstractions to make it easier to describe reactive and temporal behaviors.
Pyretic [225]	imperative	Specifies network policies at a high level of abstraction, offering transparent composition and topology mapping.

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

- Capa de aplicaciones de red: estas aplicaciones son el cerebro de la red, ya que son las que realmente crean las órdenes lógicas a enviarse al plano de control para su aplicación en el plano de datos. Estas aplicaciones de red cumplen las mismas tareas que las redes normales, pero además pueden incluir características propias de las redes SDN, como controlar el uso de energía y proporcionar tolerancia a los fallos de algún nodo controlador de la red SDN, etc. Las funcionalidades más importantes que son cumplidas por estas aplicaciones son:
    - Ingeniería de tráfico como balanceo de carga, monitoreo en del tráfico, prevención de cuellos de botella en el transporte de datos a través de la red y optimización de tráfico.

- Movilidad y servicios inalámbricos de servicios de red, los cuales también pueden ser adaptados a las redes SDN, siendo de los ejemplos más conocidos las implementaciones de OpenRadio y SoftRAN, que son la contraparte de Openflow en el campo del espectro radioeléctrico y su manejo.
- Mediciones y monitoreo de la red, donde se analiza a profundidad el comportamiento de la red en tiempo real desde el punto de vista de las redes SDN.
- Seguridad de la red, ya que es posible implementar en estas aplicaciones las seguridades necesarias para una red SDN e incluso mejorar las seguridades de una red actual.

Openflow

Openflow es una tecnología derivada de una investigación sobre redes virtuales realizada en la universidad de Standford, basado en Ethane, un producto temprano que ya contenía algunas de las funciones actuales de Openflow. En el 2011, se creó la ONF (Open Networking Foundation) para regular, promocionar y expandir el uso de Openflow en redes empresariales. Actualmente existen varios tipos de controladores como Openflow, los cuales la ONF se encarga de promocionar todas las tecnologías relacionadas con SDN.

Openflow se encarga de acercar conceptos ya conocidos en el campo, como separación de planos y agrega otros tantos, como el

uso de estándares de comunicación entre los controladores y los elementos de red y proporcionar API a los administradores de red para “programar” la red SDN.

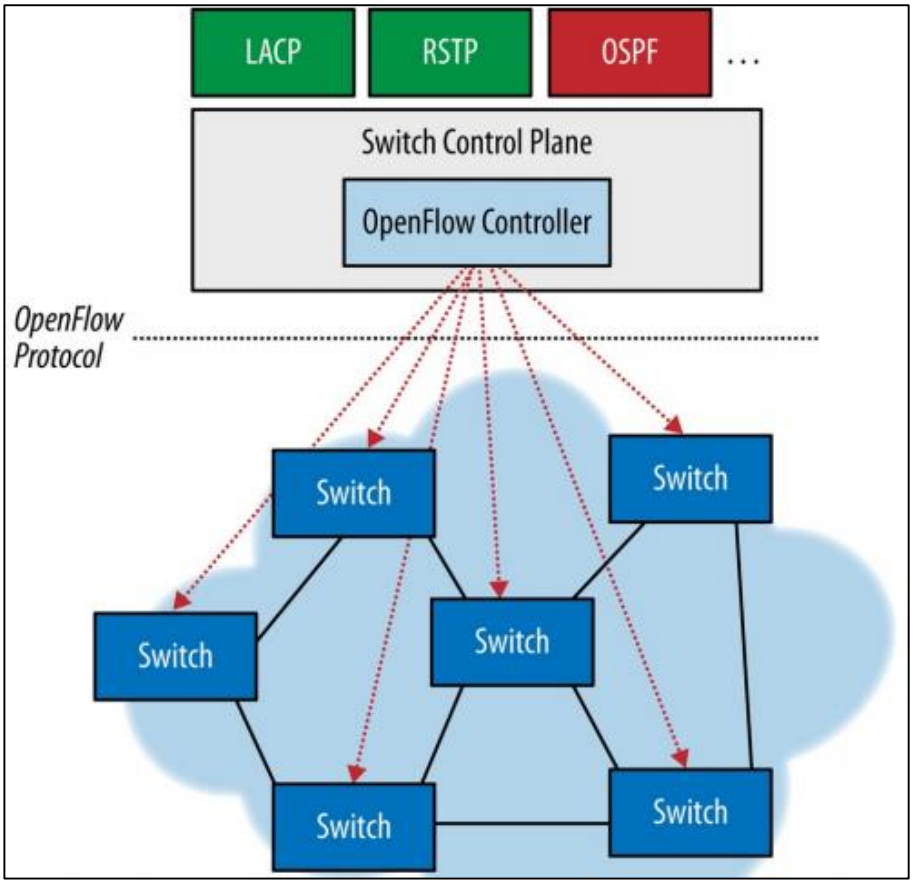


GRAFICO # 6 Estructura simple de Openflow
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Las características de Openflow han mejorado sustancialmente desde su primera iteración, la versión 1.0; actualmente está en la versión 1.4 que añade corrección de bugs y nuevas funcionalidades, que detallamos en la siguiente tabla:

CUADRO # 4 Versiones de Openflow y sus características

Version	Wire Protocol Number	Release Date MM/DD/YYYY	Feature
1.0.0	0x01	12/31/2009	Slicing
			Flow cookies
			User-specifiable datapath description
			Match on IP fields in ARP packets
			Match on IP ToS/DSCP bits
			Query port stats for individual ports
			Improved flow duration resolution in stats/expiry messages
1.1.0	0x02	02/28/2011	Multiple tables (pipelines)
			Groups
			MPLS and VLAN tags
			Virtual ports
			Controller connection failure
1.2	0x03	02/05/2011	Extensible match support
			Extensible "set_field" packet rewriting support
			Extensible context expression in "packet-in"
			Extensible Error messages via experimenter error type
			IPv6 support
			Simplified behavior of flow-mod request
			Removed packet parsing specification
			Controller role change mechanism
1.3.0	0x04	04/13/2012	Refactor capabilities negotiation
			More flexible table miss support
			IPv6 extension header handling
			Per flow meters
			Per connection event filtering
			Auxiliary connections
			MPLS Bottom of Stack (BoS) matching
			Provider Backbone Bridging (PBB) tagging
			Less restrictive tag order
			Tunnel-ID metadata
			Cookies in packet-in
			Duration for stats
			On-demand flow counters
1.3.1	0x04	09/06/2012	Improved version negotiation
1.3.2	0x04	04/25/2013	Assorted feature fixes and rule clarifications
1.3.3	0x04	09/27/2013	Assorted feature fixes and rule clarifications
1.3.4	0x04	03/27/2014	Assorted feature fixes and rule clarifications
1.4.0	0x05	10/14/2013	More extensible wire protocol
			More descriptive reasons for packet-in
			Optical port properties
			Flow-removed reason for meter delete
			Flow monitoring
			Role status events
			Eviction
			Vacancy events
			Bundles
			Synchronized tables
			Group and Meter change notifications
			Error code for bad priority
			Error code for Set-async-config
			PBB Use Customer Address (UCC) field
			Error code for duplicate instruction
			Error code for multipart timeout
			Change default TCP port to 6653 (from 6633 and 976, used by previous versions)

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Componentes: Los protocolos Openflow están distribuidos en:

- Protocolo de cable: este protocolo se encarga de controlar el "medio", es decir, posee las características necesarias para controlar las sesiones, los flujos de red, intercambio de información entre los flujos y la recolección de estadísticas. Estas características no son nuevas, pero Openflow introduce nuevas características como mantener las configuraciones finales de la red en la memoria del controlador y no en los dispositivos finales, poder manejar a placer los paquetes de capa 2 y capa 3 para hacerlos compatibles entre

todos los dispositivos de red SDN y modificarlos según la necesidad actual. Esto permite tener capacidades de emulación y enrutamiento de todos los paquetes en capa 2 y capa 3, todo controlado con un proceso de coincidencia y marca de los paquetes que fluyen a través de Openflow.

- Protocolo de configuración y gestión: el protocolo of-config, el cual se usa para realizar la gestión del controlador a nivel de puertos lógicos, mantener la alta disponibilidad de los nodos del controlador y crear las rutas necesarias para la recuperación y operación correcta del controlador en caso de una falla de conexión entre los nodos.

También es posible separar Openflow de una manera más cercana a la red normal, estableciendo elementos claros dentro del controlador, como:

- El controlador Openflow
- El switch Openflow , que a su vez posee:
    - Puertos
    - Seguridades de Openflow
    - La tabla de flujos de red, que son las reglas que aplican a los paquetes.

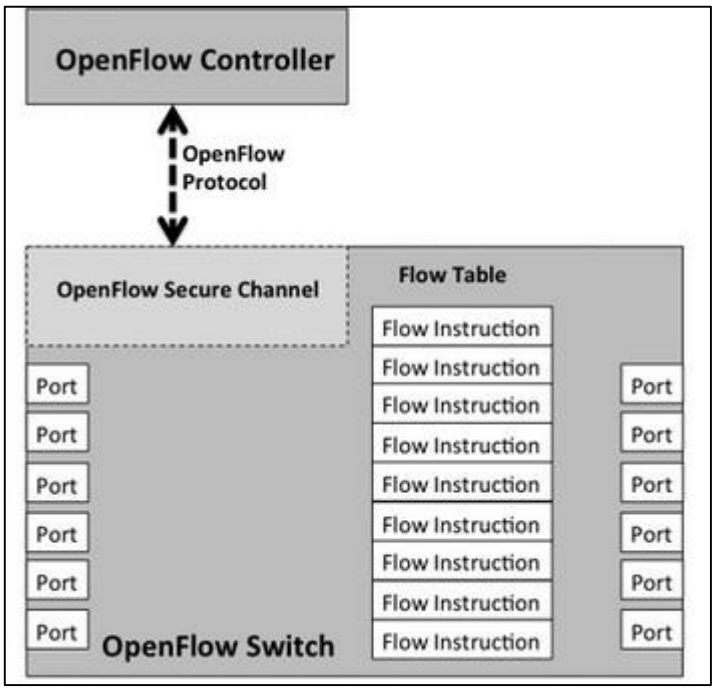


GRAFICO # 7 Estructura básica de Openflow
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Controlador Openflow: El controlador es el elemento que se encarga de transmitir las ordenes al switch Openflow para su aplicación en los elementos de red, dado que el controlador no es el que genera la orden, sino es un traductor de las ordenes enviadas por los administradores de red, o pueden ser también las reconfiguraciones de red realizadas por protocolo de enrutamiento superiores como BGP o IS-IS.

