INSTITUTO UNIVERSITARIO AERONÁUTICO

FACULTAD DE INGENIERÍA

TRABAJO FINAL DE GRADO

INGENIERÍA EN TELECOMUNICACIONES

"Medición y obtención del rendimiento y capacidades máximas del Switch HP 2920-24G OpenFlow mediante test y pruebas de estrés."

Autor: Emiliano Ortin Calabrese

Tutor: Ingeniero Juan Galleguillo

Año 2016

Dedicatoria

A mi mamá, Rosa, único pilar irremplazable de este logro.

A mi hermana, Melina, compañera incondicional.

A mi papá, Adrian, por su apoyo a la distancia.

A mis amigos, raíces, por hacer de los estudios un camino más fácil.

Emiliano Ortin Calabrese

Agradecimientos

Al Ing. Juan Galleguillo por su afectuoso trato desde el primer día en esta universidad.

Al Ing. Gastón "Fito" Borja por su ayuda y trabajo en equipo.

A mi novia Vicky por su apoyo y compañía. A mi familia por siempre estar.

Emiliano Ortin Calabrese

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

"La mente es como un paracaídas, sólo funciona si la tenemos abierta."

Albert Einstein

Glosario

- **ACL:** Access Control List. (Lista de control de acceso)
- **AH:** Authentication Header. (Cabecera de autenticación)
- **API:** Application Programming Interface. (Interfaz de aplicación)
- **ARP:** Address Resolution Protocol. (Protocolo de resolución de dirección)
- **BGP:** Border Gateway Protocol. (Protocolo de puerta de enlace fronteriza)
- **Framework**: Infraestructura digital. Conjunto estandarizado de conceptos prácticas y criterios para enfocar una problemática.
- **FTP:** File Transfer Protocol. (Protocolo de transferencia de archivo)
- **GUI:** Grafical User Interfaz (Interfaz Gráfico de usuario)
- **HTTP:** Hypertext Transfer Protocol. (Protocolo de transferencia de hipertexto)
- **HTTPS:** Hypertext Tranfer Protocol Secure.(Protocolo de transferencia de hipertexto seguro)
- **ICMP:** Internet Control Message Protocol. (Protocolo de mensaje de control de internet)
- **IP:** Internet Protocol. (Protocolo de internet)
- **IS-IS:** Intermedia System to Intermedia System.
- **ISP:** Internet Service Provider. (Proveedor de internet)
- **LAN:** Local Area Network. (Red de area local)
- **LLDP:** Link Layer Discovery Protocol. (Protocolo de descubrimiento de capa de enlace)
- **MAC:** Media Access Control.
- **NetConf:** Network Configuration Protocol. (Protocolo de configuración de red)
- **NFV:** Network Function Virtualization. (Virtualización de funciones de red)
- **NIB:** Network Information Base.
- **OF:** OpenFlow.

- **OFA:** OpenFlow Application.
- **OFC:** OpenFlow Controller.
- **ONF:**La Open Network Foundation (ONF), una organización guiada por usuarios con el objetivo de fomentar el desarrollo de la SDN.
- **OOB:** Out of Band.
- **OSPF:** Open Shortest Path First.
- **OVS:** Open vSwitch.
- **QoS:** Quality of Service. (Calidad de servicio)
- **SDN:** Redes definidas por software o SDN (del inglés Software Defined Network)
- **SNMP:** Simple Network Management Protocol.
- **SO:** Sistema Operativo.
- **SSH:** Secure Shell.
- **STP:** Spanning Tree Protocol.
- **TCP:** Transmission Control Protocol. (Protocolo de control de transmisión)
- **Threads:** Tareas.
- **TLS:** Transport Layer Security. (Seguridad en capa de transporte)
- **TTL:** Time to Live. (Tiempo de vida)
- **VLAN:** Virtual Local Area Network. (Red de area local virtual)
- **VM:** Virtual Machine. (Máquina Virtual)
- **VoIP:** Voice over IP.
- **WAN:** Wide Area Network.

Índice de ilustraciones

Índice

1. Redes definidas por software

1.1 Introducción

Las redes definidas por software (SDN) son una manera de abordar la creación de redes en la cual el control se desprende del hardware y se le da a una aplicación de software llamada controlador.

Redes definidas por software o SDN (del inglés Software Defined Network) es un conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación e implantación de servicios de red de una manera determinista, dinámica y escalable, evitando al administrador de red, gestionar dichos servicios a bajo nivel. Todo esto se consigue mediante la separación del plano de control (*software*) del plano de datos (*hardware*).

La Open Network Foundation (ONF), una organización guiada por usuarios con el objetivo de fomentar el desarrollo de la SDN, lo describe como:

"La separación física del plano de control de la red del plano de reenvío, y donde el plano de control controla varios dispositivos"

Cuando un paquete llega a un conmutador en una red convencional, las reglas integradas al *firmware* propietario del conmutador le dicen al conmutador adónde transferir el paquete. El conmutador envía cada paquete al mismo destino por la misma trayectoria – y trata a todos los paquete de la exacta misma manera.

En una red definida por software, un administrador de red puede darle forma al tráfico desde una consola de control centralizada sin tener que tocar conmutadores individuales. El administrador puede cambiar cualquier regla de los conmutadores de red cuando sea necesario – dando o quitando prioridad, o hasta bloqueando tipos específicos de paquetes con un nivel de control muy detallado.

Al entender este concepto de la separación del plano de control y el plano de datos, podemos imaginar a la red como un gran conjunto de dispositivos de conmutación (switch, routers, etc.), conectados entre sí por líneas de transmisión y, separado de estos, la capa de control por encima compuesta por un controlador que es el "cerebro" de la red y toma las decisiones sobre los paquetes que circulan por ella.

1.2 Arquitectura

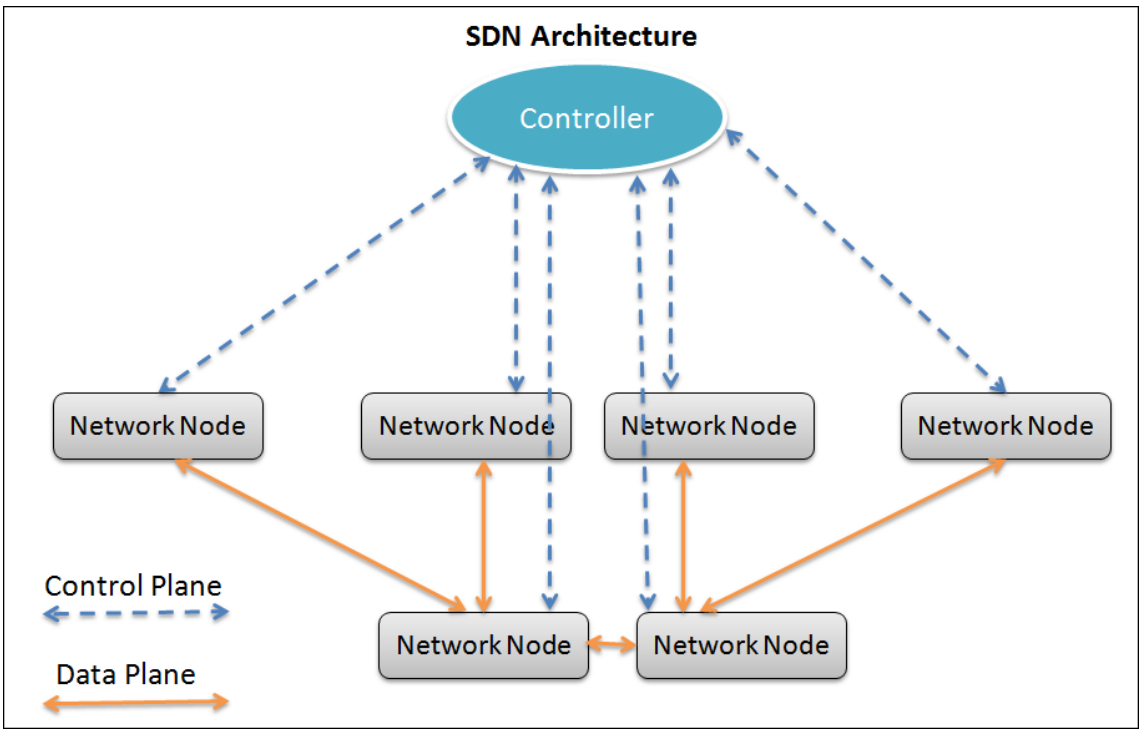


Ilustración 1: Visión básica SDN

Dentro de las características que presenta esta nueva arquitectura, y que sacan ventaja a las redes convencionales, se encuentran[1] :

Gestión centralizada: Permite gestionar la totalidad de la red desde un único punto debido a que toda la inteligencia de la red se encuentra concentrada en el controlador. Este tiene una visión global de la red y de todos los elementos que se encuentran en ella. Si se desea realizar algún cambio en la red, solo con informárselo al controlador ya tendremos el problema resuelto. Nos ofrece

soluciones con respecto a problemas que tienen las redes tradicionales, eliminando fallas de seguridad, cuellos de botellas, errores de configuración, etc. que traen consigo los múltiples puntos de decisión.

Agilidad: Se puede reservar ancho de banda para servicios especiales o QoS (Calidad de Servicio) con mayor rapidez y adaptarse a los cambios y prioridades de la red. La ventaja de esto es que, programando el controlador con anterioridad, podemos ir cambiando en "caliente" estas características y que la red vaya mutando a medida que van entrando paquetes nuevos con diferentes prioridades a la hora de ser transportados.

Bajos costos: Se considera esto al pensar en redes grandes que al no tener que optar por la homogeneidad de productos y tener un proveedor genérico, abaratan los costos de la red. Muchos de los controladores son open-source y presentan las mismas o más características que los controladores propietarios, por ende, no se debe pagar licencias para poder tener control sobre la red.

Programación automática: Software Defined Networks permite a los administradores, por medio de programas automatizados que ellos mismos pueden escribir ya que no depende de software propietario, gestionar y optimizar los recursos de la red de forma rápida y dinámica. Esto se debe a que se pueden ligar balanceadores de carga, firewalls o cualquier tipo de aplicación para mantener control sobre la red a los controladores por medio de API's. Estos programas analizan constantemente la red y van informando al controlador sobre el estado de la misma y los requerimientos de cada paquete o enlace para que este mantenga un rendimiento constante y libere al administrador de la red de esta tarea.

Fácilmente programable: Brinda la posibilidad de programar o configurar el plano de control con mayor facilidad al estar separado del plano de red. La mayoría de los controladores responden a lenguajes de programación muy usados como Python o Java, lo que hace que configurarlos y manejarlos resulte muy cómodo, y al estar separado del plano de datos, se pueden realizar cambios sin cortar el funcionamiento de la red.

Escalabilidad: Con esta arquitectura, es muy fácil proyectar una red que crezca en gran medida sin perder ninguna de sus características. Con solo conectar el nuevo switch al controlador, este se encargara de su configuración y de su puesta a punto para que realice sus funciones y monitoree su rendimiento. Un solo controlador puede administrar muchos switches al mismo tiempo y se pueden usar más de un controlador en la red para tener redundancia por si uno llega a fallar.

