ANÁLISIS COMPARATIVO DE PRESTACIONES ENTRE SDN (SOFTWARE DEFINED NETWORKING) Y REDES IP CONVENCIONALES

AUTORES

JULIÁN CAMILO SOMBREDERO ALFONSO

CÓDIGO: 2100007

ERICK FERNEY SILVA HERRERA

CÓDIGO: 2095325

ESTUDIANTES DE INGENIERÍA DE TELECOMUNICACIONES

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES

BOGOTÁ, DC.

2014

ANÁLISIS COMPARATIVO DE PRESTACIONES ENTRE SDN (SOFTWARE DEFINED NETWORKING) Y REDES IP CONVENCIONALES

AUTORES

JULIÁN CAMILO SOMBREDERO ALFONSO

CÓDIGO: 2100007

ERICK FERNEY SILVA HERRERA

CÓDIGO: 2095325

ESTUDIANTES DE INGENIERÍA DE TELECOMUNICACIONES

PROYECTO DE GRADO

DIRECTOR:

JULIO ERNESTO SUAREZ PÁEZ

INGENIERO DE TELECOMUNICACIONES

MAGISTER EN ING. DE TELECOMUNICACIONES

ASESORA:

MAYRA LILIANA SALCEDO GONZÁLEZ

INGENIERA DE TELECOMUNICACIONES

MASTER OFICIAL EN TECNOLOGÍAS, REDES Y SISTEMAS DE COMUNICACIONES

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES

BOGOTÁ, DC.

2014

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

BOGOTÁ D.C. 2014

DEDICATORIA

Este trabajo principalmente se lo dedico a mis Padres, que han sido durante todo este proceso de formación la base para que poder culminar mis metas.

Erick Ferney Silva Herrera.

Este trabajo de grado se lo dedico a mi abuelita quien en estos meses de enfermedad la he tenido presente cada día de mi vida y rezo por su pronta recuperación, por ahora le doy gracias a mi Dios por tenerla todavía a mi lado; en segundo lugar y principalmente dedicárselo a mi madre quien es mi motor de vida por la que lucho y quiero darle lo mejor, sé que sin ella no hubiera llegado a donde estoy y cada logro es gracias a ella, en tercer lugar a mi padre quien durante mi vida me ha formado como persona y estudiante, por ultimo a todos mis allegados quienes han aportado un granito de arena para mi formación, muchas gracias.

Julián Camilo Sombredero Alfonso

AGRADECIMIENTOS

Agradecer a Dios quien nos dio la oportunidad de llegar hasta aquí y continuar con nuestros estudios. Agradecimiento especial a nuestros directores, los ingenieros Julio Ernesto Suarez Páez y Mayra Liliana Salcedo González quienes creyeron en nuestras capacidades para poder ser partícipes de este proyecto y fueron nuestros guías durante todo en el desarrollo, por ultimo a nuestros profesores y amigos quienes fueron fundamentales para nuestra formación.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

RESUMEN

El presente documento muestra un análisis de prestaciones entre SDN y sistemas soportados en protocolos de enrutamiento OSPF y MPLS, este análisis se desarrolla en las siguientes fases:

La primera es la elaboración de una topología de red abarcando dos tipos de entornos: en el primero estableciendo una configuración (física) en routers Huawei (AR 1220 y AR 2220) siendo de propiedad de la Facultad de Ingeniería de Telecomunicaciones de la Universidad Santo Tomás sede Bogotá; y el segundo efectuar dicha configuración de red en routers Cisco (7200) por medio del software de simulación GNS3 implementando en ambos escenarios protocolos de enrutamiento como OSPF (Open Shortest Path First) y MPLS (Multiprotocol Label Switching) sobre direccionamiento IPv4 e IPv6, verificando su conexión y dando un diagnóstico por medio del envío de mensajes de error ICMP (Internet Control Message Protocol) con tamaños de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada uno.

La segunda consiste en implementar SDN (Software Defined Networking), para ello se realiza una configuración con características similares a las anteriormente descritas, se emplea la misma topología de red pero usando switches OpenFlow permitiendo el direccionamiento IPv4 e Ipv6 a través de un entorno de prueba de emulación.

Finalmente la tercera y última etapa consiste en el análisis de resultados, para ello se examina el comportamiento de las variaciones de tiempo de jitter y delay de todos escenarios (físico, simulado y emulado) con el fin de realizar una serie de comparaciones con respecto a protocolo, direccionamiento, tamaño de paquetes en otros.

Por último se obtuvo como resultado que SDN tiene un Delay y Jitter menor que los protocolos OSPF y MPLS trabajado en equipos físicos Huawei y simulado con CISCO GNs3.

Palabras Claves: IPv4, IPv6, OSPF, MPLS, SDN, OpenFlow.

INTRODUCCIÓN

A través de los años las redes de telecomunicaciones han ido evolucionando, iniciando con la RTPC (Red telefónica Pública Conmutada) hasta las Redes IP (Internet Protocol) que actualmente están implementadas, en este constante crecimiento, en el 2008 la universidad de Stanford desarrolla una primera versión de SDN (Software Defined Networking) (OFN, 2014), tecnología que propone una transformación a las redes basadas en IP.