Switch Openflow: es el elemento que recibe las órdenes del controlador y se encarga de aceptarlas, verificarlas y aplicarlas en el plano de control, sus componentes son:

- Puertos: representan las mismas estructuras lógicas que un puerto de red físico, donde se recibe y se envía paquetes de datos a destinos variados; la diferencia es que al ser puertos Openflow, se pueden crear, modificar y eliminar a necesidad de la red, y así mismo pueden usar los mismos puertos físicos de un switch, crear puertos lógicos o reservar puertos para tareas específicas.
    - Puertos físicos: son los puertos de una switch físico, que no son manejados por el plano de control del mismo, sino por el controlador Openflow, se puede considerar su uso en una red SDN como una subred lógica dentro del puerto físico, aislada totalmente de las demás redes en esos puertos.
    - Puertos lógicos: no tienen relación con los puertos físicos, se parecen más a las interfaces lógicas que existen en los switches actuales como túneles e interfaces locales. Estos puertos tienen la misma validez que un puerto físico usado en Openflow.
    - Puertos reservados: son los puertos que Openflow reserva para funciones internas, estos pueden ser del tipo ALL, CONTROLLER, TABLE, IN_PORT, ANY, LOCAL, NORMAL y FLOOD.

Seguridad de Openflow: Se encarga de manejar la comunicación segura de los elementos de Openflow, desde el controlador hasta los switches Openflow, es decir, todo tipo de eventos, notificaciones, configuraciones que son intercambiadas entre los planos, pasa por el canal seguro de

Openflow. El puerto usado para esto es el TCP 6653 y se encarga de mantener la cadena de seguridad a través de toda la red SDN controlada por Openflow.

Tabla de Flujos: esta tabla contiene toda la lógica necesaria para clasificar los paquetes recibidos a través de la red, posee los siguientes campos:

- Match: son las condiciones que debe cumplir un paquete para ser marcado como reconocido por el switch Openflow.
- Priority: define la prioridad de la regla en la tabla.
- Counter: Guarda estadística de la cantidad de paquetes que han sido reconocidos por esta regla.
- Instruction: es la orden que se debe aplicar al paquete reconocido.
- Timeouts: es el tiempo máximo que puede estar activa la regla en el switch.
- Cookie: usado internamente para filtrar las reglas en la tabla.
- Flags: usada para modificar el uso de los flujos en casos específicos.

Estas reglas en la tabla de flujos son seguidas en base a la primicia “regla más parecida es aplicada” y puede la orden a enviar puede usar tanto los puertos físico y lógicos reconocidos por el switch Openflow.

Esta forma de controlar la red por parte de Openflow hace que pueda integrarse totalmente en una red física normal, ya que la tabla de reglas es un conjunto de condiciones que se pueden aplicar a los paquetes antes de salir por las salidas de red indicadas por protocolo es de enrutamiento en

la red física, es decir, puede ordenar los paquetes dentro de la red local y aun así usar la topología de red normal para llegar a otro destino.

Cadena de las reglas en la tabla de flujo:

Cuando Openflow recién fue desarrollado, sólo existía una sola tabla de flujo, lo cual limitaba la escalabilidad del switch Openflow, por lo que en las subsiguientes iteraciones se decidió crear múltiples tablas de flujo con múltiples reglas en las mismas, permitiendo escalar el control de las reglas hasta la capacidad total del controlador.

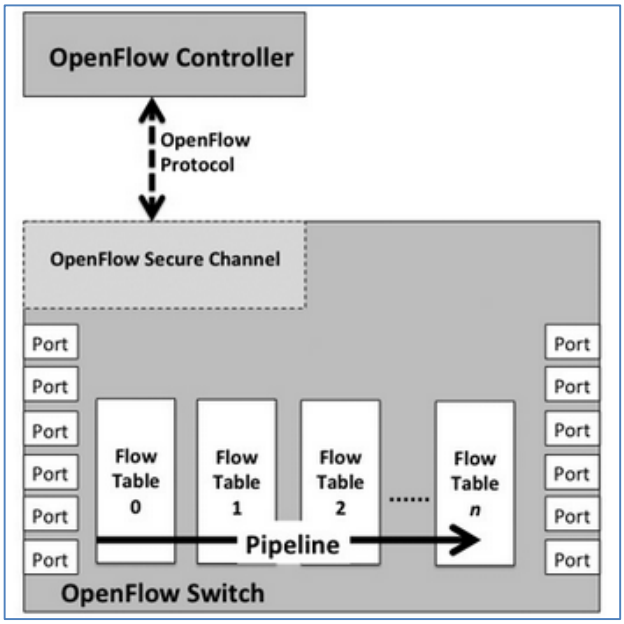


GRAFICO # 8 Cadena de las reglas en Openflow
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

El flujo de la aplicación de una regla en este escenario es el siguiente:

- El paquete busca coincidencia en la primera tabla de flujo.

- Si no encuentra coincidencia, se actualiza el número de tablas actuales y se compara con todas las encontradas hasta encontrar una tabla que posea una regla aplicable.
- Ejecuta las acciones contenidas en el campo Instruction para la regla concordante.
- Si no encuentra ninguna regla, dependiendo de la configuración del switch Openflow, redireccionará el paquete a otro elemento de la red SDN, descartará el paquete completamente o lo reenviará de vuelta al controlador para su revisión.
- En el caso de que una regla sea encontrada, se revisa el paquete y se estandariza el mismo y se actualiza la información extra, conocida como “metadata” para estableces que el paquete ha sido procesado por el switch Openflow.

Switches Openflow: variedades

CUADRO # 5 Mostrando las variedades de switches Openflow disponibles.

Group	Product	Type	Maker/Developer	Version	Short description
Hardware	8200zl and 5400zl [125]	chassis	Hewlett-Packard	v1.0	Data center class chassis (switch modules).
	Arista 7150 Series [126]	switch	Arista Networks	v1.0	Data centers hybrid Ethernet/OpenFlow switches.
	BlackDiamond X8 [127]	switch	Extreme Networks	v1.0	Cloud-scale hybrid Ethernet/OpenFlow switches.
	CX600 Series [128]	router	Huawei	v1.0	Carrier class MAN routers.
	EX9200 Ethernet [129]	chassis	Juniper	v1.0	Chassis based switches for cloud data centers.
	EZchip NP-4 [130]	chip	EZchip Technologies	v1.1	High performance 100-Gigabit network processors.
	MLX Series [131]	router	Brocade	v1.0	Service providers and enterprise class routers.
	NoviSwitch 1248 [124]	switch	NoviFlow	v1.3	High performance OpenFlow switch.
	NetFPGA [48]	card	NetFPGA	v1.0	1G and 10G OpenFlow implementations.
	RackSwitch G8264 [132]	switch	IBM	v1.0	Data center switches supporting Virtual Fabric and OpenFlow.
	PF5240 and PF5820 [133]	switch	NEC	v1.0	Enterprise class hybrid Ethernet/OpenFlow switches.
	Pica8 3920 [134]	switch	Pica8	v1.0	Hybrid Ethernet/OpenFlow switches.
	Plexxi Switch 1 [135]	switch	Plexxi	v1.0	Optical multiplexing interconnect for data centers.
	V330 Series [136]	switch	Centec Networks	v1.0	Hybrid Ethernet/OpenFlow switches.
	Z-Series [137]	switch	Cyan	v1.0	Family of packet-optical transport platforms.
Software	contrail-vrouter [138]	vrouter	Juniper Networks	v1.0	Data-plane function to interface with a VRF.
	LINC [139], [140]	switch	FlowForwarding	v1.4	Erlang-based soft switch with OF-Config 1.1 support.
	ofsoftswitch13 [141]	switch	Ericsson, CPqD	v1.3	OF 1.3 compatible user-space software switch implementation.
	Open vSwitch [142], [109]	switch	Open Community	v1.0-1.3	Switch platform designed for virtualized server environments.
	OpenFlow Reference [143]	switch	Stanford	v1.0	OF Switching capability to a Linux PC with multiple NICs.
	OpenFlowClick [144]	vrouter	Yogesh Mundada	v1.0	OpenFlow switching element for Click software routers.
	Switch Light [145]	switch	Big Switch	v1.0	Thin switching software platform for physical/virtual switches.
	Pantou/OpenWRT [146]	switch	Stanford	v1.0	Turns a wireless router into an OF-enabled switch.
	XorPlus [46]	switch	Pica8	v1.0	Switching software for high performance ASICs.

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

FUNDAMENTACIÓN LEGAL

DECRETO PRESIDENCIAL 1014

Artículo 1.- Establecer como política para las entidades de la administración pública central la utilización de software libres en sus sistemas y equipamientos informáticos.

Artículo 2.- Se entiende por software libre, a los programas de computación que se pueden utilizar y distribuir sin restricción alguna, que permitan su acceso a los códigos fuentes y que sus aplicaciones puedan ser mejoradas. Estos programas de computación tienen las siguientes libertades:

• Utilización del programa con cualquier propósito de uso común

• Distribución de copias sin restricción alguna

• Estudio y modificación del programa (Requisito: Código Fuente disponible)

• Publicación del programa mejorado (Requisitos: Código Fuente disponible).

Artículo 3: las entidades de la administración pública central previa a la instalación del software libre para sus equipos, deberá verificar la existencia de capacidad técnica que brinde soporte necesario para el uso del software.

Artículo 4: Se faculta la utilización de software propietario (no libre) únicamente cuando no exista una solución de software libre supla las necesidades requeridas, o cuando esté en riesgo la seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retorno.