Basado en estándares abiertos y proveedores neutrales: Como se implementa a través de estándares abiertos, simplifica el diseño de la red y las operaciones porque las instrucciones son proporcionadas por el controlador, en lugar de múltiples dispositivos, proveedores específicos o protocolos. Ya no se depende de los protocolos o estándares a los que uno estaba restringido cuando se adquiere un switch Cisco, HP, Juniper, etc. Ahora uno es capaz de crear sus propios protocolos y decidir el futuro de los paquetes de la red de acuerdo a las necesidades actuales.

Seguridad: Al estar la red centralizada por el controlador, no se corre el riesgo de poseer agujeros de seguridad en las configuraciones de los switches. Solo hay un punto de acceso a la configuración de la red y es el controlador.

Todas estas características que poseen las SDN y no las redes tradicionales nos llevan a pensar que en un futuro, no muy lejano, esta arquitectura será ampliamente aceptada por todos los sectores del mercado de telecomunicaciones, favoreciendo tanto a los clientes que buscan velocidad, seguridad, y robustez en sus conexiones, como a las empresas que buscan brindar un servicio de alta calidad y confiabilidad con el menor costo de operación y de inversión posible.

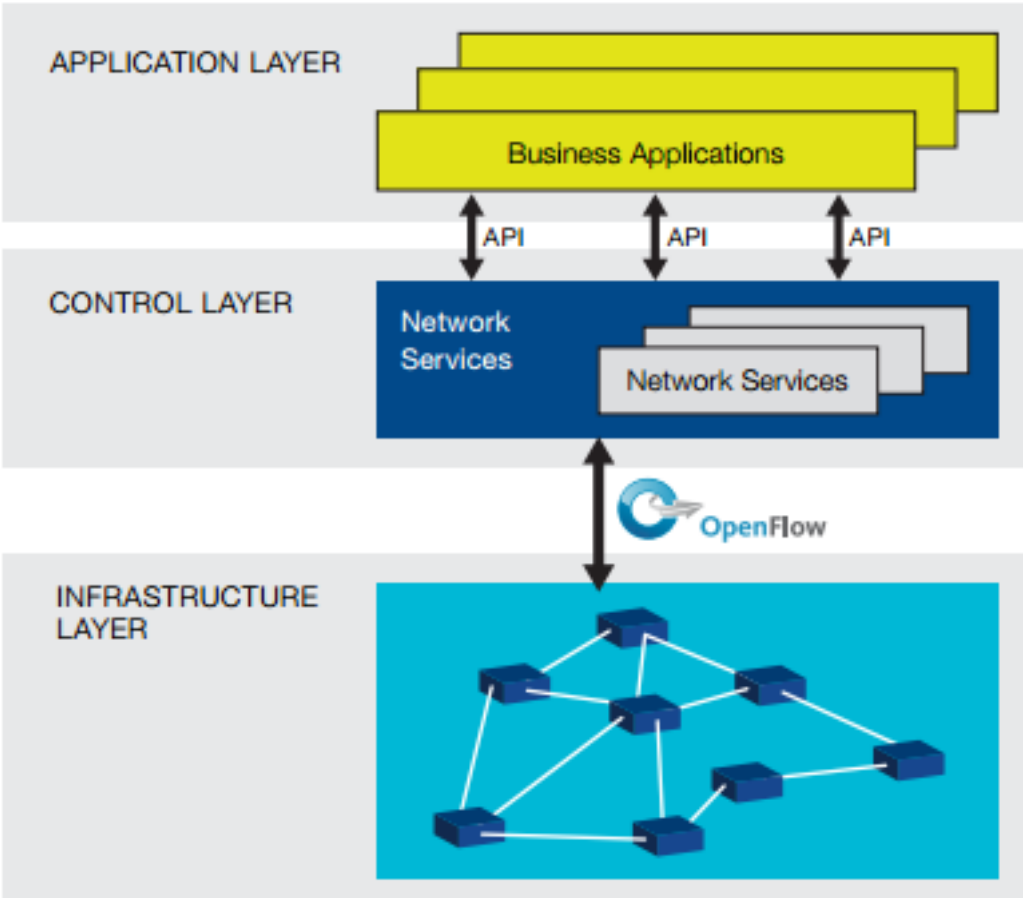


Ilustración 2: Arquitectura SDN

1.2.1 Capa de Aplicación

Las aplicaciones SDN comunican sus requisitos a la red a través de una API que conecta con la capa de control, y están diseñadas para satisfacer las necesidades de los usuarios. Ejemplos de aplicaciones SDN:

• **Enrutamiento adaptativo**. Tradicionalmente el enrutamiento conlleva una compleja implementación y convergencia lenta. Con SDN han surgido dos conceptos populares: balance de carga, a través de diversas propuestas de algoritmos que solucionen el alto coste de balanceadores de carga dedicados, y el diseño de intercambio de información entre capas, para mejorar la integración entre entidades y proporcionar calidad de servicio (QoS), con Qos-aware Network Operating System (QNOX) como principal ejemplo.

• **Itinerancia sin interrupciones.** La transferencia o handover al hacer uso de dispositivos móviles hace necesario prever un servicio continuo. Con SDN, las

redes entre diferentes portadores con diferentes tecnologías tienen un plano de control común. Hay diferentes propuestas de transferencia como Hoolock en redes Wi-Fi y WiMAX u Odin para WLANs de empresa.

• **Mantenimiento de la red.** Herramientas de configuración típicas como traceroute o tcpdump no son una solución para mantener una red extensa de forma automática, ya que, como se ha citado, tienden al error humano. Al tener una visión global de la red y un control centralizado de la configuración, SDN permite nuevas herramientas de diagnóstico como ndb u OFRewind.

• **Seguridad de la red.** Las redes tradicionales utilizan cortafuegos o servidores proxy para proteger las redes físicas, siendo su implementación una tarea pesada para el administrador. SDN permite analizar patrones de tráfico para posibles problemas de seguridad como ataques de denegación de servicio, guiar paquetes sospechosos a sistemas de prevención de intrusión (IPS), modificar reglas de reenvío para bloquear tráfico, o dar privacidad a los usuarios con ejemplos como AnonyFlow.

• **Virtualización de la red.** Lo que se pretende es permitir la existencia en una infraestructura compartida de múltiples arquitecturas de red heterogéneas. Normalmente, lo que se hace es separar la red física en múltiples instancias virtuales y asignarlas a diferentes usuarios, controladores o aplicaciones, mediante túneles o etiquetas VLAN y MPLS, convirtiéndose en una tarea compleja. Con SDN, la configuración se realiza en el controlador con plataformas como libNetVirt, una librería de virtualización de red o FlowVisor, colocando un proxy transparente para filtrar mensajes de control según la red virtual.

• **Cloud Computing.** Los centros de datos en las redes para 'cloud computing' necesitan algunas características, como escalabilidad, independencia de la localización para abastecer recursos dinámicos o diferenciación de QoS. La conmutación virtual es usada para la comunicación entre máquinas virtuales en el mismo host e implementada en SDN como Open vSwitch.

1.2.2 Southbound API

En una arquitectura SDN, el Interfaz de Programación de Aplicaciones (API) Southbound es usado para la comunicación entre el controlador SDN y los switches y routers de la red. Puede ser OpenSource o propietario. OpenFlow es el interfaz estándar definido por la ONF para la comunicación del plano de control con el de datos, permitiendo el acceso directo y la manipulación de las tablas de flujos de los dispositivos de red tanto físicos como virtuales. Aparte de OpenFlow existen otras southbound APIs como [31]:

- Border Gateway Protocol (BGP): Utilizado para intercambiar información de enrutamiento entre hosts de gateways en una red de sistemas autónomos. Se busca una utilidad en las SDN hibridas.
- NetConf: Es un protocolo de gestión de red de la Internet Engineering Task Force (IETF), es una forma segura de configurar un firewall, switch, router u otro dispositivo. Fue incorporado recientemente por la ONF y su uso se está volviendo obligatorio para la configuración de dispositivos compatibles con OF.
- Protocolo de Presencia y Mensajería Extensible (XMPP): Tiene uso en la mensajería instantánea y detección de presencia en la línea. Funciona entre o en los servidores y facilita la operación en tiempo real. Busca ser una alternativa a OF en SDN hibridas y su función sería la de la de asistir al controlador para la distribución de información de plano de control a los puntos finales de servidor.
- Protocolo de Gestión de Base de Datos OpenvSwitch (OVSDB): Es un protocolo de configuración OF destinado a administrar las implementación OpenvSwitch.

1.2.3 NorthBound API

La Northbound API es usada para comunicar el controlador SDN con los servicios y aplicaciones corriendo fuera de la red. Se usa para facilitar la

innovación y permitir una automatización y orquestación eficiente de la red para alinearse con las necesidades de las distintas aplicaciones que corren en la capa superior. Para este interfaz no hay un protocolo estandarizado y se está trabajando en la actualidad en ello. Muchos expertos en la materia consideran que la clave del éxito en SDN no se encuentra en OpenFlow sino en la Northbound API esto se debe a dos diferencias notables.

- Las Northbound API no dependen del hardware: no está atada al tiempo de desarrollo o innovación del hardware, si observamos la figura podemos ver que las dos puntas del interfaz une productos de software. En cambio, por ejemplo OpenFlow, posee versiones recientes que no todos los switches pueden soportar.

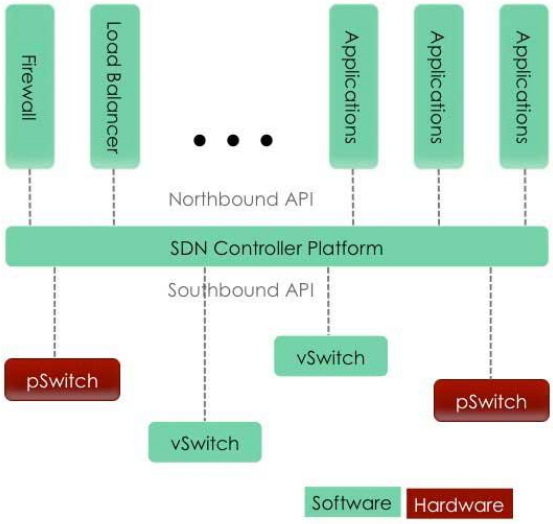


Ilustración 3: Ubicación de las Northbound APIs

- La mayor parte del valor de las SDN se creara y se capturara en la Capa de Aplicación: Serán las aplicaciones como firewalls, balanceadores de cargas, softwares de seguridad, etc., los que trabajaran para modificar la red según las necesidades de los usuarios, estas peticiones serán enviadas al controlador que modificara las configuraciones de los switches para que trabajen de la manera deseada.

Por esto es que muchas empresas ya están trabajando para proponer un estándar para este interfaz, como así también trabajan muchos proyectos Open-Source en conseguir la misma meta.