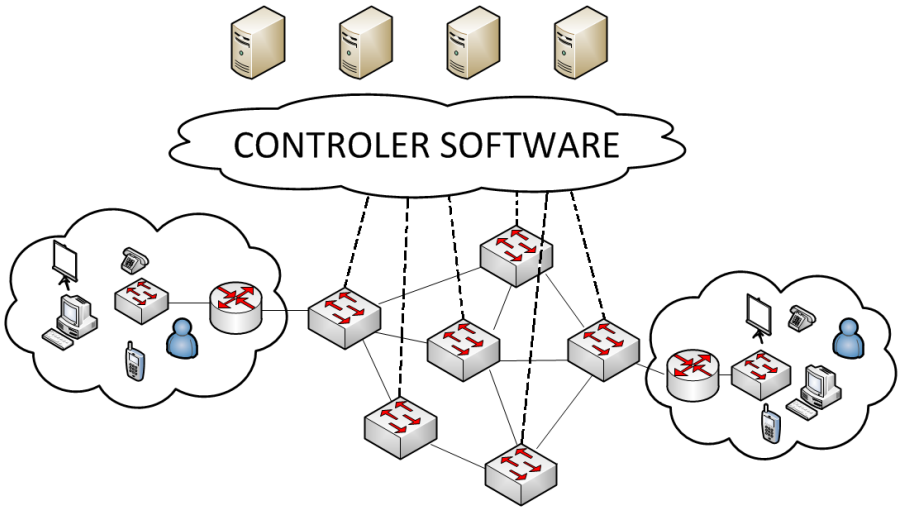
Por tal motivo, este proyecto tiene como propósito emular una red definida por Software con el protocolo OpenFlow (Figura 1) y comparar sus prestaciones con una red IP convencional, bajo los protocolos OSPF (Open Shortest Path First) y MPLS (MultiProtocol Label Switching), realizando pruebas con routers HUAWEI reales y routers CISCO mediante el simulador GNs3, por otro lado en SDN se realizaran pruebas mediante una máquina virtual con sistema Ubuntu Linux 12.04. Para cada una de estas redes se diseñan una serie de experimentos que permitan identificar diferencias con respecto al Delay y el Jitter entre cada uno de los escenarios anteriormente nombrados, por ultimo cada una de las pruebas se desarrollaran mediante IPv4 e IPv6.

En el presente documento se mostrará el desarrollo del trabajo de grado el cual está dividido de la siguiente manera: definición del problema, justificación, objetivo general, objetivos específicos, marco teórico, desarrollo del proyecto e implementación, resultados de experimentos, análisis de resultados, recomendaciones y conclusiones basadas en los protocolos OSPF y MPLS con respectó a Software Defined Networking con protocolos IPv4 e IPv6, obteniendo como resultado que el delay y el jitter manejado en SDN es más bajo que el los demás protocolos estudiados en este proyecto.

Además, destacar que en el trabajo realizado se evidenciará el beneficio que presta esta investigación, ya que la misión institucional de la Universidad Santo Tomás promueve la proyección social y el aspecto humanístico del pensamiento

de Santo Tomás de Aquino, respondiendo a los objetivos y necesidades humanísticas que presenta la sociedad actual.

Figura 1. Modelo Básico Arquitectura de SDN (Software Define Networking).



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

1. DEFINICIÓN DEL PROBLEMA

Las redes IP convencionales conectan lugares a grandes distancias y satisfacen las necesidades de conectividad de sus usuarios. Para realizar cada una de estas operaciones cada paquete tiene que pasar por routers, switches, firewalls y demás elementos los cuales toman decisiones individuales de enrutamiento que hacen difícil la gestión centralizada de la red.

La causa que origina la necesidad de este trabajo, es la tendencia de los principales fabricantes como, CISCO (Cisco, 2013), HUAWEI (Huawei, 2013), (Huawei, Carrier Network), (Huawei, 2014), HP (HP, 2014), entre otros indican que SDN será una buena opción para implementar en las redes de telecomunicaciones a futuro, ya que ofrece ventajas sobre las redes IP convencionales como la abstracción completa de la nube, conexiones globales inteligentes y envió de paquetes con poca perdida (Network Computing, 2014).

Este proyecto está orientado a comparar la tecnología SDN (la cual toma decisiones de enrutamiento centralizadas en el controlador), con una red convencional basada en IP, para determinar sus principales diferencias con respecto al jitter y delay. Por lo cual se plantea la siguiente pregunta problema a resolver:

¿SDN tiene un mejor desempeño (Jitter y Delay) con respecto a arquitecturas de red IP tradicionales?

Si no se resuelve la pregunta problema, no se podrán identificar las diferencias analizadas en este proyecto entre redes IP soportadas en protocolos OSPF y MPLS con respecto a SDN, lo que posiblemente impedirá contar con herramientas de decisión para implementaciones de SDN en el futuro.

Otra forma de solucionar la pregunta problema es realizar el estudio propuesto con los equipos reales SDN, los cuales pueden ser tarjetas NET-FPGA, sin embargo en el desarrollo de este proyecto no se contó con suficientes recursos económicos

para poder implementar esta solución, por tanto se tomó la decisión de simular la red SDN con plataformas de software libre como Linux Ubuntu.