Para efecto de este decreto se comprende cómo seguridad nacional. Las garantías para la supervisión de la colectiva y defensa del patrimonio nacional.

Para efecto de este decreto se entiende por el punto de no retorno, cuando el sistema o proyecto informático se encuentre en cualquier de estas condiciones:

a) Sistema en producción funcionando satisfactoriamente y que el análisis de costo beneficio muestra que no es razonable ni conveniente una migración a software libre.

b) Proyecto en estado de desarrollo y que un análisis de costo – beneficio maestre que no se conveniente modificar el proyecto de usar software libre.

Periódicamente se evaluaran los sistemas informáticos propietario con la finalidad de migrarlos a software libres.

Articulo 5.- Tanto para software libres como software propietario, siempre y cuando se satisfagan los requerimientos, se debe referir las soluciones en este orden:

a) Nacionales que permitan autonomía y soberanía tecnológica.

b) Regionales con componente nacional

c) Regionales con proveedores nacionales

d) Internacionales con componentes nacionales

e) Internacionales con proveedores nacionales

f) Internacionales

Artículo 6.- La Subsecretaria de informática como órgano regulador y ejecutor de las políticas y proyectos informáticos en las entidades del Gobierno Central deberá realizar control de seguimiento de este decreto.

Para todas las evaluaciones constantes en este decreto la Subsecretaria de Informática establecerá los parámetros y metodologías obligatorias.

Artículo 7.- Encárguese de la ejecución de este decreto los señores Ministros Coordinadores y el señor Secretario General de la Administración Pública y comunicación.

Dado en el Palacio Nacional de la ciudad de San Francisco de Quito, distrito Metropolitano, el día 10 de abril de 2008.

PREGUNTAS A CONTESTARSE

H1: ¿Que funcionalidades beneficiosas para una red se obtendrá al aplicar una red SDN?

H2: ¿Es Openflow una alternativa de software libre viable para la implementación de una red SDN?

VARIABLES DE LA INVESTIGACIÓN

Variable independiente: definidas por software

Variable dependiente 1: redes

Variable dependiente 2: Diseño y simulación de un prototipo

CAPÍTULO III

METODOLOGÍA PPDIOO

En este capítulo analizaremos la metodología escogida para realizar este proyecto y su aplicación sobre la implementación del mismo.

PPDIOO es una metodología para definir ciclos de vida de un diseño de red creada por Cisco, la cual provee los siguientes beneficios:

- Disminuye el TCO de un proyecto de red validando los requisitos previos para el diseño e implementación del mismo y planeando el requerimiento de recursos, cambios de infraestructura a lo largo del proyecto.
- Incrementa la disponibilidad de la red creando diseños sólidos de red y validando la operación de la red final.
- Mejora la agilidad del negocio uniendo las necesidades con la estrategia tecnológica del mismo.
- Incrementa los tiempos de acceso a las aplicaciones y servicios mejorando la estabilidad, escalabilidad, seguridad y rendimiento de la red.

Análisis de factibilidad

- **Factibilidad Operacional**

  El diseño de una red SDN prototipo es una propuesta que no ha sido explorada en su extensión dentro del ámbito académico, como se puede mostrar en los siguientes gráficos, donde se realizó una

encuesta básica a una población de 100 estudiantes de la Universidad de Guayaquil, donde se encontró que la gran parte de estudiantes conocen sobre virtualización pero no sobre la virtualización de redes o redes SDN, y que además, están abiertos a conocer más de esta tecnología:

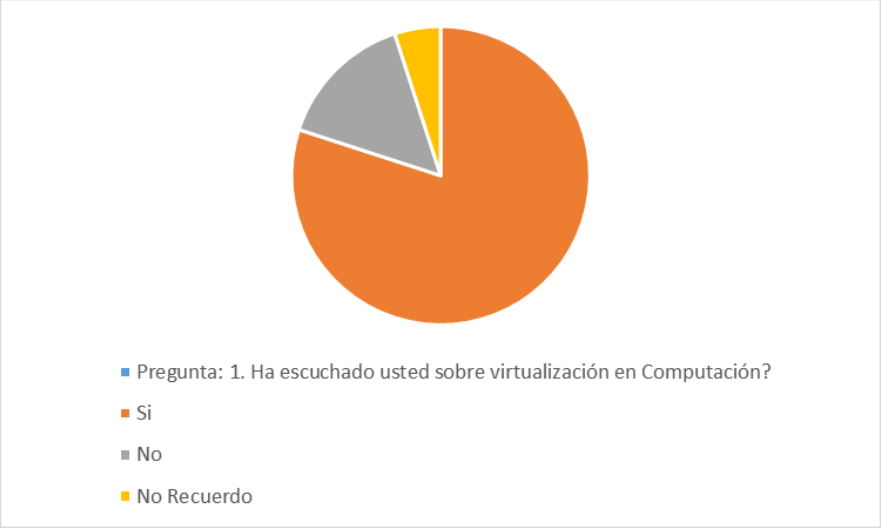


GRAFICO # 9 PREGUNTA DE ENCUESTA # 1
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

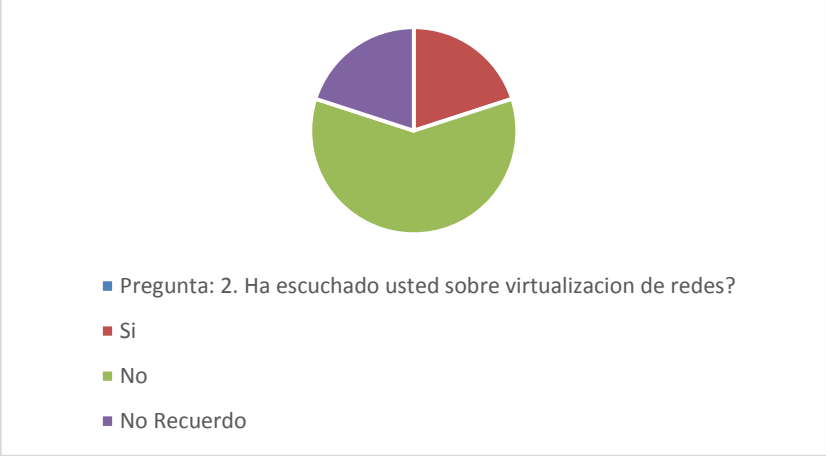


GRAFICO # 10 PREGUNTA DE ENCUESTA # 2
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

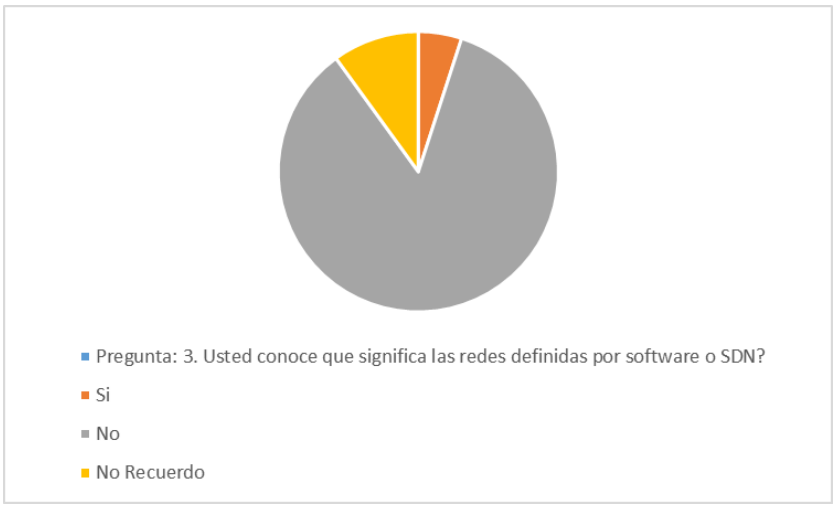


GRAFICO # 11 PREGUNTA DE ENCUESTA # 3
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

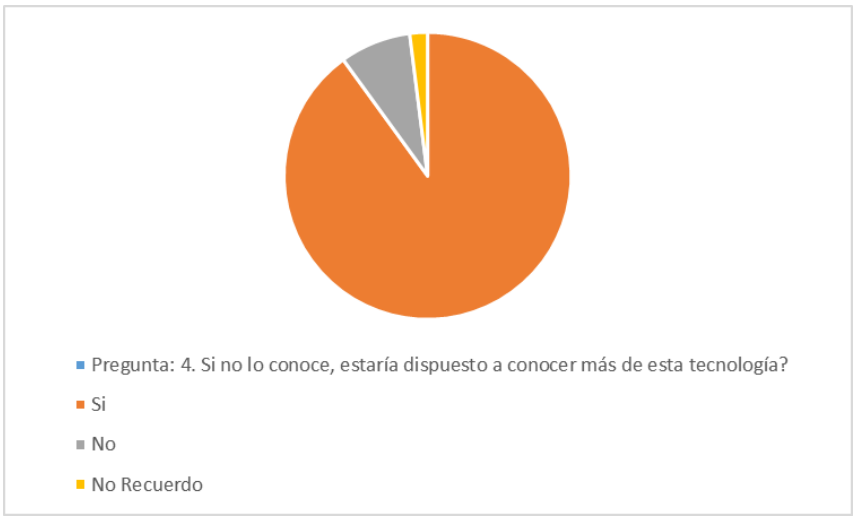


GRAFICO # 12 PREGUNTA DE ENCUESTA # 4
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

- **Factibilidad Técnica**

  En este proyecto se encuentran disponibles todas las herramientas a usarse para la implementación del prototipo:

  - Una computadora personal de las siguientes características:
    - Procesador Intel Core i5
    - Memoria de 8 GB
    - Disco duro sólido de 256 GB
    - Tarjeta de red de 10/100/1000 MB
  - Un hipervisor de sistemas operativos como VirtualBox 5.0.8.
  - Software libre:
    - Mininet 2.2 para la creación de la topología de red.
    - Opendaylight “Helium” para la configuración de la red SDN.
    - Xming 6.9.0.31 para visualizar de forma local la pantalla de las máquinas virtuales.
    - Ubuntu 14.04
    - Putty 0.65
  - Software propietario:
    - Microsoft Office 2013
    - Microsoft Windows 7
    - Mozilla Firefox 42.0
    - Adobe Acrobat Reader 11

  Dado que el proyecto está enfocado en la implementación de un prototipo no será integrado con redes comerciales, el objeto de este

diseño e implementación es proporcionar un laboratorio de pruebas listo para usarse a partir de los entregables del proyecto, para su posterior desarrollo con propósitos académicos.