1.2.4 Capa de Control

Lógicamente centralizado, el controlador es el componente más importante de la arquitectura SDN ya que gestiona la capa de aplicación e infraestructura mediante dos interfaces, una con cada plano adjunto. Además, debe monitorizar todos los elementos para dar una visión global de la red.

Mediante la comunicación con el plano de infraestructura, se recoge el estado de la red y, según las exigencias de las aplicaciones, actualiza en los dispositivos las reglas de reenvío, ya que pueden producirse cambios como recuperación tras un fallo, migración de máquinas virtuales o balance de carga. También se debe mantener una validez y consistencia a fin de evitar bucles o agujeros de seguridad. Por otra parte, se comunica con las aplicaciones SDN mediante la "traducción" de sus requisitos con un lenguaje de alto nivel, teniendo varias opciones como lenguajes existentes (Python, Java, C++), librerías en un kit software de desarrollador (SDK) como OnePK de Cisco, o lenguajes nuevos como Flowbased Management Language (FML), Frenetic (similar a SQL) o Nettle. Además, el estado de la red (basado en número de paquetes, tamaño de los datos o ancho de banda en un flujo) que se recoge de la capa de infraestructura se debe comunicar a la aplicaciones para informarlas y construir, por ejemplo, un gráfico con la topología existente. En este sentido, la manera más común de hacerlo es mediante una Matriz de Tráfico (TM), que representa todos los flujos entre los posibles pares orígenes y destinos de la red.

En cuanto a la capa de control en sí, se necesita que un controlador sea capaz de comunicarse con otros debido a que:

• No se tiene un controlador estándar en SDN, y en una red grande pueden existir diversas opciones, por lo que es necesaria la transferencia de información para la visión global de la red y la toma de decisiones en la misma.

• Un solo controlador puede ocasionar problemas de congestión ya que se trata de un punto crítico. Se puede tener controladores de 'backup' o de réplica.

La solución más citada en este sentido es HyperFlow, que proporciona una vista sincronizada y consistente entre múltiples controladores.

Para finalizar, hay que destacar algunas características de los controladores:

• Pueden aparecer conflictos de configuración al haber una comunicación constante entre aplicaciones y controladores, por lo que hay diferentes soluciones propuestas como FlowChecker, NICE (basado en verificación de modelos, un método para sistemas formales) o VeriFlow.

1.2.5 Capa de Infraestructura

Esta capa consta de los dispositivos hardware de conmutación que forman una red y realizan dos tareas de acuerdo a sus dos componentes lógicos:

• Control. Recoge el estado de la red (topología o estadísticas de tráfico) y se lo comunica al controlador, el cual a su vez le indica las reglas de reenvío de paquetes. Esta información es almacenada temporalmente en la memoria local (como Memoria Direccionable de Contenido Ternario, TCAM o Memoria de Acceso Aleatorio Estático, SRAM), elemento fundamental de esta entidad lógica, ya que en caso de ser insuficiente los paquetes serían descartados o enviados directamente al controlador, con una obvia degradación de la red. Hay diferentes soluciones a este respecto como la agregación de las rutas (mediante un prefijo común) o algoritmos para optimizar la caché.

• Datos. El procesador de red reenvía los paquetes en base a las decisiones tomadas por el plano de control; al recibir un paquete, el dispositivo identifica la decisión de reenvío que coincide con el paquete. Además de realizar el proceso en base a las direcciones IP o MAC como en la redes tradicionales, en SDN puede ser también basándose en puertos TCP/UDP, etiqueta VLAN o puerto de entrada en el conmutador.

Es importante destacar el comportamiento de un dispositivo SDN al recibir un paquete: el primer paquete de un flujo se envía al controlador para su procesado e inspección de la cabecera del paquete, actualizando las tablas de reenvío. Una vez llegan más paquetes de dicho flujo, éstos se reenvían de acuerdo a la tabla, lo que se traduce en mayor sencillez al fabricarlos y menor coste.

1.2.6 Controladores de código abierto

Ilustración 4: Controlador NOX

NOX[3] : Es uno de los primeros controladores creados para redes OpenFlow, fue escrito por la empresa Nicira, y desde su lanzamiento bajo la licencia GPL fue uno de los más usados para desarrollos e investigación en este campo. Se trata de una plataforma para la creación de aplicaciones de control de red y algunas de sus características son:

- Proporciona soporte y una API para OpenFlow 1.0. (No funciona con OpenFlow 1.3)
- Las reglas para el tráfico son escritas en C++.
- Está dirigido a las recientes distribuciones de Linux.
- Incluye componentes que le permiten realizar algunas acciones como: descubrimiento de la topología, aprendizaje de conmutación y el análisis de redes extendidas de conmutación.
- NOX proporciona una API de alto nivel para OpenFlow, así como a otras funciones de control de red.

A pesar de sus características, este controlador no es muy usado en la actualidad, su mayor defecto es que no es compatible con versiones de

OpenFlow mayores a 1.0 y su bajo rendimiento en comparación con controladores más nuevos.

Ilustración 5: Controlador POX

POX[4] : Desarrollado a partir de NOX, este controlador OpenFlow es uno de los más usados en la actualidad, siendo uno de los más fáciles de entender, muy completo en sus componentes y la posibilidad de poder programar el control de la red en el lenguaje de programación Python. A pesar de que no posee soporte para versiones mayores a OpenFlow 1.0 (Igual que NOX), es muy usado para realizar desarrollos e investigaciones en el campo de las SDN y es usado como punto de partida para muchos otros controladores que salieron luego. Algunas de sus características son:

- Interfaz OpenFlow basada en Python.
- Los componentes muestran la selección de ruta de acceso, detección de la topología, entre otras.
- Está dirigido específicamente a Linux, Mac OS y Windows.
- Interfaz gráfica y herramientas de visualización similares a las de NOX.

Ilustración 6: Controlador Floodlight

FloodLight[5]: Este controlador es capaz de trabajar tanto con switch virtuales como físicos, que sean capaces de trabajar con el protocolo OpenFlow. Basado en código abierto, bajo la licencia Apache y escrito en JAVA

provee una forma de poder modificar el comportamiento de la red. Posee un módulo principal que es el encargado de escuchar los paquetes OpenFlow y realizar la distribución de eventos y módulos secundarios que agregan funciones extras a las del módulo principal como detección de hosts, determinación de la topología, etc.

Algunas de las características de este controlador son:

- Puede trabajar tanto con switches virtuales como con switch físicos, siempre y cuando soporten el protocolo OpenFlow.
- No se necesitan grandes requerimientos para hacerlo funcionar.
- Está desarrollado por una gran comunidad abierta de desarrolladores que participan activamente para reparar errores, realizar investigaciones y buscar mejoras.
- Está diseñado para soportar grandes rendimientos.
- Soporta versiones mayores a OpenFlow 1.0
- Curva de aprendizaje baja comparado con otros controladores.

Ilustración 7: Controlador Beacon

Beacon[6]: Es un controlador OpenFlow rápido, multiplataforma, modular, basado en Java, compatible con la programación basada en threads y eventos. Algunas de las características más relevantes son:

- Estable: Beacon ha sido utilizado en muchos proyectos de investigación y algunas implementaciones de prueba. Actualmente es capaz de soportar 100 switches virtuales y 20 switches físicos en redes de datos experimentales.
- Multiplataforma: Está escrito en Java y se ejecuta en muchas plataformas, desde servidores Linux hasta teléfonos Android.

- Código abierto: Esta licenciado bajo una combinación de la licencia GPL11v.2 y la Licencia de la Universidad de Stanford.
- Dinámico: Los paquetes de código en Beacon pueden ser inicializados, detenidos, actualizados e instalados en tiempo de ejecución, sin interrumpir otros paquetes.
- De rápido desarrollo: Ya que es fácil de descargar y correr. Java y Eclipse simplifican el desarrollo y la depuración de sus aplicaciones.
- Rápido: Debido a su característica de multiproceso.
- Interfaz gráfica de usuario: Opcionalmente incorpora al servidor web Jetty Enterprise y un framework extensible con una interfaz de usuario personalizada.
- Amplia documentación: La documentación de Beacon incluye tutoriales y guías para ayudar a conseguir un funcionamiento productivo de inmediato, además de un foro de usuarios activos.

Ilustración 8: Controlador ODL

OpenDayLight[7]: Este controlador es un proyecto de código abierto que tiene su sede en la Fundación Linux pero es fruto de la colaboración entre varias empresas, entre ellas Cisco, Dell e Intel. La meta de este proyecto esacelerar la adopción de las redes definidas por software y crear una fundación sólida para la Virtualización de Funciones de Red (NFV). Este software también está escrito en Java. Sus características son:

- Escrito en Java y multiplataforma.
- Soporta distintas southbound APIs como OpenFlow (En todas sus versiones), BGP-LS, SNMP, etc.
- Capaz de adaptarse a una variedad muy grandes de topologías.
- Es el controlador más ampliamente documentado.

- Continuamente actualizado y con soporte permanente.
- Cuenta con interfaz gráfica de usuario web con posibilidad de realizar cambios en los switches muy fácilmente.

Ilustración 9: Controlador Trema

Trema[8]: Es un software completo y fácil de usar para el desarrollo de controladores OpenFlow que permite definir las reglas de control mediante los lenguajes de programación Ruby y C.

Posee bibliotecas y módulos que funcionan como interfaz con los switches. Una gran característica es que posee un emulador de red OpenFlow y no es necesario disponer de switches físicos o virtuales para poder probar las aplicaciones del controlador. Algunas características de este controlador son:

- Permite ser ejecutado en una red virtualizada, es decir, permite ser desarrollado en una computadora simple, con el uso de máquinas virtuales y sin necesidad de switches físicos.
- Permite especificar y construir topologías virtuales arbitrarias.
- El controlador desarrollado en una red virtualizada puede ser perfectamente implementado en una red real.
- Soporta GNU/Linux y ha sido probado en los siguientes entornos:
- Ruby 1.8.7 (1.9.x no está soportada al momento).
- Ubuntu (de 10.04 en adelante).
- Debian GNU / Linux 6.0 (i386/amd64).

Ilustración 10: Controlador Ryu

Ryu[9]: Es un controlador basado en componentes para SDN que nos ofrece un framework amplio y fácil de utilizar. Provee componentes de software con API's bien definidas que le facilitan la tarea a los desarrolladores de crear nuevas aplicaciones de gestión y control de red. Soporta varios protocolos para la gestión de dispositivos de red como OF, NetConf, OF-Config, etc. Con respecto a OpenFlow, Ryu soporta completamente las versiones 1.0, 1.1, 1.2, 1.3, 1.4 y 1.5 y las extensiones de Nicira. Todo el código es gratuito y disponible bajo la licencia Apache 2.0. Como dato adicional, "Ryu" en japonés significa "flujo".

En esta tabla podemos comparar algunas de las características que presentan los controladores nombrados.