2. JUSTIFICACIÓN

La evolución de las telecomunicaciones lleva a la necesidad de estar en constante relación con las demás personas, no solo vía telefónica sino con las diferentes aplicaciones creadas para dicha labor, de otro modo, esta necesidad conduce a un camino más amplio y de más exigencia para los ingenieros del mañana. Hoy en día las redes evolucionan acorde el pasar de los años y estas tienen como propósito la transmisión de paquetes siendo capaces de proveer servicios que integren diferentes elementos funcionales o aplicaciones para los usuarios, este proceso de cambio y evolución de la red está dado por algunas empresas que se ven en la obligación de tener más velocidad e información para sus usuarios y una mejoría en la gestión de red. Para ello se ha desarrollado SDN (OFN, 2014), el cual abarca el control de la red que es programable y desacopla funciones de enrutamiento, haciéndola ágil en el control, lo cual permitirá ajustar el tráfico del sistema (OFN, 2014). Por otra parte, SDN permite una gestión centralizada lo cual proporciona ventajas en la operación y el mantenimiento, por tal motivo esta investigación está encaminada a comparar SDN con las redes IP convencionales, proporcionando herramientas que preparen al sector de telecomunicaciones al cambio en las redes que posiblemente se dará en un futuro, no muy lejano.

2.1 FACTIBILIDAD ECONÓMICA

Para la realización de este proyecto existe la posibilidad de hacerlo con las tarjetas NET-FPGA sin embargo no se dispone de los recursos económicos para adquirir estas tarjetas, por tal razón se plantea hacerlo mediante simulación, lo que no requerirá grandes inversiones económicas.

Por otro lado las empresas como big Switch Networks (Big Switch Networks, 2014) y HP (HP, 2014) ya están implementando esta tecnología en los switches de última generación y posiblemente el valor aumentará con respecto a los anteriores.

2.2 FACTIBILIDAD TECNOLÓGICA

SDN mejorara la gestión de la redes lo que probablemente baje los costos de aplicación de esta tecnología ya que según los estudios realizados por HUAWEI, CISCO y HP, Software Defined Networking será una buena opción para implementar en redes IP dado que cuenta con unas ventajas que darán un cambio a una arquitectura de tres niveles ayudando a la agilidad de la red, eliminando la complejidad y los estándares basados en el envío de datos (Big Switch Networks, 2014).

2.3 FACTIBILIDAD OPERATIVA

En un caso futuro donde se implemente SDN, su operación puede ser más sencilla que una red IP convencional, dado que la gestión de la misma se centralizara en el controlador y en las aplicaciones de la red, por tanto SDN propone una arquitectura de red que se opere más eficientemente que una red convencional.

2.4 FACTIBILIDAD REGULATORIA

Dado que SDN es una arquitectura de red basada en IP enfocada en el transporte de datos y fundamentada en la Regulación Colombiana, según la ley 1341 de 2009 (Minsterio de Tecnologias de la Información y las Comunicaciones , 2009) (la cual tiene como objeto regular los servicios de telecomunicaciones). La posible implementación de SDN quedaría comprendida dentro del mismo marco regulatorio, dado que cumpliría las mismas funciones que las redes IP convencionales, por tanto su implantación en Colombia es viable mientras cuente con las mismas características de lo ya mencionado.

2.5 ASPECTOS HUMANÍSTICOS

Al implementar SDN es importante destacar que la Universidad Santo Tomás tenga un crecimiento académico y tecnológico, ya que al desarrollar diferentes plataformas de telecomunicaciones que hoy en día son trabajadas por operadores y empresas multinacionales da un incentivo en el campo laboral y apoya la

formación de cada uno de los estudiantes, además se evidencia el arduo trabajo y beneficio social que promueve la investigación con el objetivo de dar soluciones a las necesidades y obligaciones del ser humano que establece Santo Tomás de Aquino.

2.6 VALOR ACADÉMICO

Este proyecto trasciende en un nuevo campo de investigación en la Universidad Santo Tomás, el cual está siendo trabajado por grandes actores mundiales como: CISCO, HUAWEI y HP. Por tanto, el valor académico en las líneas de investigación del grupo INVTEL, conllevan a que a partir del mes de junio del 2014 se inicien estudios sobre esta temática, así mismo el semillero de investigación de redes de nueva generación, propone este trabajo de grado para iniciar una investigación en esta nueva tecnología de punta, debido a que este trabajo de grado tiene buenos componentes de investigación formativa con respecto al diseño y ejecución de experimentos, los cuales se realizaran con equipos reales, simuladores de red conocidos y nuevos simuladores para SDN. El proyecto cuenta con suficiente complejidad para ser realizado por dos estudiantes, porque para la realización de lo anteriormente mencionado, el trabajo para un solo estudiante tendría una complejidad demasiado alta.

Por otro lado el desarrollo de la medición (jitter y delay) estará enfocado a un desempeño pasivo, estos solo capturan el tráfico que pasa a través de una interfaz de red (Velazquéz E, 2009), por último, según el análisis en el marco teórico el jitter y delay son variables válidas para medir el desempeño en las redes y pueden ser usadas para realizar la comparación objetivo de este proyecto.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Analizar el desempeño de SDN (Software Define Networking) con el protocolo OpenFlow, con respecto a redes IP convencionales, que usan los protocolos OSPF y MPLS usando el jitter y delay como variables de comparación.