- **Factibilidad Legal**

  Este proyecto no registra ningún impedimento de tipo legal, de acuerdo a las leyes de Ecuador, ya que se está usando software propietario correctamente licenciado y todas las ideas y aplicaciones que se realizarán en el proyecto son basadas en software libre, que nos permite realizar modificaciones a las herramientas utilizadas con la licencia GPL.

  Además, dado que el proyecto es un prototipo práctico de una tecnología, no se encuentra restricciones a nivel empresarial o personal para la realización del mismo.

- **Factibilidad Económica**

  En el aspecto económico, el proyecto no reviste gastos en el mismo durante el curso del mismo ya que se usaran herramientas libres para el diseño e implementación. En caso de implementar el prototipo para el acceso general en una institución académica, se usará el siguiente presupuesto referencial para la implementación de los equipos necesarios para soportar el prototipo

CUADRO # 6 Costo de implementación de prototipo

Cantidad	Descripción	Costo/Unitaria	Costo
1	PC de escritorio	450	$ 450,00
15	1metro Cable UTP	0,4	$ 6,00
8	Conector RJ45	0,25	$ 2,00
Costo de Hardware			**$ 458,00**

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Etapas de la metodología del proyecto

Las etapas de la metodología PPDIOO para este proyecto serán las siguientes:

PREPARE: esta etapa se recolecta los requerimientos de la organización y de negocio, se desarrolla una estrategia para el proyecto de red y se explican las tecnologías a usarse en el mismo.

Para este proyecto, se analizó el estado actual del conocimiento de las redes SDN entre los estudiantes de la Universidad de Guayaquil, lo que nos proporcionó los requerimientos básicos que debe cumplir el prototipo a implementarse:

- Debe ser de fácil implementación.
- Debe ser asequible para el público al que está dirigido.
- Debe tomar un tiempo corto su implementación.
- Debe cubrir los aspectos básicos de la tecnología a evaluar.
- Debe ser didáctico.

En este proyecto, las tecnologías a usarse se explican en las siguientes líneas:

VirtualBox

Es una aplicación multiplataforma de virtualización que fue desarrollada por Sun Microsystems y luego adquirida por Oracle Corporation, que permite extender la funcionalidad de equipos de computación personales mediante la ejecución de múltiples sistemas operativos al mismo tiempo, en forma de máquinas virtuales; esto trae algunos beneficios como facilitar la instalación de software por parte de proveedores que usar virtualización para proveer productos listos para el uso, usar eficientemente los recursos libres de las computadores personales, desarrollar escenarios de pruebas y recuperación de desastre para los sistemas operativos y las aplicaciones que se ejecutan sobre los mismos.

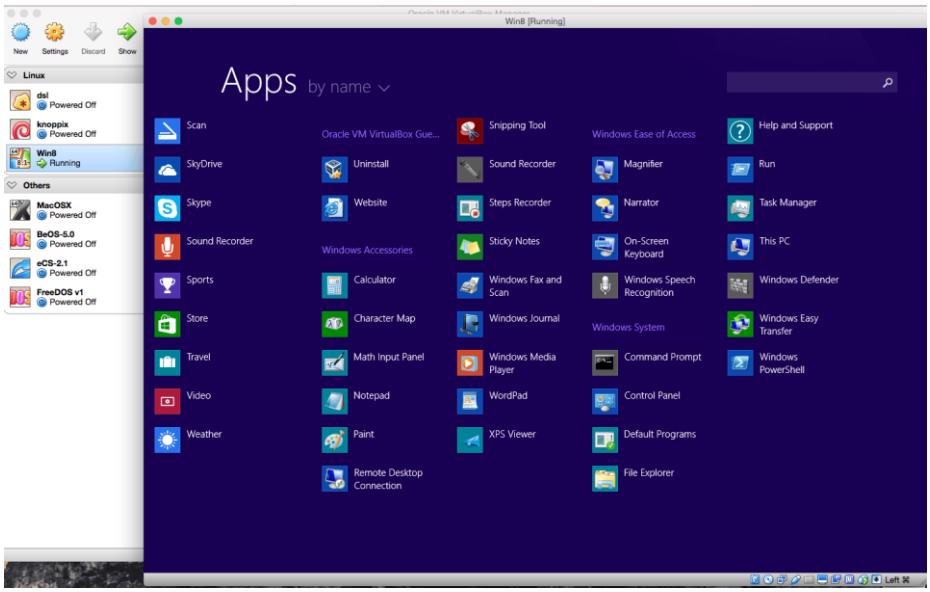


GRAFICO # 12 Ejecución de VirtualBox en un host con MacOS y una máquina virtual Windows 8
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Ubuntu

Es una distribución del sistema operativo Linux, basada en Debian, que se diseñó como una alternativa amigable y segura a los sistemas operativos propietarios como Microsoft Windows y Apple MacOS. Es desarrollada por Canonical Ltda., la cual se apega a los principios del software libre y su promoción a lo largo de la sociedad.

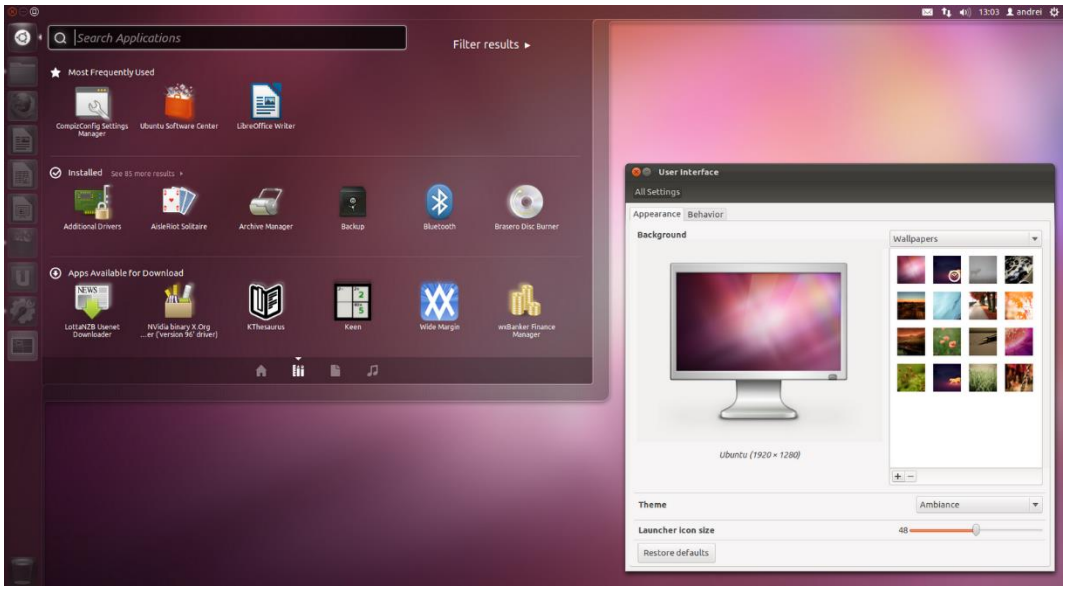


GRAFICO # 13 Escritorio de Ubuntu
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Mininet

Es una aplicación basada en Openflow, que permite la creación de redes virtuales corriendo kernels, switches y aplicaciones reales en una sola máquina virtual. Mininet simplifica en gran medida el proceso de creación de redes virtuales y es muy importante su uso como puerta de entrada para Openflow y tecnologías relacionadas con SDN. Posee 2 formas de interactual con el usuario: por medio de la CLI y por la interfaz gráfica:

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

GRAFICO # 14 Línea de comando de Mininet
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

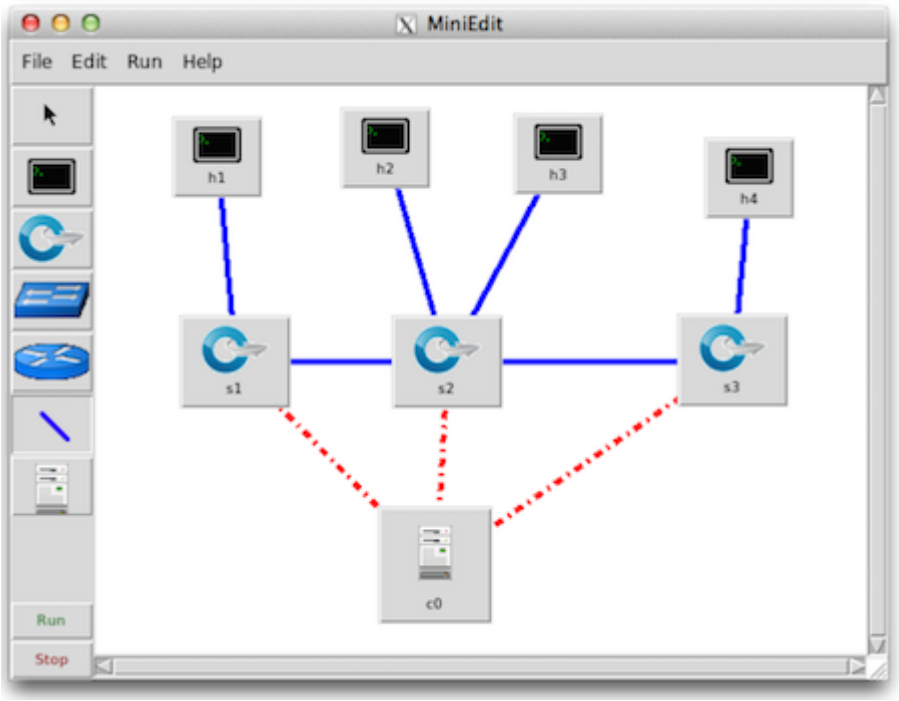


GRAFICO # 15 Interfaz gráfica de Mininet
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Opendaylight

Es una plataforma SDN completa, basada en Openflow, que provee todas las funciones de una plataforma SDN, desde la conexión con los dispositivos de red hasta las API necesarias para la instrucción de órdenes y reglas a la red. Opendaylight fue creado en el 2013 con la unión de varios productos relacionados con Openflow como Beacon, Open vSwitch, etc., que proporcionaron una sola solución SDN para los requerimientos de la industria para las redes SDN.