Controlador	FloodLight	POX	Beacon	NOX	Trema	Ryu	ODL
Facilidad de instalación	FÁCIL	FÁCIL	REGULAR	FÁCIL	FÁCIL	FÁCIL	REGULAR
Simplicidad en el manejo de funcionalidades	FÁCIL	FÁCIL	REGULAR	FÁCIL	DIFÍCIL	REGULAR	DIFÍCIL
Facilidad de expansión	FÁCIL	FÁCIL	REGULAR	DIFÍCIL	FÁCIL	FÁCIL	REGULAR
Soporte de los desarrolladores	BUENO	BUENO	REGULAR	MALO	MALO	BUENO	BUENO
Módulos para el desarrollo de aplicaciones	BUENO	BUENO	REGULAR	MALO	REGULAR	BUENO	BUENO
Bien documentado	SÍ	SÍ	SÍ	NO	SÍ	SÍ	SÍ
Manejo de interfaz web	SÍ	NO	SÍ	NO	NO	SÍ	SÍ

Tabla 1: Comparación de controladores Open-Source [Fuente: Propia]

1.3 Controladores propietarios

Estos son controladores diseñados por empresas para ser usados principalmente con switches OpenFlow de la misma marca, pero también pueden ser usados con dispositivos de otros fabricantes, suelen poseer algunas características que los controladores Open-source no poseen y son más amigables con el usuario, teniendo un interfaz de usuario más intuitivo.

Ilustración 12: Controlador HP VAN

HP VAN SDN[11]: El controlador de Hewlett Packard, Virtual Applications Networks (VAN), provee un punto unificado de control para una red OpenFlow, simplificando la gestión, orquestación y la provisión de la red. Permite el desarrollo servicios de red basada en aplicaciones y brinda APIs para que programadores puedan innovar soluciones para cualquier red en particular. Está diseñado para operar en campus, centro de datos, y redes de proveedores de servicios. Algunas características de este controlador son:

* Plataforma de clase empresarial que permite el desarrollo de un gran número de innovaciones en la red.

* Soporta desde OpenFlow versión 1 hasta la versión 1.3

* Soporta más de 50 switches físicos

* APIs abiertas para la interacción de desarrolladores externos

* Arquitectura de controlador extensible, escalable y resiliente.

Ilustración 13: Logo NEC

ProgrammableFlow PF6800[12]: Este controlador de la empresa japonesa NEC (Nippon Electric Company)está en el centro del tejido de red basado en OpenFlow. Proporciona un punto de control para las redes físicas y de gestión para las redes virtuales y físicas. El controlador se considera programable, así como estandarizado. Se integra tanto con OpenStack, como con Microsoft System Center Virtual Machine Manager para una gestión de red y orquestación añadidas. El controlador también incluye la tecnología de red de inquilino virtual de NEC, que permite redes aisladas, con varios inquilinos.

Ilustración 14: Logo Nuage

El Controlador de Servicios Virtualizados (VSC): Perteneciente a Nuage Networks, permite la visión completa de una red por inquilino y las topologías de servicio, mientras externaliza plantillas de servicios de la red definidos a través del Directorio de Servicios Virtualizados de Nuage Networks. El directorio de servicios es un motor de políticas que utiliza el análisis y las reglas de red para autorizar permisos basados en roles. El VSC envía mensajes usando esas reglas a la plataforma de conmutación y ruteo virtual de Nuage. La plataforma detecta ya sea la creación o supresión de una máquina virtual y luego le pregunta al controlador SDN si hay una política instalada para ese inquilino. Si existe una regla, la conectividad de red se establece inmediatamente.

Ilustración 15: Controlador NSX

El controlador NSX: Una creación de VMWare, se considera un sistema de gestión de estado distribuido que controla las redes virtuales y superpone los túneles de transporte. Es el punto de control central para todos los switches lógicos dentro de una red. El controlador mantiene la información de las máquinas virtuales, hosts, switches lógicos y VXLANs, mientras usa APIs en dirección norte para hablar con las aplicaciones.Cuando se trabaja con el controlador, las aplicaciones comunican lo que necesitan, y el controlador programa todos los switches virtuales bajo el control NSX en dirección sur para cumplir con esos requisitos. El controlador podría ejecutarse de dos maneras dentro de NSX: ya sea como un clúster apartado de máquinas virtuales en un entorno vSphere, o en los aparatos físicos para aquellos con hipervisores mixtos.

Como ya se vio más arriba, el controlador es el núcleo de estas redes, es la parte más importante, el encargado de la "inteligencia" de la red. Él es quien toma las decisiones como la distribución de recursos, decisiones sobre paquetes que no coinciden con las tablas de flujo de los switches, creación, modificación, o eliminación de entradas de flujo, ejecuta las instrucciones que le proporcionan las distintas aplicaciones, básicamente, centraliza el funcionamiento de la red que es la característica básica de las SDN.

1.4 Clasificación de los controladores.

Según el tipo de red en el que se encuentre, se lo puede clasificar en distintos modos de funcionamiento:

Emplazamiento: Se separa en dos configuraciones, una es centralizada, donde un solo controlador maneja a toda la red y otra en donde hay un controlador para un conjunto de dispositivos de una red.

Por flujo: Donde cada entrada define un flujo en particular o donde una entrada define un conjunto de flujos.

Comportamiento: Puede ser reactivo, donde el controlador va a agregar una nueva entrada de flujo cuando llegue un paquete al switch que no tenga

coincidencia con ninguna entrada, o proactivo, donde el controlador completa la tabla de flujo antes de que lleguen flujos nuevos. Cuando es reactivo, cada paquete nuevo que llegue al switch va a tener un tiempo de retardo que se corresponde a su procesamiento en el controlador y a la implementación de una nueva entrada de flujo, y si se pierde la conexión entre ambos se corta, el switch no podrá conmutar paquetes de flujos nuevos, limitando su conectividad. En el modo proactivo, no hay retardo y no se pierde conectividad si se corta la comunicación entre el controlador y el switch.

Un controlador puede ser, desde un simple software en una computadora, encargado de gestionar las tablas de flujos de los dispositivos de una red, quedando en mano de un solo administrador encargado de la gestión, hasta múltiples administradores de red que pueden gestionarla por medio de distintas cuentas y con la posibilidad de poder configurar distintos conjuntos de flujos de la red.

2. Protocolo Openflow

OpenFlow es una tecnología de switching que surgió a raíz del proyecto de Investigación: "OpenFlow: Enabling Innovation in Campus Networks" de 2008 en la Universidad de Stanford. Se define como un protocolo emergente y abierto de comunicaciones que permite a un servidor de software determinar el camino de reenvío de paquetes que debería seguir en una red de switches. Con el protocolo OpenFlow, una red puede ser gestionada como un todo, no como un número de dispositivos que gestionar individualmente, es el propio servidor el que dice a los switches dónde deben enviar los paquetes. Con esta tecnología, las decisiones que impliquen el movimiento de paquetes están centralizadas, por lo que la red puede ser programada independientemente de los switches. En un switch convencional, el reenvío de paquetes Datapath y el encaminamiento de alto nivel (control path) se realizan en el mismo dispositivo,

sin embargo en los switches OpenFlow ambos se separan. Con OpenFlow, una parte del datapath reside en el mismo switch, pero es un controlador el que realiza las decisiones de encaminamiento de alto nivel. Ambos elementos se comunican por medio del protocolo OpenFlow. Esta metodología, conocida como SDN permite una mayor efectividad en el uso de los recursos de la red que en una red convencional. OpenFlow está pensada para afrontar la movilidad de máquinas virtuales (VM), redes con misiones críticas o redes NGN móviles

La idea básica es simple: explotar el hecho de que la mayoría de los switches Ethernet contienen tablas de flujos (Flow-Tables), que trabajan a la velocidad de la línea para implementar firewalls, NAT, QoS y recolectar estadísticas. Aunque las Tablas de Flujos que maneja cada fabricante son propias, se han aprovechado características observadas y comunes a todas ellas. Precisamente eso es lo que ofrece OpenFlow, un protocolo abierto para poder programar las tablas de flujo en diferentes switches y routers. Los administradores de la red solo necesitarían dividir el tráfico entre producción y el dedicado a la investigación. De esta manera, se conseguiría experimentar con nuevos protocolos, nuevos modelos de seguridad, esquemas de direccionamiento, incluso alternativas para IP y en definitiva una innovación mayor. Visto de esta manera, OpenFlow podría ser una generalización de las VLAN´s. El tráfico de producción no se vería afectado ya que está aislado y se procesaría de la misma manera que se ha estado realizando hasta hoy día. Las acciones que pueden soportar los switches OpenFlow son extensibles, si bien es necesario tener unas características mínimas y comunes a todos ellos.

2.1 Partes de un Switch OpenFlow

Un switch OpenFlow consiste en al menos tres partes:

- Tabla de flujos: con una acción asociada a cada entrada de la tabla, indicando al switch cómo debe procesar ese flujo

- Canal seguro que conecte el switch a un proceso de control remoto (controlador), permitiendo comandos y paquetes se puedan enviar entre el switch y el controlador usando el protocolo OpenFlow
- Controlador: Un controlador añade y elimina entradas en la tabla de flujos para poder permitir los experimentos.

Un controlador estático podría ser una simple aplicación funcionando en un PC que añadirá estáticamente flujos y que puede interconectarse a su vez a una serie de ordenadores con funciones de analizadores de tráfico. Existen también controladores más sofisticados que dinámicamente añaden/eliminan flujos. De esta manera, un Administrador de Red será capaz de gestionar todos los flujos y cómo serán procesados. Un controlador de estas características podría ser soportado por varios investigadores, cada uno con diferentes cuentas y permisos, permitiendo que todos ellos puedan realizar múltiples experimentos con diferentes conjuntos de flujos. El protocolo del controlador del switch trabaja sobre la capa TLS (Transport Layer Security) o conexiones TCP sin seguridad.

2.2 Tipos de Switches

Existen dos tipos de switches: aquellos switches OpenFlow-only (solo OpenFlow) que no son compatibles con las capas 2 y 3 convencionales y los híbridos.

Switches OpenFlow-only Un switch que soporta OpenFlow, contiene múltiples tablas de flujos, y cada flujo contiene múltiples entradas de flujo. El procesado define cómo los paquetes interactuarán con estas tablas de flujos. Requiere que tenga al menos una tabla de flujo y opcionalmente más. Cuantas menos entradas más simplificado será el procesado.

En este contexto los flujos están ampliamente definidos, solo están limitados por la capacidades de una determinada implementación de la Tabla de Flujo. Por ejemplo un flujo puede ser una conexión TCP, o todos los

paquetes de una determinada dirección MAC o IP, todos los paquetes con la misma etiqueta VLAN, o todos los paquetes de un mismo puerto del switch. Para experimentos que no incluyen paquetes IPv4 , un flujo podría ser definido para que todos los paquetes cumplan una cabecera específica (no estándar). En la sección ejemplos se explicará más ampliamente este concepto.