3.2 OBJETIVOS ESPECÍFICOS

- Identificar herramientas de simulación SDN e IP convencional idóneas para la realización del proyecto.
- Diseñar experimentos que permitan analizar el desempeño de SDN con respecto a IP convencional.
- Realizar pruebas mediante la plataforma anteriormente identificada, que usen protocolos SDN e IP convencional para obtener datos del desempeño de estas redes.
- Analizar los resultados generados por las pruebas realizadas anteriormente para Identificar las diferencias que se presentan en el jitter y delay.

4. MARCO TEÓRICO

4.1 REDES BASADAS EN EL PROTOCOLO INTERNET (IP)

Durante el avance de los modelos de comunicación de una red se han logrado instaurar diversas maneras de clasificación, entre las cuales se tienen: de acuerdo a su topología (bus, estrella, anillo, malla), su extensión geográfica (LAN, MAN, WAN) o por los medios que se emplean para su transmisión de datos (líneas dedicadas, circuitos conmutados que sean soportados por medio de fibra óptica, cableado Ethernet o de forma inalámbrica entre otros, es decir existen múltiples formas que obligan a surgir numerosas redes que se establecen de acuerdo a las necesidades, por lo cual por más de 20 años se ha buscado implantar un lenguaje que converge con todas éstas (redes) logrando unificarlas bajo un solo sistema de comunicación basado en IP (Unión Internacional de Telecomunicaciones (UIT), 2005).

4.2 IP (INTERNET PROTOCOL)

El protocolo de Internet, tiene como propósito la interconexión de sistemas informáticos a través de paquetes. Dicho protocolo establece los elementos y recursos que son adecuados para la transmisión de paquetes de datos denominados datagramas desde un origen a destino, siendo éstos últimos dos reconocidos por una dirección fija cada uno. De ser necesario el protocolo de internet permite realizar una segmentación y posteriormente reintegración de grandes datagramas con el fin de ser transmitidas por medio de una trama pequeña (Unión Internacional de Telecomunicaciones (UIT), 2005).

Las redes IP se apoyan en un proceso llamado enrutamiento el cual consiste básicamente en que los equipos activos de la red tomen decisiones para encontrar la ruta a través de las diferentes redes interconectadas. Los dispositivos de red que emplean el protocolo de internet deben tener la capacidad de recibir los datagramas y poder interpretar los campos que se encuentran presentes en la cabecera IP (Unión Internacional de Telecomunicaciones (UIT), 2005).

Ahora bien cuando algún datagrama debe pasar de una red a otra y esta no soporta su tamaño, dicho protocolo realiza su fragmentación. Se debe recordar que IP trata cada uno de los datagramas como una entidad independiente. Además este utiliza mecanismos como lo son el tipo de servicio, tiempo de vida y checksum. El tiempo de servicio muestra la calidad de servicio requerido, el tiempo de vida es un remitente del datagrama y se relaciona con los puntos donde se procesa, por último el checksum detecta errores en los bits de verificación del encabezado a fin de descartar en caso de error (Unión Internacional de Telecomunicaciones (UIT), 2005), (Pérez Herrera, 2003).

IP no promueve un sistema de comunicación confiable, debido a que no utiliza acuses de recibo, ni cuenta con corrección de error en los datos, no se presentan nuevas transmisiones y no consta de un control de flujo (Unión Internacional de Telecomunicaciones (UIT), 2005).

4.2.1 IPv4 (Internet Protocol versión 4)

IPv4 es un protocolo de capa de red que se fundamente a la no conexión (definido en el RFC 791). En el momento de presentarse problemas lo ideal es que el nodo implicado deseche el paquete. Como un paquete debe circular por diversos nodos, donde es posible transitar por una ruta que no obligatoriamente fue la misma emplea anteriormente, los datos pueden arribar desorganizados (Cisco), (Rey, 1981).

IPv4 utiliza direcciones de 32 bits compuestas de cuatro octetos, realizando las direcciones de la siguiente manera $2^{32} = 4.294.967.296$ dichas direcciones son únicas. Dentro de este número de direcciones encontramos 5 grupos, estos están clasificados de la siguiente forma:

Tabla 1. Direcciones IPv4

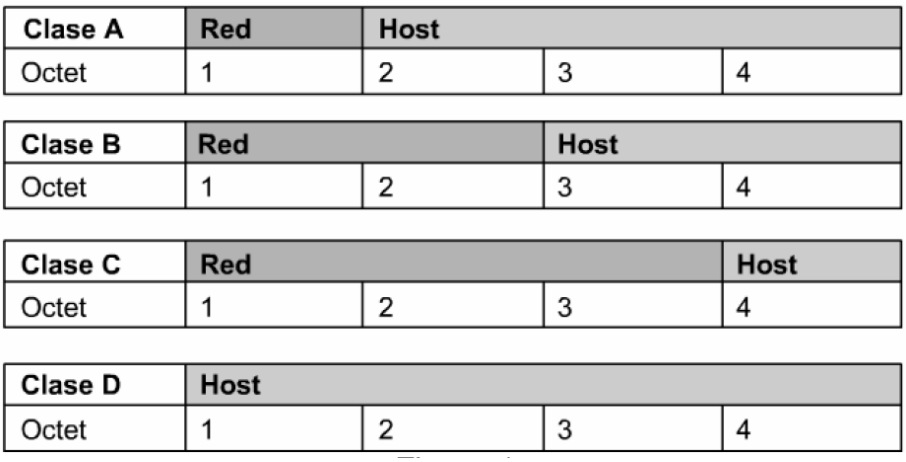
CLASE	RANGO	MÁSCARA DE RED
A	1.0.0.0 – 126.255.255.255	255.0.0.0
B	128.0.0.0 - 191.255.255.255	255.255.0.0
C	192.0.0.0 – 223.255.255.255	255.255.255.0
D	224.255.255.255 – 239.255.255.255	----------
E	240.0.0.0 – 255.255.255.255	----------