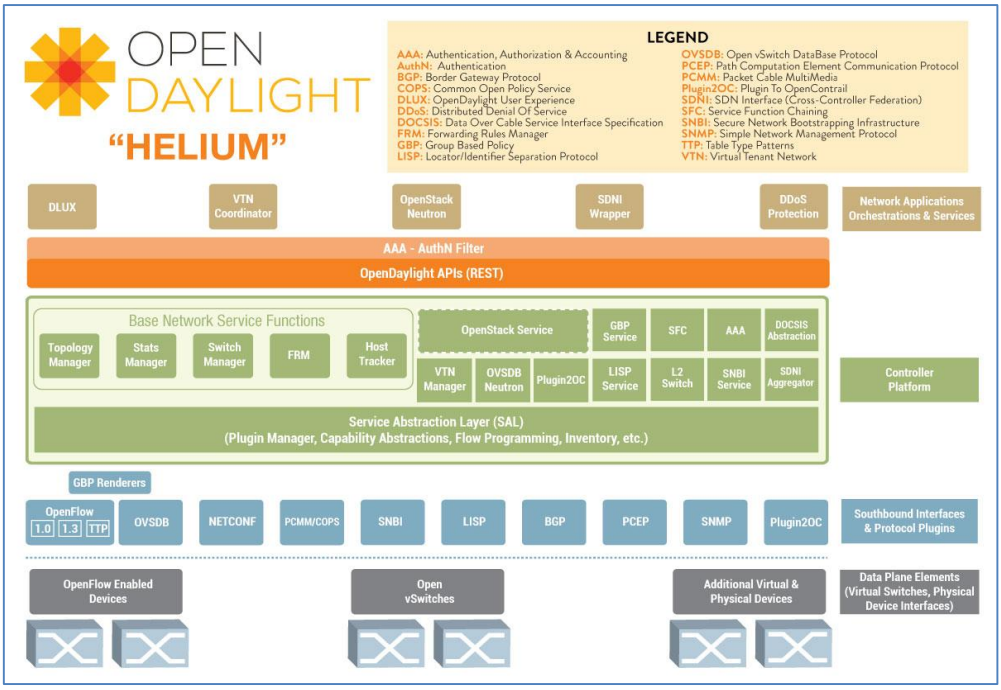


GRAFICO # 16 Estructura de Opendaylight
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

PLAN: esta fase se encarga de identificar los requerimientos de la red en base a los objetivos, los recursos presentes y las necesidades de los

usuarios. Para el proyecto presente, hemos identificado las siguientes características que debe cumplir:

- La plataforma física que debe poseer las siguientes características:
    - Un PC que posea mínimo 8 GB de RAM, un procesados Quad-Core de 2.0 Ghz o mayor, capacidad de disco duro mínima de 256 GB y conectividad de red por medio de cable físico o interfaz WLAN.
- El diseño de red que debe poseer las siguientes características:
    - Un controlador Openflow al menos.
    - Un switch Openflow al menos.
    - 2 host al menos,
- Los recursos humanos para realizar este proyecto:
    - 1 persona con experiencia con sistemas operativos.

El plan a seguir de este proyecto será el siguiente:

Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos	% completado
	Diseño e implementacion de un prototipo de una red SDN basada en Openflow.	**61 días**	**vie 28/08/15**	**vie 20/11/15**		**Christian Valencia**	**100%**
	Preparación del proyecto	**15 días**	**vie 28/08/15**	**jue 17/09/15**		**Christian Valencia**	**100%**
	Planeación del proyecto	**8 días**	**vie 18/09/15**	**mar 29/09/15**	2	**Christian Valencia**	**100%**
	Diseño de la red	**7 días**	**mié 30/09/15**	**jue 08/10/15**	10	**Christian Valencia**	**100%**
	Implementacion del prototipo	**15 días**	**vie 09/10/15**	**jue 29/10/15**		**Christian Valencia**	**100%**
	Operación del prototipo	**7 días**	**vie 30/10/15**	**lun 09/11/15**		**Christian Valencia**	**100%**
	Optimizacion del prototipo	**9 días**	**mar 10/11/15**	**vie 20/11/15**		**Christian Valencia**	**100%**

GRAFICO # 17 Plan de trabajo del proyecto
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

La escala de tiempo del proyecto es la siguiente:

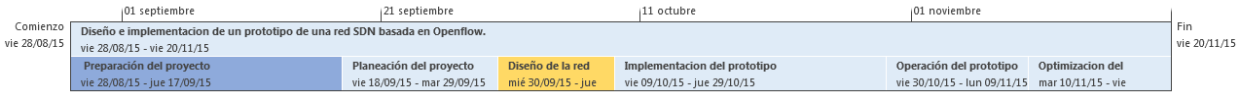


GRAFICO # 18 Línea de tiempo del plan de trabajo.
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

En el Anexo #7 se encontrará el documento original del plan del proyecto.

DESIGN: En esta fase de la metodología PPDIOO, se realiza el diseño de la red según los requerimientos recolectados en las fases anteriores, las necesidades del negocio al cual el proyecto va a aplicar y detalla la especificación de los componentes del proyecto.

Para este proyecto se ha desarrollado el siguiente diseño de red:

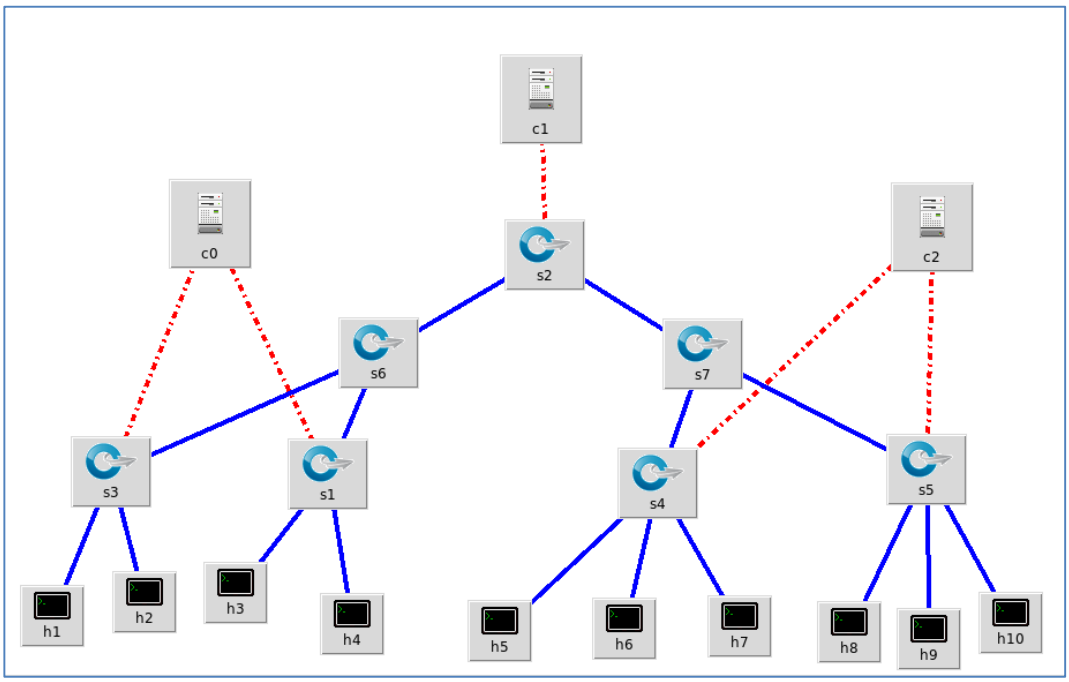


GRÁFICO # 19 Diseño de red propuesto para el proyecto.
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Este diseño de red tiene las siguientes características:

- Posee alta disponibilidad en la red SDN, con equipos redundantes en el plano de control y en el plano de datos.
- Posee escalabilidad, ya que se puede agregar nuevos elementos virtuales a la red creada.
- Posee tolerancia a fallos de red.

La red diseñada posee los siguientes componentes:

Controlador: el controlador distribuido que se usará es el controlador por defecto de Openflow, este controlador proporciona las características básicas para la implementación del prototipo.

Switches: El switch a usar es el Openflow Reference Switch.

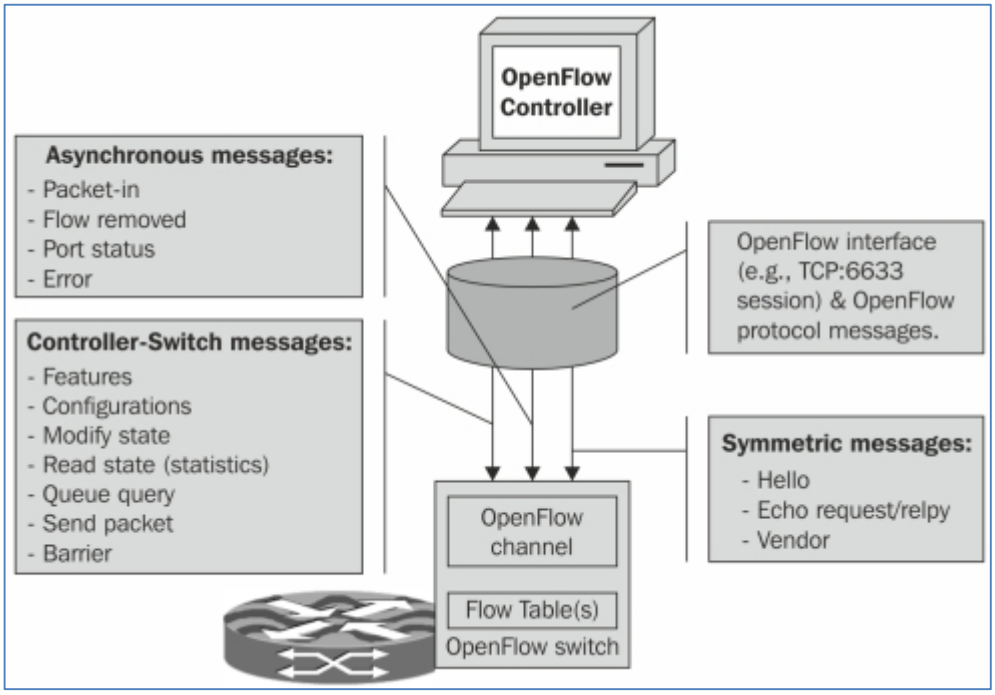


GRÁFICO # 20 Diseño del Openflow Reference Switch
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Hosts: se usarán maquinas virtualizadas dentro del prototipo.