Switches OpenFlow-híbridos Switches que soportan tanto la operación OpenFlow como el Switching Ethernet convencional. Operaciones habituales pueden ser: Switching Ethernet de Capa 2, aislamiento de tráfico con VLAN, nivel 3 o de routing (IPV4, IPV6), listas de acceso o QoS. Estos switches deberán proveer un mecanismo de clasificación fuera de OpenFlow que enrute el tráfico o bien ser procesado por OpenFlow. Por ejemplo, un switch deberá usar una etiqueta VLAN o puerto de entrada para decidir si se procesa el paquete de una manera u otra o debe redirigirlos hacia el procesador OpenFlow. Este mecanismo de clasificación sale fuera del ámbito de la especificación. El protocolo OpenFlow permite que el switch sea controlado por dos o más controladores para incrementar el rendimiento y la robustez del sistema. A continuación se muestra una posible configuración de este tipo de switches, en el que todas las tablas de flujos se gestionan por el mismo controlador.

2.3 Como funciona OpenFlow

Los switches tradicionales usan STP, SPB, TRILL para determinar cómo se reenvían los paquetes. OpenFlow traslada esta decisión de reenvío de los switches a los controladores, típicamente un servidor o una estación de trabajo. Una aplicación de gestión se ejecutará en las interfaces del controlador que une todos los switches en la red, facilitando la configuración de caminos de reenvío que utilizarán todo el ancho de banda disponible. La especificación define el protocolo entre el controlador y los switches y un conjunto de operaciones que se pueden realizar entre ellos. Las instrucciones de reenvío se basan en el flujo, que consiste en que todos los paquetes comparten una serie

de características comunes. Existen infinidad de parámetros que pueden especificarse para definir un flujo. Entre los posibles criterios podemos incluir los puertos por donde se reciben los paquetes cuando llegan, el puerto Ethernet de origen, la etiqueta VLAN, el destino Ethernet o el puerto IP y otro número de características de los paquetes. Un nuevo flujo se debe crear cuando un paquete que llega no encuentra ninguna coincidencia con ninguna entrada de la tabla. El switch debería tener configurado un descartado de paquetes para el flujo que no haya sido definido, pero en la mayoría de los casos, el paquete será enviado al controlador. El controlador entonces define un nuevo flujo para ese paquete y crea una o más entradas para la tabla. Éste envía la entrada o entradas al switch para que sean añadidas a las tablas de flujo. Finalmente, el paquete se envía de vuelta al switch para ser procesado con las nuevas entradas creadas.

2.4 Flujos

Cada flujo de entrada tiene asociada una acción simple. Las tres básicas son:

- Reenvío de flujo de paquetes a un puerto dado (o puertos). Esto permite que los paquetes ser encaminados a través de la red. En la mayoría de los switches sucede a la velocidad de la línea.
- Encapsulación y reenvío este flujo de paquetes al controlador. El paquete será enviado al Canal Seguro, donde se encapsula y se envía al controlador. Típicamente se usa para el primer paquete en un nuevo flujo, para que el controlador pueda decidir si el flujo debe ser añadido a la tabla de flujos. También se puede usar para reenviar todos los paquetes al controlador para que sean procesados.
- Descartar este flujo de paquetes. Puede ser usado por seguridad , para parar ataques de denegación de servicio o reducir el falso tráfico de descubrimiento broadcast desde los hosts finales.

2.5 Tablas de Flujos

Una entrada de flujo tendrá este formato:

Se identifica principalmente por sus coincidencias y prioridad. Ambos campos identifican un flujo único en la Tabla de Flujos. Podemos citar 3 campos principales:

- La cabecera del paquete que define el flujo.
- La acción, que define como deben ser procesados los paquetes.
- Las estadísticas que llevan la monitorización del número de paquetes y los bytes de cada flujo y el tiempo desde el cual el último paquete coincidió el flujo (esto permite el borrado automático de flujos inactivos).

- ✓ Match fields: puerto y cabecera. Opcionalmente metadatos especificados en una tabla anterior.
- ✓ Priority: coincidencia con el flujo de entrada.
- ✓ Counters: Se actualiza cuando se encuentra una coincidencia.
- ✓ Instructions: para modificar el conjunto de acciones.
- ✓ Timeouts: tiempo máximo antes de que el switch descarte el flujo.
- ✓ Cookie: Valor que elige el controlador para filtrar las estadísticas, las modificaciones y el borrado de los flujos. No se usa: cuando se procesan los paquetes.

2.6 Matching

Cuando se recibe un paquete el Switch OpenFlow realiza las acciones que se ven en la figura. El switch comienza haciendo una búsqueda en la primera tabla de flujo. Las tablas de flujos se numeran empezando por el cero, y basado en esta primera búsqueda realizara búsquedas en otras tablas de flujo.

A los campos que coinciden con alguna entrada se les extrae del paquete. Los campos que se utilizan para buscar coincidencias dependen del tipo del paquete y típicamente incluyen varios campos de cabecera, las coincidencias también se pueden hacer en base al puerto de entrada o por campos de metadatos.

Los metadatos se usarán para poder mandar información entre las tablas en un switch. Un paquete coincide con una entrada de la tabla de flujos si los valores de los campos del paquete que se usan para la búsqueda están definidos en la misma.

Si tiene un valor ANY (omitido), coincidirá con todos los valores posibles en la cabecera. Si el switch trabaja con mascaras arbitrarias para campos específicos se podrá afinar mucho más las coincidencias. El paquete solo coincidirá también tiene la prioridad más alta Los contadores asociados a la tabla seleccionada deben ser actualizados y el set de instrucciones incluido en el flujo seleccionado, será aplicado.

Si hay múltiples coincidencias y todas tienen configuradas la misma prioridad, la entrada de flujo seleccionada esta explícitamente indefinida. Está especificación no tiene en cuenta si el switch recibe un paquete corrupto o con un formato que no es el adecuado.

2.7 Tablas de Grupos

Los grupos proveen una eficiente manera de indicar que el mismo conjunto de acciones deben ser llevadas a cabo por múltiples flujos. Por ello, es útil para implementar multicast y unicast.

Cada entrada consiste en su identificador, el tipo de grupo, un contador y un set de action buckets.

- Identificador de grupo: entero sin signo que define unívocamente el grupo.
- Tipo de Grupo: para determinar la semántica del grupo.

- Contadores: Actualizados cuando los paquetes son procesados por el grupo.
- Set de acciones (action buckets): lista ordenada de acciones, donde cada acción contiene un conjunto de acciones a ejecutar y unos parámetros asociados.

2.8 Meter Tables

Con estas tablas, se mide la tasa de paquetes asignadas a ellas y se facilita el control de la tasa de esos paquetes. Están unidas directamente a las entradas de flujo (al contrario que las colas que están relacionadas con los puertos). Consiste en entradas de medidas, definiendo medidas por flujo. Habilitan a OpenFlow para poder implementar operaciones de QoS simples, como limitar el tráfico, y pueden ser combinadas con colas por puerto de entrada, para implementar configuraciones de QoS complejas como DiffServ.

Una tabla tiene los siguientes campos:

- Identificador de la medida: entero sin signo de 32 bits que lo define unívocamente.
- Meter bands: lista ordenada de ellas, donde cada una especifica el valor de la banda y la manera de procesar el paquete.
- Contadores: Se actualizan cuando el paquete es procesado por alguna de ellas.

2.9 Instrucciones

Cada tabla de flujo contiene un set de instrucciones que se ejecutan cuando el paquete encuentra una coincidencia con la entrada. Estas

instrucciones dan lugar a cambios en el paquete, en el conjunto de acciones y en el procesamiento.

Un switch no tiene por qué soportar todos los tipos de instrucciones, solo las que se marcan como requeridas y son: Instrucción de escritura (Write-Actions "action"): Si la acción del tipo dado existe, se sobrescribe, si no, se añade. Ir a la tabla (Goto-Table "next-table-id"): Indica la siguiente tabla en el procesado. El identificador de la siguiente tabla debe ser mayor que el de la actual. Las entradas del flujo de la última tabla no pueden incluir esta instrucción. Solo se permite una instrucción por cada tipo. Los switches OpenFlow con una tabla no lo necesitarán.

Las instrucciones de la tabla de flujos modifican la acción asociada a cada paquete. Los paquetes empiezan a procesarse con un conjunto de acciones vacío. Las acciones pueden especificar que el paquete que se va a reenviar a través de un puerto específico, modificar el TTL, VLAN, etiqueta MPLS o la QoS.

Además, una instrucción puede especificar un identificador de grupo. Se pueden definir en el switch mediante entradas en la tabla de grupo Las instrucciones de la primera tabla deberán realizar una acción en el paquete o añadir acciones que se realizarán después. Las instrucciones asimismo, deberán permitir que el procesado de paquetes continúe comparándolos con las entradas de otras tablas de flujos.

3. Desarrollo

Este trabajo surgió de la necesidad de determinar las capacidades del Switch HP 2920-24G disponible en el aula de becarios de la facultad de ingeniería trabajando con el protocolo OpenFlow y asentar resultados para futuros trabajos o investigaciones.

3.1 Tecnologías utilizadas.

Hosts: Para la creación de la red se utilizaran dos computadoras como hosts. La primera presenta estas características importantes:

Marca: Bangho

Procesador: Intel® Core™ I3-2330M CPU @ 2.20 GHz 2.20 GHz

Memoria RAM: 6 GB

Puertos Ethernet: 1 puerto de 10/100/1000 Mbps

Sistema Operativo: Windows 7 Profesional 64bits

El segundo host posee las siguientes características importantes:

Marca: Lenovo

Procesador: Intel® Core™ I7-2702MQ CPU @ 2.20 GHz x8

Memoria RAM: 6 GB

Puertos Ethernet: 1 puerto de 10/100/1000 Mbps

Sistema Operativo: Windows 7 Profesional 64bits

Controlador: Para este proyecto se eligió el controlador POX por su facilidad de uso, su rendimiento, la cantidad de bibliografía y documentación que posee y sus complementos adicionales que lo transforman en una buena opción para redes de pequeña escala, además de ser OpenSource.

Ilustracion 16: Controlador POX

Luego se eligió el controlador RYU, para realizar comparaciones respecto al POX.

Ilustración 17: Controlador RYU

Tanto para el controlador POX como para el RYU, se utilizo una computadora como servidor que presenta las siguientes características:

Marca: Bangho

Procesador: Intel® Core™ I7-4702MQ CPU @ 2.20 GHz x 8

Memoria RAM: 7,7 GB

Puertos Ethernet: 1 puerto de 10/100/1000 Mbps

Sistema Operativo: Ubuntu 14.04 LTS 64bits

Switch OpenFlow: El switch a utilizar será de la marca *Hewlett-Packard*, más precisamente el switch HP Aruba 2920-24G[37], que posee estas características:

- Puertos: 20 puertos RJ-45 10/100/1000 con detección automática, 4 puertos RJ-45 10/100/1000 con doble función y 2 ranuras para módulos adicionales de 10 Gbps.
- Memoria y Procesador: Tri Core ARM1176 a 625 Mhz con 512 MB de SDRAM con tamaño de buffer para paquetes de 11,25 MB.
- Latencia: 10 Mb< 9µs, 100 Mb< 3,3 µs y 1 GB< 3,3 µs.
- Version del Firmware: Software Revision WB.16.01.0007
- Version del Protocolo OpenFlow: 1.0

Ilustración 18: Switch HP 2920-24G [Fuente: www.hp.com]

El proyecto se dividió en 2 partes, primero la implementación de la red (switch y 2 host), con el controlador POX y luego RYU, y la segunda parte, la implementación con el switch en modo tradicional, es decir, sin el protocolo OpenFlow .