Fuente**: Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

4.2.1.1 Direcciones IP Clase, A, B, C, D y E

Para implementarse en redes de diferentes tamaños, las direcciones IP se componen de conjuntos denominados clases. Identificado como direccionamiento classful. La dirección IP consta de 32 bits dividido en parte de red y host. La secuencia de bits inicial establece su clase (Cisco).

Figura 2. Clase de Direcciones IP



Fuente: **Cisco Networking Academy Program.** CCNA 1 and 2.

- **Clase A**: Establece un máximo de 128 redes cada una con 16 millones de hosts. Los primeros 8 bytes reconocen la red y los siguientes 24 bytes son asignados a los hots.

- **Clase B**: Consiste en 16382 redes con 64000 host cada una. Los primeros 16 bytes establecen la red y los dos siguientes se asignan a los host.

- **Clase C**: Reconoce hasta 2 millones de redes cada una con 256 hots, Los primeros 24 bytes (tres octetos) se emplean establecer la red y los 8 bytes restantes se utilizan al host.

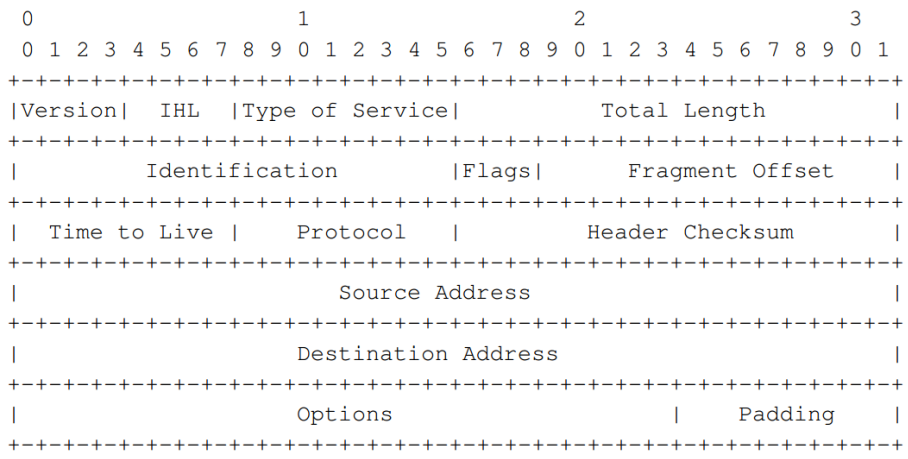
- **Clase D**: Multidifusión. Comprende las direcciones 224.0.0.0 a la 239.255.255.255. La dirección IP menor 0.0.0.0 y la mayor es 255.255.255.255.

También son importantes las direcciones de “loopback”, esta dirección permite la comunicación de procesos de internet internos en un host, la dirección de loopback está dada por la 127.0.0.1.

4.2.1.2 Formato del paquete IPv4

El protocolo IPv4 dispone de una cabecera (header) de longitud variable, consta de una parte necesaria de 20 bytes, seguido de una cadena de opciones. Teniendo en cuenta a limitaciones de la cabecera las opciones deben contener de una longitud en múltiplos de 4 bytes (Rey, 1981).

Figura 3. Formato del Paquete IPv4



Fuente: **IETF.** Internet Protocol. Marina del Rey, California: RFC 791, 1981.

- **Versión (4 bits)**: Hace referencia al formato de la cabecera internet. Es decir el IPv4.
- **IHL (4 bits)**: Longitud de cabecera, es decir registra el inicio de los datos.
- **Tipo de Servicio (8 bits)**: Establece las medidas de calidad de servicio. Diversas redes disponen de prioridad de servicio, es decir conoce el tráfico de alta prioridad como el más importante sobre el resto, establecido en tres niveles catalogados como baja demora, alta fiabilidad y alto rendimiento (Rey, 1981).
- **Longitud Total (16 bits)**: Longitud del datagrama medida en octetos. En este campo el datagrama puede llegar a obtener 65535 octetos; usualmente los host en las redes están en capacidad de recibir datagramas de hasta 576 octetos (Rey, 1981). De presentarse fragmentación del datagrama dicho campo dispondrá el fragmento del datagrama pero no el original (Ávila Mejía, 2011).
- **Identificación (16 bits)**: En este espacio se encuentra la información de identificación instaurada por el remitente con el propósito de poder reestablecer los fragmentos de un datagrama. Se presenta una característica

y es que los fragmentos de un datagrama disponen del mismo número que los idéntica (Rey, 1981) (Ávila Mejía, 2011).