Implementation: En esta fase, encontramos los detalles de la instalación y configuración de las herramientas y el diseño de red; por lo que procedemos a describir la instalación de cada uno de los elementos del proyecto.

Ubuntu 14.04

- Descargar la imagen correspondiente a 64 bits desde Ubuntu.com.

- Cargar la ISO en VirtualBox y realizar una instalación normal con interfaz gráfica

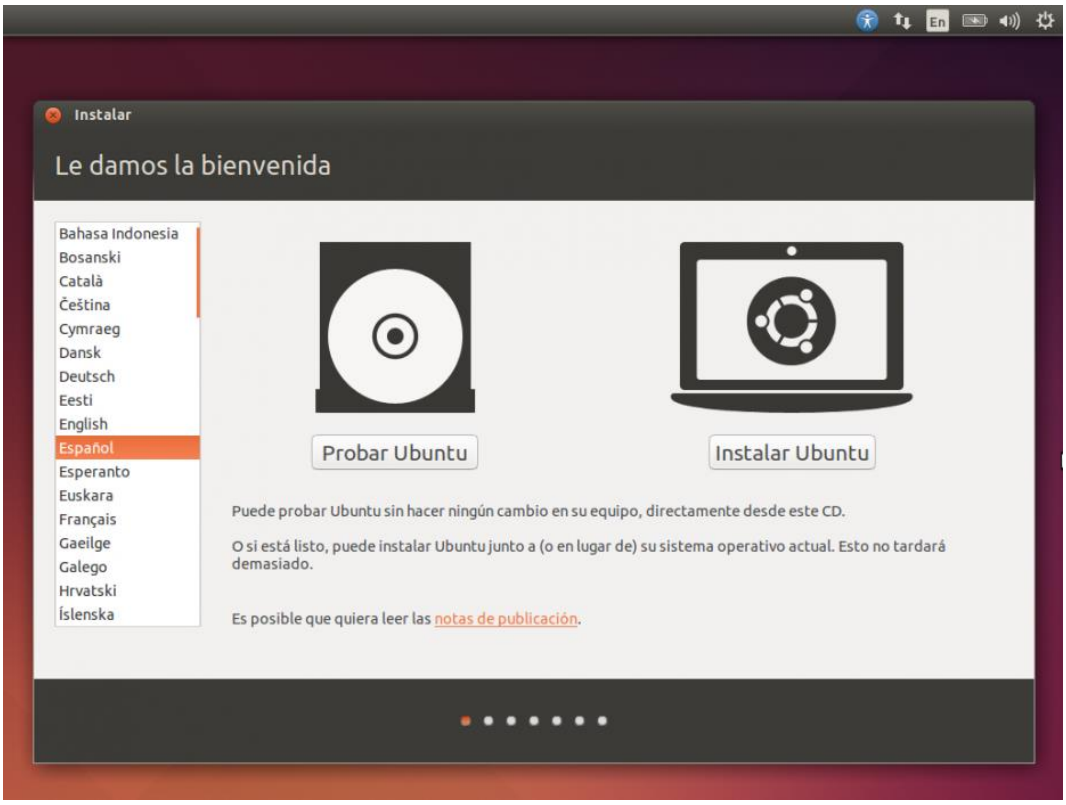


GRÁFICO # 21 Pantalla inicial de instalación de Ubuntu
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

- Una vez instalado, verifica la correcta funcionalidad del sistema operativo virtualizado

Mininet 2.2.0

- Entrar como usuario privilegiado al ambiente del sistema operativo Ubuntu.
- Ejecutar los siguientes comandos:
    - `git clone git://github.com/mininet/mininet`
    - `git checkout -b 2.2.0b3`
    - `~/mininet/util/install.sh -a`

- Una vez instalado, comprobamos que funcione correctamente haciendo pruebas de ping en una topología de pruebas.

```
mininet@mininet-vm:~$ sudo mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.427 seconds
mininet@mininet-vm:~$
```

GRÁFICO # 22 Prueba de funcionalidad de Mininet 2.2.0
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Opendaylight

- Como usuario privilegiado, ejecutamos los siguientes comandos:
- Para instalar la máquina virtual JAVA

```
sudo apt-get install openjdk-7-jdk
```

- Instalamos Maven para construir a partir del código fuente el software Opendaylight

```
sudo mkdir -p /usr/local/apache-maven
wget http://ftp.wayne.edu/apache/maven/maven-3/3.3.3/binaries/apache-maven-3.3.3-bin.tar.gz
sudo mkdir /usr/local/apache-maven
sudo mv apache-maven-3.3.3-bin.tar.gz /usr/local/apache-maven
sudo tar -xzvf /usr/local/apache-maven/apache-maven-3.3.3-bin.tar.gz -C /usr/local/apache-maven/
sudo update-alternatives --install /usr/bin/mvn mvn /usr/local/apache-maven/apache-maven-3.3.3/bin/mvn 1
sudo update-alternatives --config mvn
sudo apt-get install vim
vim ~/.bashrc
```

- Añadimos variables de ambiente al usuario

```
export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.3
export MAVEN_OPTS="-Xms256m -Xmx512m"
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

- Instalamos GIT

```
sudo apt-get install git
git clone
https://github.com/opendaylight/integration/distribution.git
```

- Paramos los servicios de Open vSwitch, si están corriendo

```
sudo service openvswitch-controller stop
sudo service openvswitch-switch stop
```

- Instalamos Opendaylight