3.2 Desarrollo de los controladores

Instalación del pox.

Abrimos una consola en Ubuntu y luego escribimos las siguientes líneas:

```
sudo apt-get install git
git clone http://github.com/noxrepo/pox
cd pox
./pox.py openflow.of_01 --address=&lt;NUESTRAIP&gt; --port=6633
```

4. Pruebas y comparación

4.1 Introducción

El último objetivo de este proyecto es el de realizar pruebas sobre nuestra red con el protocolo OpenFlow, medir sus capacidades y compararlas con la del Switch en modo tradicional. Los dos parámetros que analizaremos

serán el throughput del switch y el delay de los paquetes. Para realizar estas mediciones utilizaremos el software "Iperf" que es Open-Source y funciona bien tanto en distribuciones de Windows como Linux.

Iperf

Es una herramienta que se utiliza para hacer pruebas en redes informáticas. Su funcionamiento consiste en crear y enviar flujo de paquetes TCP y/o UDP desde un cliente hacia un servidor y viceversa, y medir el rendimiento de la red. Fue desarrollado por el DAST (Distributed Applications Support Team) y está escrito en C++.

IPerf es un programa cliente-servidor muy sencillo que permite medir la velocidad máxima que alcanzan 2 ordenadores conectados en red local.

Esto es útil si queremos ver la velocidad máxima de nuestro switch o router y si cambiando ciertos parámetros como el MTU de la red, conseguimos más velocidad.

Este programa está para Windows y también para GNU/Linux.

Se necesitan 2 hosts, uno actuara como servidor, el cual escuchara en cierto puerto (normalmente el 5001) y se quedara a la espera de paquetes enviados por el cliente, este último es el encargado de generar paquetes con dirección al servidor. Al ser de código abierto podemos analizar su metodología de funcionamiento.

Iperf posee una versión con GUI llamada Jperf, esa será la que utilizaremos en estas pruebas.

4.2 Diagrama de prueba

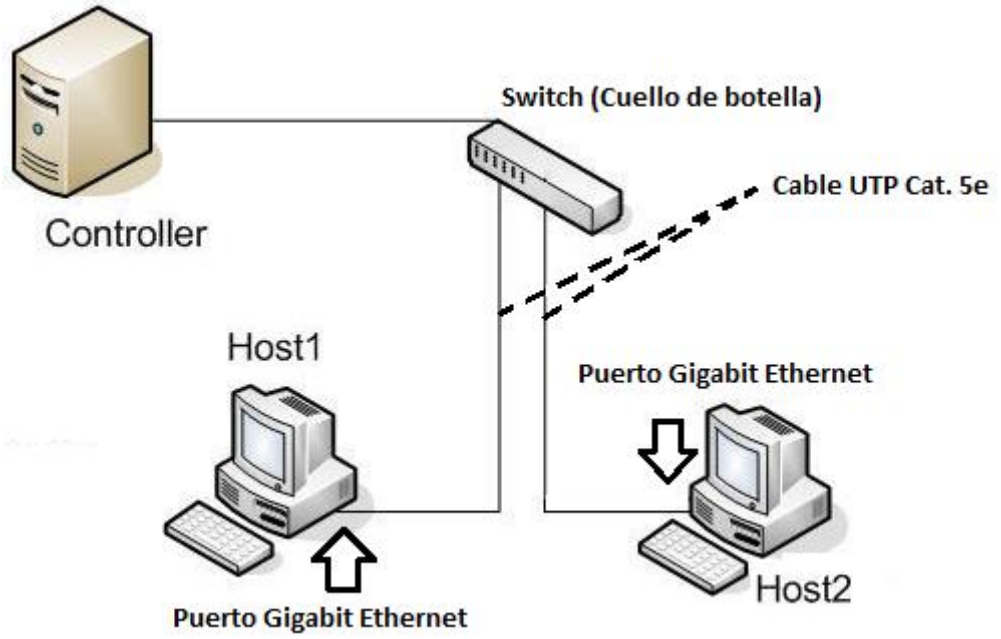


Ilustración 19: Diagrama de red

Al saber la cantidad de datos que puede manejar la red podemos asegurarnos que el único limitante va a ser la velocidad de procesamiento que posee nuestro switch.

Las pruebas consistieron en inundar la red con paquetes UDP, utilizando un host como cliente (generador de los paquetes) y otro como servidor (Receptor de los paquetes). Luego Jperf nos entregó los resultados. Se eligió UDP por encima de TCP porque no posee control de flujo, por ende el cliente mandara datos al servidor solo teniendo en cuenta el ancho de banda especificado por el usuario. Al no haber control de flujo Iperf permite a la red desechar paquetes, por lo tanto, el throughput reportado por el servidor será el número de paquetes que circularon por la red sin ser desechados y es la actual velocidad de transferencia de la red.

Se tomaron varias pruebas, donde se enviaba un flujo de paquetes UDP por 10 segundos, en las cuales se fueron cambiando parámetros para ver cómo reaccionaba el switch, tanto con el protocolo OpenFlow como en el modo tradicional.

4.3 Mediciones

1 MBits de ancho de banda

POX

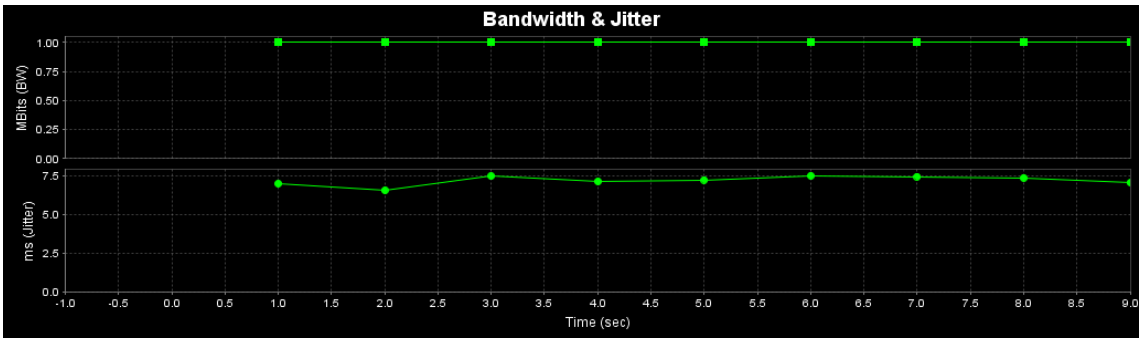


Ilustración 20: Paquetes en servidor a 1 Mbits controlador POX

```
240] local 192.168.2.71 port 56152 connected with 192.168.2.70 port 5001
 ID] Interval       Transfer     Bandwidth
240]  0.0- 1.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  1.0- 2.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  2.0- 3.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  3.0- 4.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  4.0- 5.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  5.0- 6.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  6.0- 7.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  7.0- 8.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  8.0- 9.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  9.0-10.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  0.0-10.0 sec  1.19 MBytes  1.00 Mbits/sec
240] Server Report:
240]  0.0-10.0 sec  1.19 MBytes  1.00 Mbits/sec  7.487 ms    1/  851 (0.12%)
240] Sent 851 datagrams
```

Ilustración 21: Medicion del ancho de banda y perdida de paquetes con POX

RYU

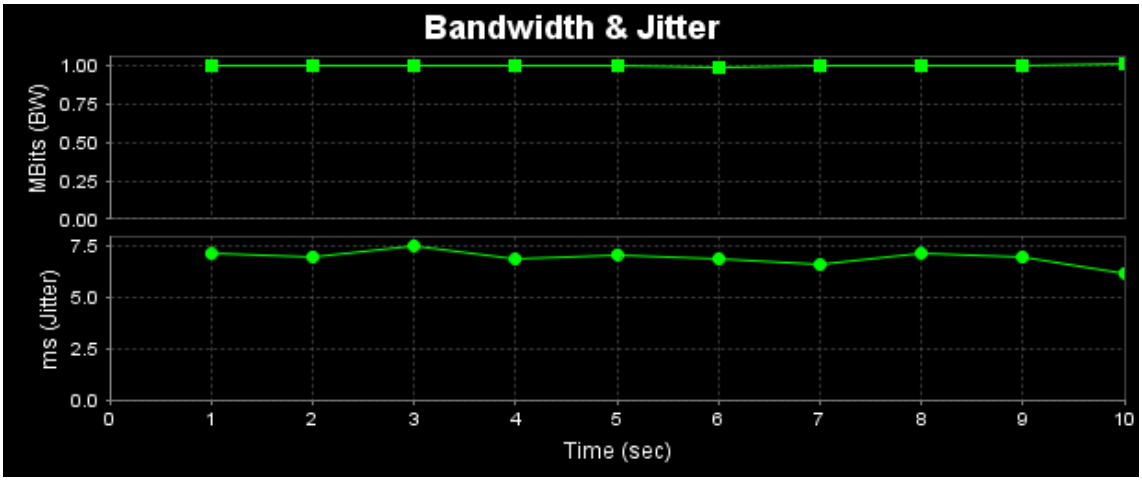


Ilustración 22: Paquetes en servidor a 1 Mbits controlador RYU

```
[244] local 192.168.2.71 port 63403 connected with 192.168.2.70 port 5001
[ ID] Interval       Transfer     Bandwidth
[244]  0.0- 1.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  1.0- 2.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  2.0- 3.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  3.0- 4.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  4.0- 5.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  5.0- 6.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  6.0- 7.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  7.0- 8.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  8.0- 9.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  9.0-10.0 sec  0.12 MBytes  1.00 Mbits/sec
[244]  0.0-10.0 sec  1.19 MBytes  1.00 Mbits/sec
[244] Server Report:
[244]  0.0-10.0 sec  1.19 MBytes  1.00 Mbits/sec  7.430 ms    0/  851 (0%)
[244] Sent 851 datagrams
Done.
```