- **Indicadores (Flags) DF, DM (3 bits)**: Se usa para especificar valores de fragmentación de paquetes.
- **Posición del fragmento (13 bits)**: Aquí se muestra a que posición del datagrama corresponde un fragmento del mismo (Rey, 1981).
- **Tiempo de vida (8 bits)**: Se determina el máximo tiempo que dispone un datagrama para perdurar en la red, a mayor cantidad de saltos que realice el datagrama dentro de la red sea mayor, su TTL (Time To Live) decrece hasta llegar a cero, luego de llegar a este valor el datagrama será destruido con el fin de no congestionar la red (Rey, 1981).
- **Protocolo (8 bits)**: Indica el protocolo que sigue en niveles superiores y al cual debe ser entregado el paquete, es decir el datagrama. Los protocolos de capa superior a IP que se utilizan en el modelo TCP/IP son UDP (User Datagram Protocol) y TCP (Transmision Datagram Protocol). Los valores de los protocolos de capa superior son especificados con un “Número asignado” (Rey, 1981), (Ávila Mejía, 2011).
- **Suma de control de cabecera (16 bits)**: Confirma la cabecera IP con el fin de controlar su integridad y descartar errores en el datagrama recibido. La suma de comprobación es la suma de todos los campos de 16 bits sin tener en cuenta este campo (suma de control de cabecera) (Rey, 1981).
- **Dirección de origen (32 bits)**: Muestra el número de red y de host del remitente.
- **Dirección de destino (32 bits)**: Muestra el número de red y de host del destinatario del datagrama.
- **Opciones (Variable)**: Reservado para proporcionar recursos que permitan que los protocolos de internet subsiguientes incluyan información de cabecera que no esté presente en el presente protocolo (Rey, 1981), (Ávila Mejía, 2011).

- **Valor de relleno (Variable):** En este campo se confirma que la cabecera IP emplee un multipo de 32 bits (Rey, 1981).

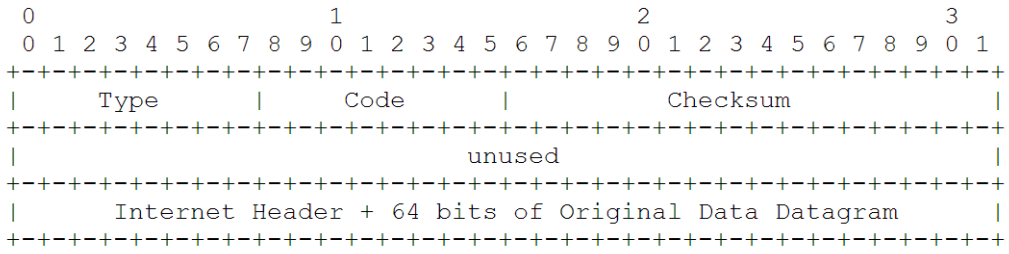
4.2.2 ICMP (Internet Control Message Protocol)

IP se le identifica como una técnica de entrega de mejor esfuerzo. Es decir que no consta de ningún procedimiento asociado que certifique la entrega de paquetes en caso que se presente un inconveniente de comunicación en la red. De igual forma nada en su estructura hace que IP confirme al emisor de que la transmisión se ha perdido. El Protocolo de control de mensajes de Internet (ICMP) es el elemento del grupo de protocolos TCP/IP que de una forma hace corrección de esta condición de IP (Network Working Group , 1981), (Cisco).

4.2.2.1 Formato del mensaje ICMP

Todos los mensajes disponen de su formato adecuado, pero inician con campos similares:

Figura 4. Formato mensaje ICMP



Fuente: **Internet Control Message Protocol**, California: RFC 792, 1981

- **Tipo (8 bits)**: Registra el tipo de mensaje, entre los cuales están:

Tabla 2. Tipos de Mensajes ICMP

Tipo	Mensaje ICMP
0	Red inaccesible
1	“Host” inaccesible
2	Protocolo inaccesible
3	Puerto inaccesible
4	Fragmentación necesaria conjunto DF (Drop Frame)
5	Fallo en la ruta de origen

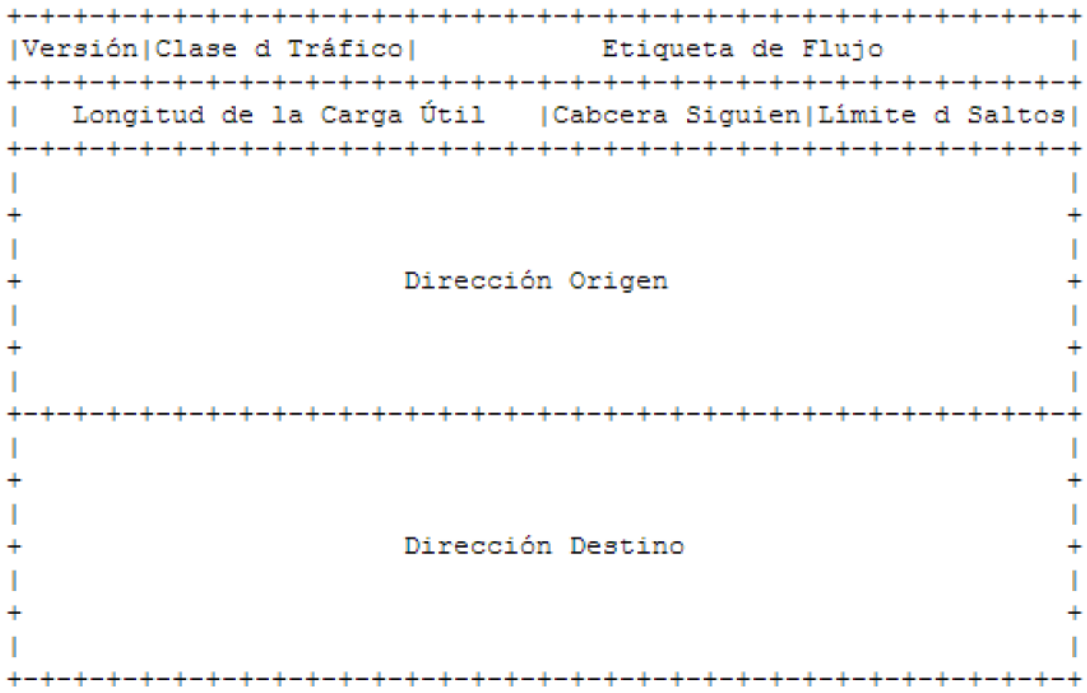
Fuente: **Internet Control Message Protocol**, California: RFC 792, 1981