```
cd distributions/karaf/target/assembly/bin

./karaf -of13
```

Una vez instalado, podremos ver la siguiente pantalla:

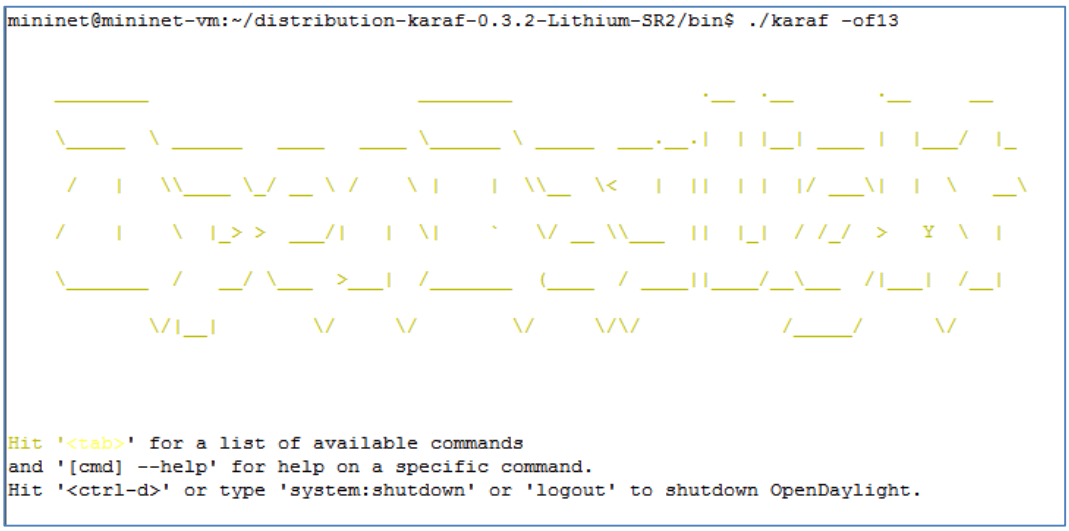


GRÁFICO # 23 Prueba de funcionalidad de Opendaylight
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Además, para acceder a la interfaz web de Opendaylight, nos dirigimos a la URL http://ip_del_servidor:8181/index.html

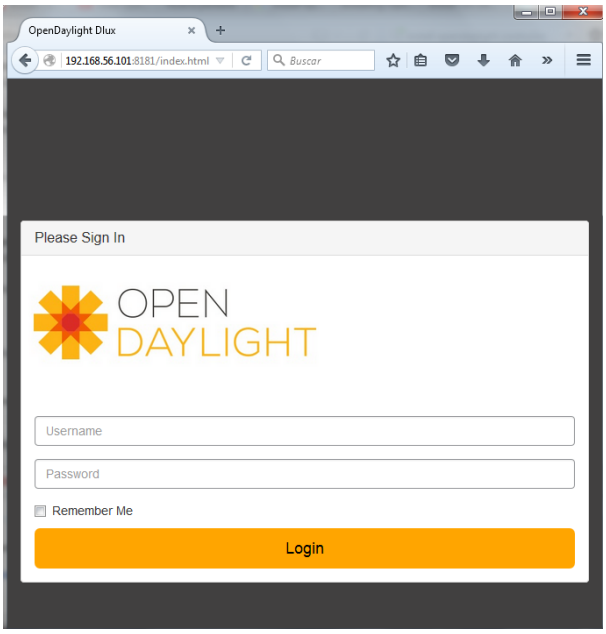


GRÁFICO # 24 Interfaz Opendaylight
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Creación del diseño en MiniEdit

MiniEdit es una herramienta de interfaz gráfica para poder crear elementos de red virtuales en Mininet, nuestro hipervisor de red.

Para el prototipo que se está desarrollando, se ha tomado en cuenta una red tipo general, que contenga suficientes elementos que puedan mostrar la tecnología SDN.

Se han añadido al diseño los siguientes elementos:

- 3 controladores Openflow
- 7 switches Openflow
- 10 hosts

Se eligió la topología tipo árbol, ya que permite realizar configuraciones interesantes sin mayores conocimientos de la red.

Se configuraron los controladores, los switches y los hosts de la siguiente manera:

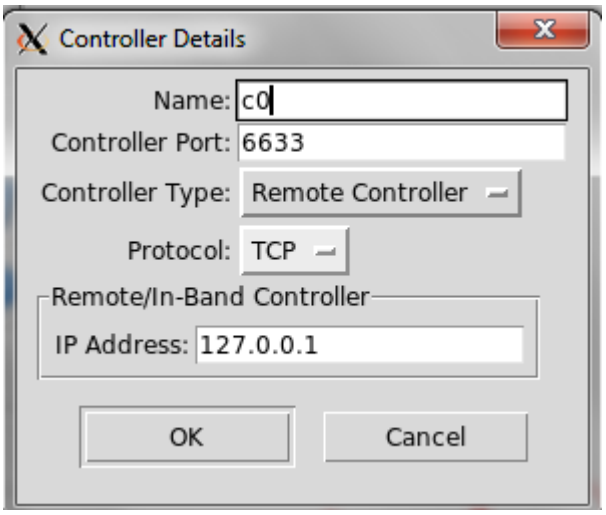


GRÁFICO # 25 Configuración de controladores
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

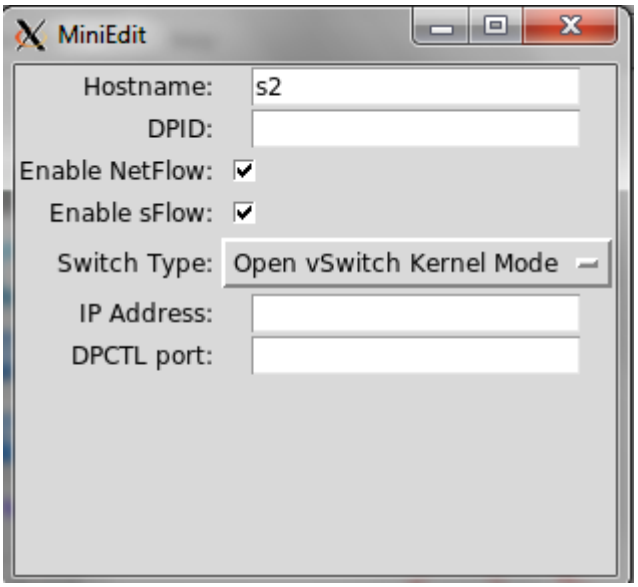


GRÁFICO # 26 Configuración de switches
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

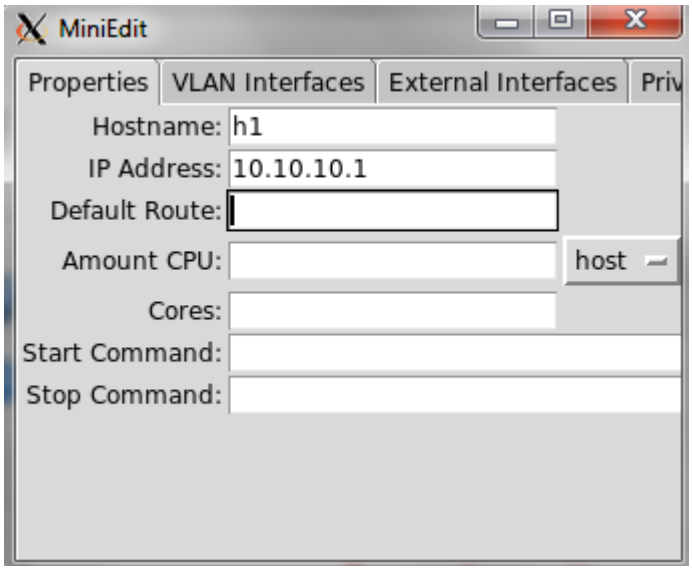


GRÁFICO # 27 Configuración de hosts
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Ejecución del prototipo en Mininet

Una vez realizado el diseño, MiniEdit presenta opciones para ejecutar el diseño sobre el hipervisor virtual, así mismo se pueden realizar las mismas tareas sobre la línea de comandos, como lo podemos observar en la siguiente imagen