Ilustración 23: Medición del ancho de banda y perdida de paquetes con RYU

Tradicional

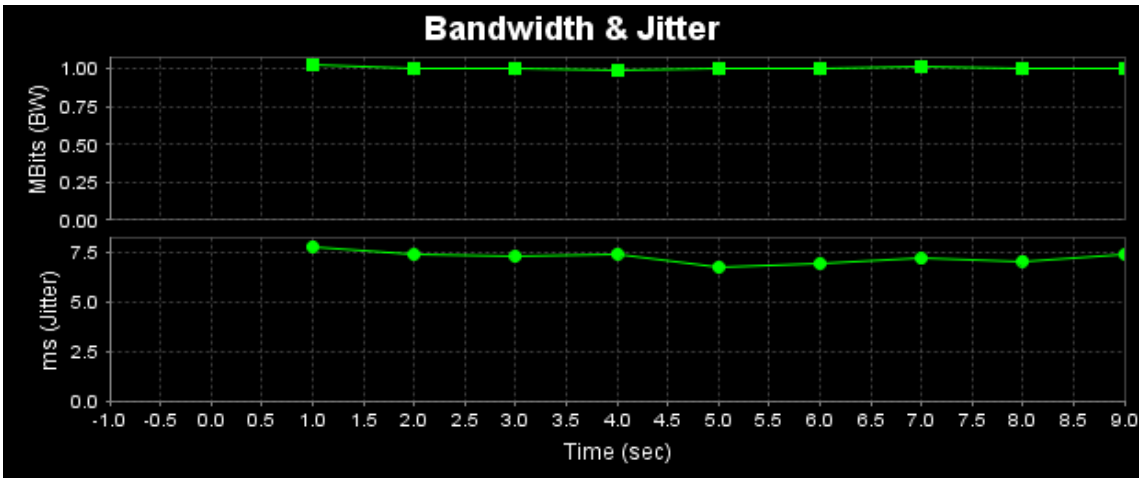


Ilustración 24: Paquetes en servidor a 1 Mbits modo tradicional

```
240] local 192.168.2.71 port 56818 connected with 192.168.2.70 port 5001
 ID] Interval       Transfer     Bandwidth
240]  0.0- 1.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  1.0- 2.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  2.0- 3.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  3.0- 4.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  4.0- 5.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  5.0- 6.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  6.0- 7.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  7.0- 8.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  8.0- 9.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  9.0-10.0 sec  0.12 MBytes  1.00 Mbits/sec
240]  0.0-10.0 sec  1.19 MBytes  1.00 Mbits/sec
240] Server Report:
240]  0.0-10.0 sec  1.19 MBytes  1.00 Mbits/sec  6.654 ms    0/  851 (0%)
240] Sent 851 datagrams
one.
```

Ilustración 25: Medición del ancho de banda y perdida de paquetes modo tradicional

50 Mbits de ancho de banda

POX

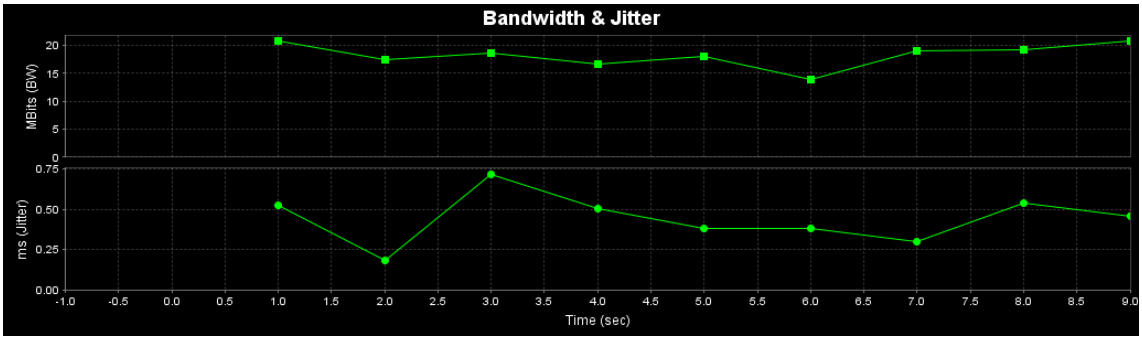
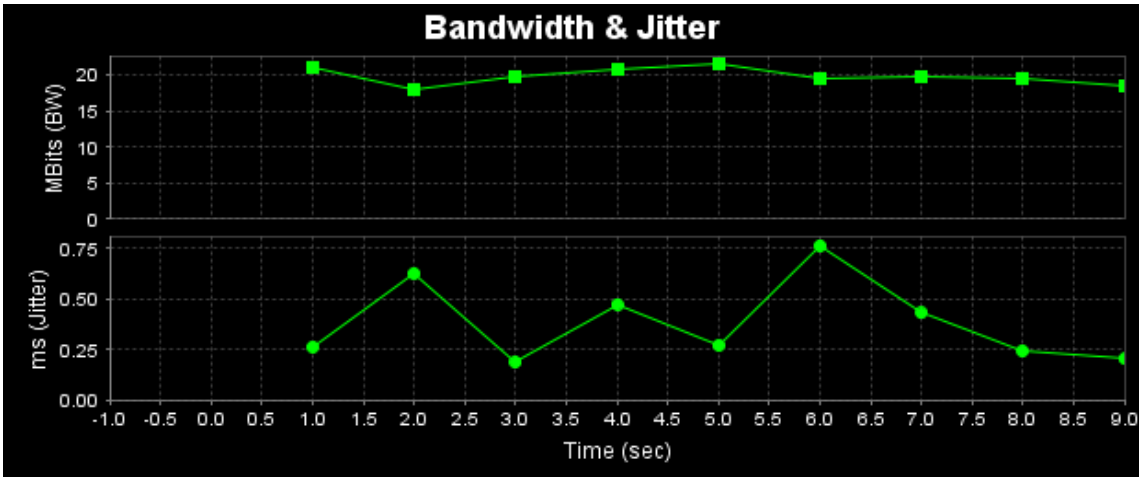


Ilustración 26: Paquetes en servidor a 50 Mbits con POX

```
[240] local 192.168.2.71 port 56168 connected with 192.168.2.70 port 5001
[ ID] Interval       Transfer     Bandwidth
[240]  0.0- 1.0 sec  5.91 MBytes  49.6 Mbits/sec
[240]  1.0- 2.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  2.0- 3.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  3.0- 4.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  4.0- 5.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  5.0- 6.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  6.0- 7.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  7.0- 8.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  8.0- 9.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  9.0-10.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  0.0-10.0 sec  59.6 MBytes  50.0 Mbits/sec
[240] Server Report:
[240]  0.0- 9.9 sec  21.5 MBytes  18.3 Mbits/sec  1.828 ms 27156/42516 (64%)
[240] Sent 42516 datagrams
Done.
```

Ilustración 27: del ancho de banda y perdida de paquetes con POX

RYU



Ilustracion 28: Paquetes en servidor a 50 Mbits con RYU

```
[240] local 192.168.2.71 port 63324 connected with 192.168.2.70 port 5001
[ ID] Interval       Transfer     Bandwidth
[240]  0.0- 1.0 sec  5.91 MBytes  49.6 Mbits/sec
[240]  1.0- 2.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  2.0- 3.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  3.0- 4.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  4.0- 5.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  5.0- 6.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  6.0- 7.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  7.0- 8.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  8.0- 9.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  9.0-10.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  0.0-10.0 sec  59.6 MBytes  50.0 Mbits/sec
[240] Server Report:
[240]  0.0- 9.7 sec  23.0 MBytes  19.9 Mbits/sec  1.083 ms 26075/42515 (61%)
[240] Sent 42515 datagrams
```

ilustración 29: Medición de ancho de banda y perdida de paquetes con RYU.

Tradicional

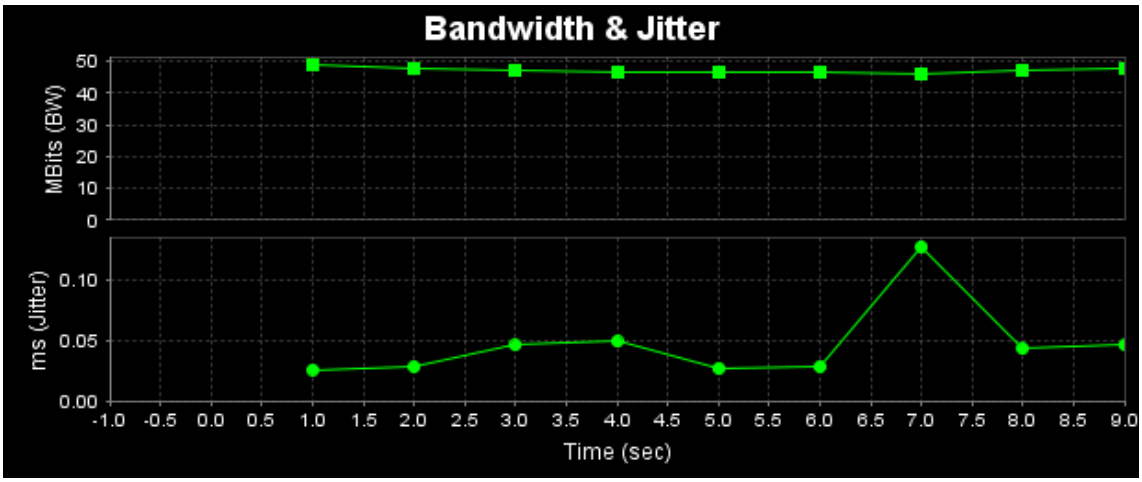


Ilustración 30: Paquetes en servidor a 50 Mbits modo tradicional

```
[240] local 192.168.2.71 port 52369 connected with 192.168.2.70 port 5001
[ ID] Interval       Transfer     Bandwidth
[240]  0.0- 1.0 sec  5.91 MBytes  49.6 Mbits/sec
[240]  1.0- 2.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  2.0- 3.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  3.0- 4.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  4.0- 5.0 sec  5.98 MBytes  50.2 Mbits/sec
[240]  5.0- 6.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  6.0- 7.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  7.0- 8.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  8.0- 9.0 sec  5.97 MBytes  50.1 Mbits/sec
[240]  9.0-10.0 sec  5.97 MBytes  50.0 Mbits/sec
[240]  0.0-10.0 sec  59.6 MBytes  50.0 Mbits/sec
[240] Server Report:
[240]  0.0-10.0 sec  56.0 MBytes  47.1 Mbits/sec  0.613 ms 2554/42526 (6%)
[240] Sent 42526 datagrams
Done.
```

ilustración 31: Medición de ancho de banda y perdida de paquetes modo tradicional

4.4 Resultados

	Pox	Ryu	Tradicional
Ancho de banda	1 Mbits/s	1 Mbits/s	1 Mbits/s
Transferido	1,19 Mbyte	1.19 Mbyte	1,19 Mbyte
Recibido	1,19 Mbyte	1.19 Mbyte	1,19 Mbyte
% Perdida paquetes	0,12 %	0 %	0 %
	Pox	Ryu	Tradicional
Ancho de banda	5 Mbits/s	5 Mbits/s	5 Mbits/s
Transferido	5.96 MBytes	5.96 MBytes	5.96 MBytes
Recibido	5.93 MBytes	5.61 MBytes	5.96 MBytes
% Perdida paquetes	0.45%	5.8%	0 %
	Pox	Ryu	Tradicional
Ancho de banda	10 Mbits/s	10 Mbits/s	10 Mbits/s
Transferido	11.9 MBytes	11.9 MBytes	11.9 MBytes
Recibido	10.4 MBytes	10.2 MBytes	11.4 MBytes
% Perdida paquetes	13 %	14%	4%
	Pox	Ryu	Tradicional
Ancho de banda	15 Mbits/s	15 Mbits/s	15 Mbits/s
Transferido	17.9 MBytes	17.9 MBytes	17.9 MBytes
Recibido	14.2 MBytes	13.9 MBytes	16.6 MBytes
% Perdida paquetes	20 %	22%	8 %
	Pox	Ryu	Tradicional
Ancho de banda	20 Mbits/s	20 Mbits/s	20 Mbits/s