- **Código (8 bits)**: Información adicional acerca del tipo de mensaje.
- **Checksum (16 bits)**: Se desarrolla en todo el mensaje ICMP.

4.2.3 IPv6 (Internet Protocol versión 6)

Protocolo de Internet Versión 6, surge de la necesidad de disponer de más direcciones IP para todos los dispositivos que la requieran, actualmente no solamente se habla de computadores sino también de smartphones, tables, stmartTvs, etc. Por este motivo la IETF, (Internet Engineering Task Force), en los últimos años viene desarrollando una nueva versión en el Protocolo de Internet, IPv6, que adopta un espacio de 16 octetos, lo cual indica que tiene 2^{128} direcciones disponibles, esto equivale a 340 sixtillones de direcciones. El protocolo IPv6 trae consigo ventajas enormes como el espacio de direcciones disponibles, autoconfiguración (Plug & Play), calidad de servicio QoS, carga útil de hasta 65535 bytes, etc (Internet Engineering Task Force, 1998).

Figura 5. Formato paquete IPv6



Fuente**: Internet Protocol, Versión 6 (IPv6)**, California: RFC 2460, 1998.

- **Versión (4 bits)**: Define la versión del protocolo de la trama que se va a enviar en este caso IPv6.
- **Clase de tráfico (8 bits)**: Hace referencia al tipo de tráfico en IPv4.
- **Etiqueta de flujo (20 bits)**: Se utiliza para admitir tráfico, para su funcionamiento requieren de tiempo real (Ávila Mejía, 2011).
- **Longitud de carga útil (16 bits)**: El paquete de red en octetos que sigue de la cabecera IPv6 (Ávila Mejía, 2011).
- **Cabecera siguiente (8 bits)**: Reconoce el tipo de cabecera que prosigue la cabecera IPv6, emplea valores similares que en la cabecera IPv4, debido a este tipo de identificación de cabeceras sucesivas desaparece en el campo de “Opciones” que estaba presente en IPv4 (Internet Engineering Task Force, 1998).

- **Límite de saltos (8 bits)**: Equivale al TTL en IPv4, es decir, en este espacio se especifica la permanencia de un datagrama IPv6 según el número de saltos que este realice hasta llegar a cero.
- **Dirección Origen**: Dirección de 128 bits (8 campos de 16 bits cada uno).
- **Dirección Destino**: Dirección de 128 bits (8 campos de 16 bits cada uno).

4.2.3.1 Direccionamiento IPv6

El formato está dado por ocho segmentos de 16 bits (2 Bytes) cada uno, los ocho campos compren 128 bits. La expresión de las direcciones en IPv4 se realizaba por medio de números decimales separados por ".", aquí en IPv6o cada segmento presente en la dirección IPv6 se identifica por medio del formato numérico hexadecimal que va desde "0000" hasta "FFFF" separados por el símbolo ":" en cada segmento (ZHANG, 2004), por lo tanto la expresión completa de una dirección de este tipo sería xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx, donde cada "x" representa un valor hexadecimal.

Al obtener expresiones de direcciones demasiadas largas el protocolo IPv6 establece algunas facilidades para enunciarlas de una forma más sencilla, entre las cuales se tienen:

Omitir los valores cero iniciales, es decir:

0xxx:0xxx:0xxx:0xxx:0xxx:0xxx:0xxx:0xxx = xxx:xxx:xxx:xxx:xxx:xxx:xxx:xxx

Al disponer de un segmento en donde las cuatro cifras hexadecimales son cero entonces se comprime el segmento en un solo cero:

xxxx:0000:xxxx:0000:xxxx:xxxx:000x:xxxx = xxxx:0:xxxx:0:xxxx:xxxx:0x:xxxx

Los grupos continuos de ceros es posible comprimirlos con "::", sin embargo, esta acción se puede realizar únicamente una sola vez por cada dirección:

xxxx:0:0:0:xxxx:0:xxxx:xxxx = xxxx::xxxx:0:xxxx:xxxx

4.2.3.2 Tipos de Direccionamiento en IPv6

- **Direcciones Unicast IPv6 Globales:** Las direcciones unicats especifican una relación uno a uno entre un origen y destino. Un paquete enviado es entregado únicamente a la interfaz nombrada con la dirección.