```
mininet@mininet-vm:~$ sudo ./custom_mininet.py
*** Creating nodes
Unable to contact the remote controller at 127.0.0.1:6634
Unable to contact the remote controller at 127.0.0.1:6635
*** Creating links
*** Starting network
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Running CLI
*** Starting CLI:
mininet> exit
*** Stopping network
*** Stopping 3 controllers
c19 c20 c21
*** Stopping 12 links
............
*** Stopping 7 switches
s11 s13 s14 s15 s16 s17 s18
*** Stopping 10 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Done
mininet@mininet-vm:~$
```

GRÁFICO # 28 Uso de línea de comandos de Mininet
Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Manejo de la infraestructura SDN por medio de Opendaylight

En la interfaz gráfica de Opendaylight, una vez que se configura la red virtual para que se conecte al controlador remoto, aparecen automáticamente los controladores, listos para ser manejados por Opendaylight

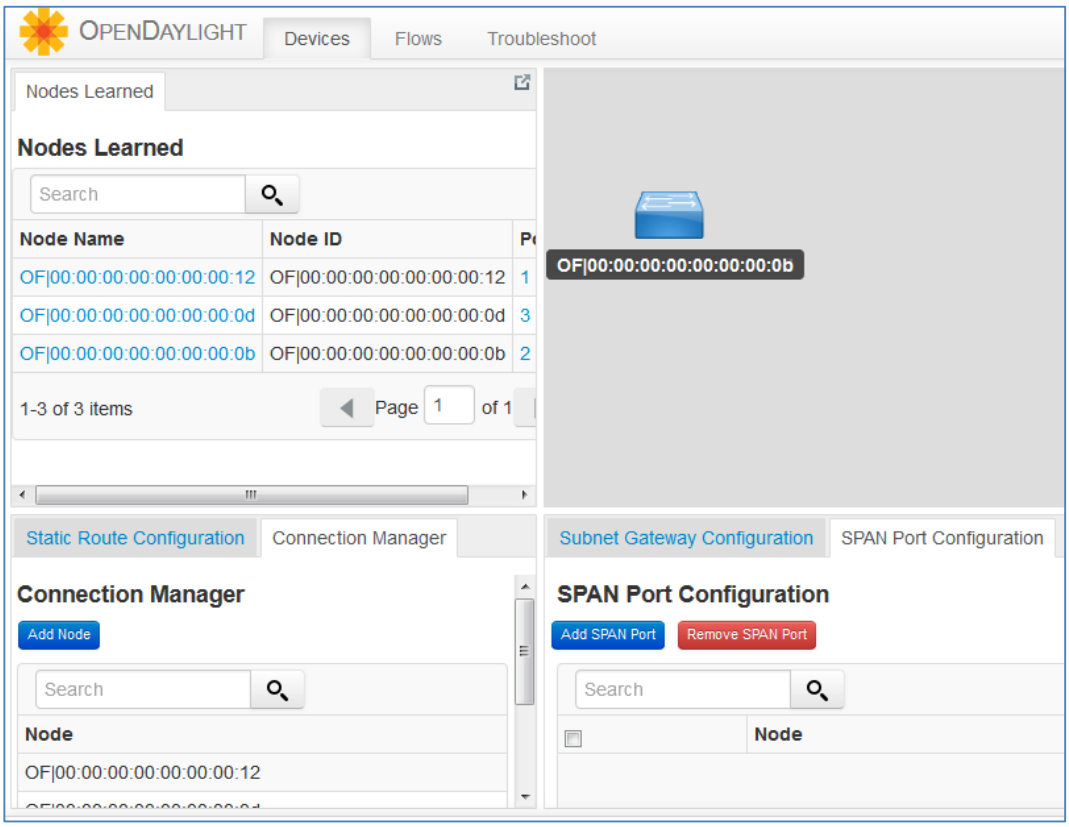


GRÁFICO # 29 Manejo de red SDN con Opendaylight

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Entregables del proyecto

Pare este proyecto, los entregables serán:

- CD con diseño de red.
- CD con las herramientas usadas y configuradas para implementar el prototipo.

Criterio de validación de la propuesta

Informe de pruebas: Este informe contiene el detalle de las pruebas realizadas al prototipo para asegurar su correcto funcionamiento y cumplimiento de los objetivos señalados:

CUADRO # 7 INFORME DE PRUEBAS

INFORME DE PRUEBAS			
PROYECTO: DISEÑO E IMPLEMENTACION DE UN PROTOTIPO DE UNA RED SDN USANDO OPENFLOW			
PRUEBAS DE HARDWARE			
		CUMPLE	NO CUMPLE
PRUEBA 1:	HARDWARE DE IMPLEMENTACION CUMPLE LOS REQUISITOS REQUERIDOS	X	
PRUEBA 2:	HARDWARE DE IMPLEMENTACION SE ENCUENTRA DENTRO DEL PRESUPUESTO INDICADO	X	
PRUEBA 3:	HARDWARE DE IMPLEMENTACION SE ENCUENTRA DISPONIBLE AL MOMENTO DEL PROYECTO	X	
PRUEBA 4:	HARDWARE DE IMPLEMENTACION FUNCIONA CORRECTAMENTE ANTES DEL INICIO DEL PROYECTO	X	
PRUEBA 5:	HARDWARE DE IMPLEMENTACION FUNCIONA CORRECTAMENTE DESPUES DEL PROYECTO	X	
PRUEBA 6:	HARDWARE DE IMPLEMENTACION ES COMPATIBLE CON LOS REQUERIMIENTOS DEL PROYECTO	X	
PRUEBAS DE SOFTWARE			
PRUEBA 1:	SISTEMA OPERATIVO: UBUNTU CUMPLE CON REQUISITOS PARA LA IMPLEMENTACION	X	
PRUEBA 2:	SISTEMA OPERATIVO: UBUNTU CUMPLE CON LA CONFIGURACION REQUERIDA PARA LA IMPLEMENTACION	X	
PRUEBA 3:	HIPERVISOR: MININET CUMPLE CON LOS REQUISITOS PARA LA IMPLEMENTACION	X	
PRUEBA 4:	HIPERVISOR: MININET CUMPLE CON LA CONFIGURACION REQUERIDA PARA LA IMPLEMENTACION	X	
PRUEBA 5:	HIPERVISOR: MINIEDIT CUMPLE CON LOS REQUISITOS PARA LA IMPLEMENTACION	X	
PRUEBA 6:	HIPERVISOR: MINIEDIT CUMPLE CON LA CONFIGURACION REQUERIDA PARA LA IMPLEMENTACION	X	
PRUEBA 7:	CONTROLADOR: OPENDAYLIGHT CUMPLE CON LOS REQUISITOS PARA LA IMPLEMENTACION	X	
PRUEBA 8:	CONTROLADOR: OPENDAYLIGHT CUMPLE CON LA CONFIGURACION REQUERIDA PARA LA IMPLEMENTACION	X	

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

Esta matriz de pruebas ha sido validada por el Ingeniero Carlos Beltrán, experto en redes de comunicaciones CISCO y que labora en el área de referencia para este proyecto.

CAPÍTULO IV

Criterio de aceptación para nuestro proyecto de “Diseño y simulación de un prototipo de red definida por software (SDN) usando el protocolo Openflow”

Como parte del proyecto presentado en este documento, detallamos los criterios que deben considerarse para analizar la validez de estas ideas:

- Es un proyecto que expone un tema no conocido por los estudiantes de la Universidad de Guayaquil.
- La realización del proyecto de manera didáctica para su fácil entendimiento
- La entrega del diseño base para su análisis por parte de los interesados
- La entrega de las herramientas que se usaron para este proyecto para su replicación por parte de otros interesados en las redes SDN.
- La posibilidad de modificación del proyecto actual según sea necesario

Requisitos y criterios de aceptación de la nueva tecnología

MATRIZ DE REQUISITOS Y CRITERIOS

CUADRO # 8 MATRIZ DE REQUISITOS Y CRITERIOS

Requisito	Criterios
Cubrir las necesidades de la población a la que está dirigida	Se adjunta encuesta y sus resultados de las consultas realizadas a los estudiantes de la Universidad de Guayaquil.
Uso de herramientas de software libre	La parte central del proyecto se basa en el protocolo Openflow, el cual es software libre.
Simulación del proyecto realizado.	Se entrega un diseño en DDVROM para su uso.

Elaboración: Christian Valencia Echeverría
Fuente: Christian Valencia Echeverría

ANEXO VIII

FORMULA DE LAS MUESTRAS TOMADAS PARA LAS ENCUESTAS

Para calcular el tamaño de la muestra suele utilizarse la siguiente fórmula:

$$n = \frac{N\sigma^2 Z^2}{(N-1)e^2 + \sigma^2 Z^2}$$

Donde:

n = el tamaño de la muestra.

N = tamaño de la población.

σ =Desviación estándar de la población que, generalmente cuando no se tiene su valor, suele utilizarse un valor constante de 0,5.

Z = Valor obtenido mediante niveles de confianza. Es un valor constante que, si no se tiene su valor, se lo toma en relación al 95% de confianza equivale a 1,96 (como más usual) o en relación al 99% de confianza equivale 2,58, valor que queda a criterio del investigador.

e = Límite aceptable de error muestral que, generalmente cuando no se tiene su valor, suele utilizarse un valor que varía entre el 1% (0,01) y 9% (0,09), valor que queda a criterio del encuestador.

La fórmula del tamaño de la muestra se obtiene de la fórmula para calcular la estimación del intervalo de confianza para la media, la cual es:

$$\bar{X} - Z\frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}} \leq \mu \leq \bar{X} + Z\frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}}$$

De donde el error es:

$$e = Z\frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}}$$

De esta fórmula del error de la estimación del intervalo de confianza para la media se despeja la n, para lo cual se sigue el siguiente proceso:

Elevando al cuadrado a ambos miembros de la fórmula se obtiene:

$$(e)^2 = \left(Z\frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}}\right)^2$$

$$e^2 = Z^2\frac{\sigma^2}{n}\frac{N-n}{N-1}$$

Multiplicando fracciones:

$$e^2 = \frac{Z^2\sigma^2(N-n)}{n(N-1)}$$

Eliminando denominadores:

$$e^2n(N-1) = Z^2\sigma^2(N-n)$$

Eliminando paréntesis:

$$e^2nN - e^2n = Z^2\sigma^2N - Z^2\sigma^2n$$

Transponiendo n a la izquierda:

$$e^2nN - e^2n + Z^2\sigma^2n = Z^2\sigma^2N$$

Factor común de n:

$$n(e^2N - e^2 + Z^2\sigma^2) = Z^2\sigma^2N$$

Despejando n:

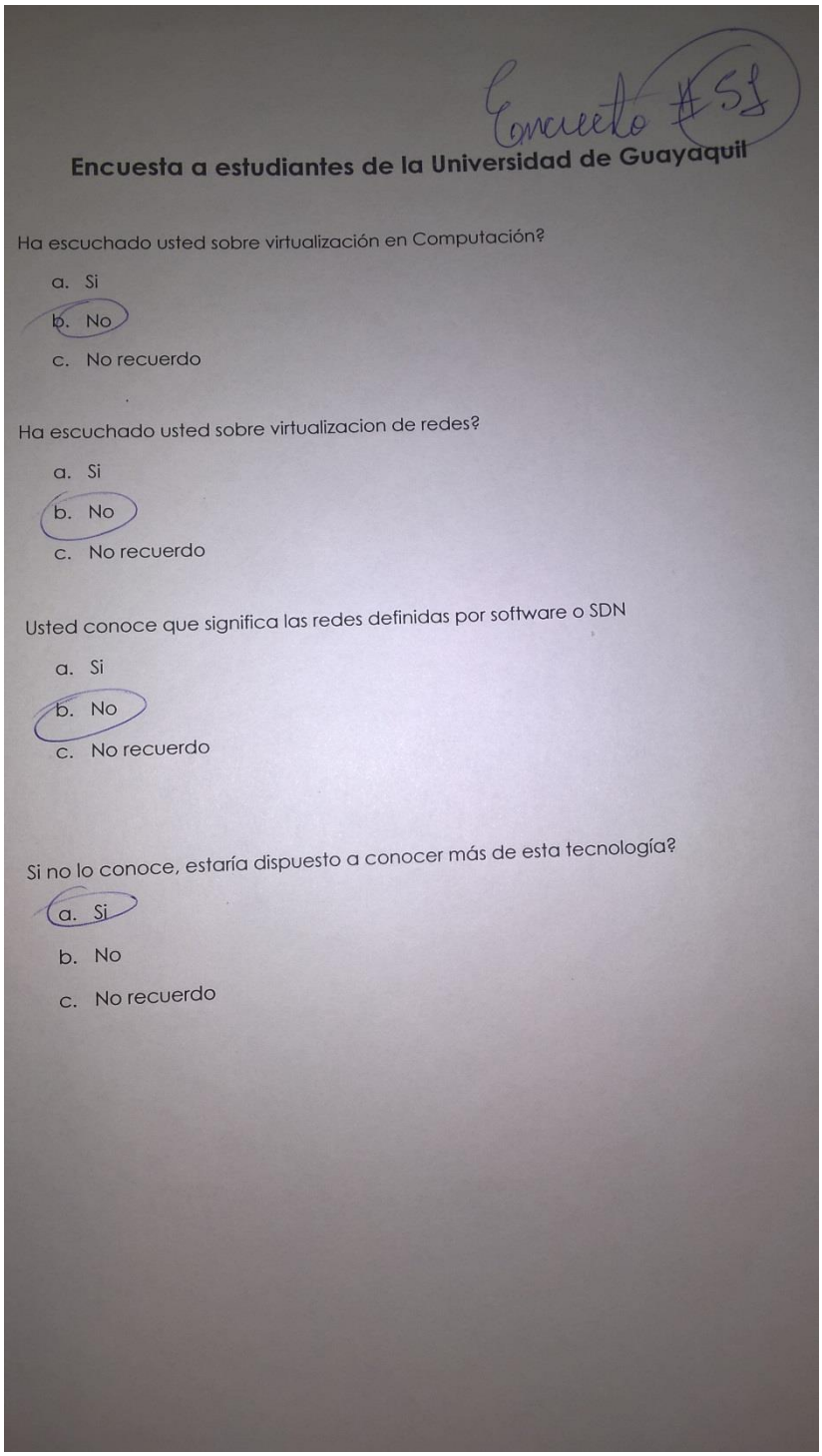
$$n = \frac{Z^2\sigma^2N}{e^2N - e^2 + Z^2\sigma^2}$$

Ordenando se obtiene la fórmula para calcular el tamaño de la muestra:

$$n = \frac{Z^2 \sigma^2 N}{e^2 (N-1) + Z^2 \sigma^2}$$

ANEXO IX

FOTOS DE LAS ENCUESTAS



Encuesta #51

Encuesta a estudiantes de la Universidad de Guayaquil

Ha escuchado usted sobre virtualización en Computación?

a. Si
b. No
c. No recuerdo

Ha escuchado usted sobre virtualizacion de redes?

a. Si
b. No
c. No recuerdo

Usted conoce que significa las redes definidas por software o SDN

a. Si
b. No
c. No recuerdo

Si no lo conoce, estaría dispuesto a conocer más de esta tecnología?

a. Si
b. No
c. No recuerdo

ANEXO X

FOTOS DE LAS PRUEBAS REALIZADAS