Transferido	23.8 MBytes	23.8 MBytes	23.8 MBytes
Recibido	15.9 MBytes	17.1 MBytes	21.4 MBytes
% Perdida paquetes	33 %	28%	10%
	Pox	Ryu	Tradicional
Ancho de banda	30 Mbits/s	30 Mbits/s	-
Transferido	35.7 MBytes	35.7 MBytes	-
Recibido	21.2 MBytes	22.5 MBytes	-
% Perdida paquetes	41%	37%	-
	Pox	Ryu	Tradicional
Ancho de banda	50 Mbits/s	50 Mbits/s	50 Mbits/s
Transferido	59.6 MBytes	59.6 MBytes	59.6 MBytes
Recibido	21.6 MBytes	23.0 MBytes	56.0 MBytes
% Perdida	64 %	61%	6%
	Pox	Ryu	Tradicional
Ancho de banda	100 Mbits/s	100 Mbits/s	100 Mbits/s
Transferido	120 MBytes	120 MBytes	120 MBytes
Recibido	20.7 MBytes	23.1 MBytes	115 MBytes
% Perdida paquetes	83 %	81%	4.3%
	Pox	Ryu	Tradicional
Ancho de banda	200 Mbits/s	200 Mbits/s	200 Mbits/s
Transferido	241 MBytes	241 MBytes	248 Mbyte
Recibido	23.4 MBytes	23.1 MBytes	242 Mbyte
% Perdida paquetes	90 %	90%	2 %
Ancho de banda	400 Mbits/s	400 Mbits/s	400 Mbits/s
Transferido	424 Mbyte	424 Mbyte	483 MBytes
Recibido	5 Mbyte	5.5 Mbyte	426 MBytes
% Perdida paquetes	99 %	99 %	12%

Tabla 2: Total de mediciones

4.5 Gráficos de rendimiento y saturación.

Porcentaje de saturación con respecto Ancho de banda en Mbits/s

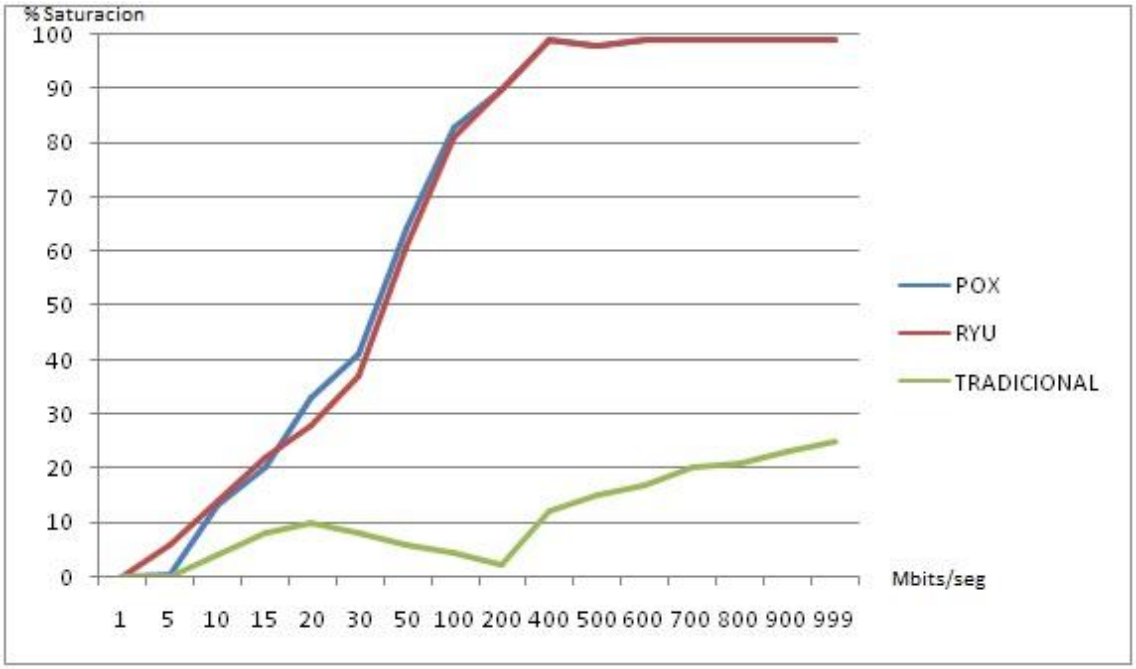


Grafico 1: Rendimiento y saturación del switch

Los resultados arrojaron valores muy bajos de ancho de banda, ese es el mayor déficit del switch HP 2920-24G operando con el protocolo OpenFlow. Esto puede tener muchos orígenes que se analizaran más adelante.

Para voz IP es necesario una mejora de estos valores, pero para redes pequeñas y solo de datos son valores esperados.

5. Conclusiones

Luego de las pruebas realizadas al switch HP se puede concluir que:

Como era de esperarse, los switches HP en modo tradicional poseen un mayor desempeño que el switch con el protocolo OpenFlow, ya sea con el controlador POX o el RYU.

El Switch en modo OpenFlow controlado por el POX , llega a su máxima capacidad a partir de los 400 Mbits/segundo. Con anchos de banda menores, el switch no se satura, pero pierde gran cantidad de paquetes. Con el controlador RYU, el switch se satura exactamente con el mismo ancho de banda, y los niveles de saturación con ancho de bandas menores, van variando, pero siempre cercanos a los valores medidos con el POX.

Las mediciones realizadas en la red diagramada en nuestro proyecto no arrojaron los resultados esperados. Encontramos saturaciones del Switch grandes con respecto a anchos de banda pequeños, en el cual uno no experimenta problemas de conexión pero sus defectos aparecen a la hora del intercambio de archivos que es muy lento.

Las saturaciones del switch en modo OpenFlow a diferencia del modo tradicional se deben principalmente a dos factores. El primero, la diferencia de performance entre los controladores de código abierto, en este caso POX y RYU, y el controlador propietario de HP, siendo sumamente superior el software propietario. En segundo lugar, el switch HP 2920-24G solo utiliza el 50 % de su performance para OpenFlow , dejando el otro 50 % para otras operaciones. Es decir no destina tu máxima capacidad de operación para OpenFlow.

Para mejorar los resultados obtenidos, para trabajos futuros, se podría modificar la performance del switch dedicado para OpenFlow (en estas mediciones designa el 50 %) [37], como así también realizar pruebas con otros controladores o con versiones mejoradas de los mismo.

A pesar de su bajo rendimiento, la red funciona bien para una exigencia mínima, que era el objetivo buscado. Se puede ver que se pierde menos del 10% de los paquetes que circulan por la red con un ancho de banda de 7,5 Mbits/seg y no hay problemas de conexión ya que se mantiene estable durante

toda la comunicación. Para una red pequeña en donde se busque una conectividad entre pocos hosts su uso puede ser satisfactorio.

Su uso para la educación y el aprendizaje sobre Redes Definidas por Software y el protocolo OpenFlow es más que aceptable, ya que podemos realizar tráfico de datos con anchos de bando menores a 10 Mbits/segundo con pérdidas de paquetes relativamente bajas.

6. Bibliografía y Anexo

[1] Open Networking Foundation, Disponible en: https://www.opennetworking.org

[2] Sdn Central, Disponible en: https://www.sdxcentral.com/

[3] NOX, Disponible en: https://github.com/noxrepo/nox

[4] POX, Disponible en: https://github.com/noxrepo/pox

[5] Project Floodlight, Disponible en: http://www.projectfloodlight.org/floodlight/

[6] Beacon, Disponible en: http://www.projectfloodlight.org/floodlight/

[7] OpenDayLight, Disponible en: https://www.opendaylight.org/

[8] Trema, Disponible en: https://trema.github.io/trema/

[9] Ryu, Disponible en: https://osrg.github.io/ryu/

[10] SDN Hub, Disponible en: http://sdnhub.org/

[11] HP VAN, Disponible en: http://h17007.www1.hp.com

[12] ProgrammableFlow PF6800, Disponible en: https://www.necam.com/sdn/doc.cfm?t=PFlowController

[13] El Controlador de Servicios Virtualizados (VSC), Disponible en: http://www.nuagenetworks.net/

[14] El controlador NSX, Disponible en: http://www.vmware.com/ar/products/nsx

[15] Open vSwitch, Disponible en: http://openvswitch.org/

[16] Iperf, Disponible en: https://iperf.fr/

[17] James F. Kurose & Keith W. Ross, Redes de computadoras, un enfoque descendente, 5° ed., Pearson, 2010.

[18] Joseph D. Sloan, Network Troubleshooting Tools, 1° ed., O'Reilly, 2001.

[19] William Stallings, Comunicaciones y redes de computadores, 7° ed., Person-Prentice Hall, 2004.

[20] Thomas D. Nadeau & Ken Gray, SDN Software Defined Network, 1° ed., O'Reilly, 2013.

[21] Óscar Roncero Hervás, Tesis "Software Defined Network", Universidad Politécnica de Catalunia, España. Disponible en: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf

[22] Wikipedia, OpenFlow. Disponible en: http://es.wikipedia.org/wiki/Openflow

[23] Wikipedia, Redes definidas por software. Disponible en: http://es.wikipedia.org/wiki/Redes_definidas_por_software

[24] Departamento de ciencias de la computación de Stanford. Disponible en: http://yuba.stanford.edu/cs244wiki/index.php/Overview.

[25] Principia Technologica. Disponible en: http://principiatechnologica.com/2014/03/25/investigacion-de-redes-sdn-parte-iii-la-arquitectura-sdn/

[26] Brian Linkletter Blog, Disponible en: http://www.brianlinkletter.com/

[27] Researchgate, Disponible en: https://www.researchgate.net/

[28] Academia GNS3, Disponible en: http://academy.gns3.com/

[29] SDN testbed, Disponible en: https://sdntestbed.wordpress.com

[30] Pakiti, Disponible en: http://pakiti.com/

[31] Redes Cisco, Disponible en: http://www.redescisco.net/

[32] Mikronauts, Disponible en: http://www.mikronauts.com/

[33] Tesis de David Andrés Serrano Carrera, "Redes Definidas por Software (SDN): OpenFlow", Universidad Politécnica de Valencia.

[34] Tesis Diana Gabriela Morillo Fueltala "Implementacion de de un prototipo de una Red Definida por Software (SDN) empleando una solución basada en software, Escuela Politécnica Nacional, Ecuador.

[35] Tesis de Gaston Borja "Diagramacion de una red definida por software de bajo costo utilizando mini ordenadores de la marca Raspberry pi".

[36] Anexo 1. Disponible en https://drive.google.com/drive/folders/0B3aoSS7lesCYZ1JKYVI2eFhpbDQ

[37] Anexo 2. Disponible en https://drive.google.com/drive/folders/0B3aoSS7lesCYRE5UVUdGb1I0WGs