Los nodos IPv6 disponen de poco conocimiento sobre la organización interna de la dirección IPv6, acatando el rol que jueguen (host ó router). Un nodo puede suponer que las direcciones unicast no se montan sobre una estructura interna (Universidad Nacional del Rosario, 2006).

Tabla 3. Estructura de Dirección IPv6

128 bits
Dirección del Nodo

Fuente: **6SOS**. El Protocolo IPv6. IPv6 Servicio de Información. 2004

Un host más avanzado puede conocer el prefijo de subred de los enlaces a los que se encuentra conectado.

Tabla 4. Prefijos de subred

n-bits	128-n bits
Prefijo de Subred	**Identificador de Interfaz**

Fuente: **6SOS**. El Protocolo IPv6. IPv6 Servicio de Información. 2004

El identificador de interfaz se usa para reconocer las interfaces de un enlace, por lo cual son únicos (Universidad Nacional del Rosario, 2006).

- **Direcciones Locales de Enlace (Link-Local Unicast):** Estas direcciones, identificadas por el prefijo 1111111010 (FE80::/64), son utilizadas por los nodos cuando se comunican con nodos adyacentes en el mismo enlace (por ejemplo en una misma LAN). Son para direccionamiento en un único enlace con el fin de configuración de auto-dirección, descubrir vecinos, o cuando no

hay routers presentes (Universidad Nacional del Rosario, 2006) (Fernández, 2010).

Tabla 5. Estructura de dirección del enlace local

10 bits	54 bits	64 bits
1111111010	**0**	**Identificador de interface**

Fuente: **6SOS**. El Protocolo IPv6. IPv6 Servicio de Información. 2004

- **Direcciones Locales de Sitio (Site Local Unicast):** Estas direcciones, identificadas por el prefijo 1111111011 (FEC0), son las equivalentes al área privada de direcciones de IPV4 (10.0.0.0/8, 172.16.0.0/12, y 192.168.0.0/16).

  Admiten direccionar dentro de un entorno local. No son direcciones alcanzables (Fernández, 2010).

Tabla 6. Estructura de dirección de local de sitio

10 bits	38 bits	16 bits	64 bits
1111111011	**0**	**ID de subred**	**Identificador de interface**

Fuente: **6SOS**. El Protocolo IPv6. IPv6 Servicio de Información. 2004

Los primeros 48 bits son siempre fijos, comenzando por FEC0::/48. Después de éstos viene el identificador de subred 16-bit (Subnet ID field) que permite crear (65536) subredes dentro de la red. Después del identificador de subred viene un campo de 64 bits que identifica una interface específica dentro de la subred (Fernández, 2010).

- **Direcciones Anycast (RFC2526):** Dirección que es instaurada a más de una sola interface (que normalmente pertenecen a diferentes nodos), con la característica de que el paquete es encaminado a la interface, siendo ésta la más cercana que posea dicha dirección de acuerdo con la métrica de los protocolos de enrutamiento. Si una dirección multicast define una

comunicación “uno” a “muchos”, una dirección anycast se define como “uno” a “uno entre muchos” (Universidad Nacional del Rosario, 2006).

Tabla 7. Estructura dirección Anycast

n-bits	128-n bits
Prefijo de Subred	**0**

Fuente: **6SOS**. El Protocolo IPv6. IPv6 Servicio de Información. 2004

- **Direcciones Multicast (RFC2375):** Es un identificador para un conjunto de interfaces (normalmente en diferentes nodos) (Universidad Nacional del Rosario, 2006). Una interface puede pertenecer a cualquier número de grupos multicast. Las direcciones multicast poseen el siguiente formato:

Tabla 8. Estructura dirección multicast

8 bits	4 bits	4 bits	112 bits
11111111	**000T**	**Ámbito**	**Identificador de Grupo**

Fuente: **6SOS**. El Protocolo IPv6. IPv6 Servicio de Información. 2004

El número binario 11111111 identifica que el mensaje es multicast.

El 000T estable que los tres bits iniciales son de reserva y deben permanecer en cero. Si T = 0 identifica una dirección multicast asignada permanentemente (bien conocida) y T = 1 indica una dirección multicas temporal. Los siguientes 4 bits están reservados para futuras actualizaciones (Universidad Nacional del Rosario, 2006).

El Identificador de Grupo reconoce el grupo multicast al que se está refiriendo ya sea temporal o permanente dentro de un ámbito.

4.2.4 ICMPv6 (Internet Control Message Protocol version 6)

ICMPv6, protocolo de control de mensajes en el procesamiento del flujo de paquetes en los nodos IPv6, está diseñado para detectar errores encontrados en

la interpretación de paquetes (IEFT, 2006), también es el encargado de otras funciones en la capa de red como el “ping”.

ICMPv6 emplea operaciones de suma en la cabecera de IP de tal manera de identificar errores y forjar mensajes de control que sean perceptibles para el usuario que está en la red.

ICMPv6 apila los códigos de información de control en dos tipos de mensajes: de error y de información. Los mensajes de error son identificados por un cero “0” en el bit de orden superior, por ende los mensajes de error tienen un rango entre 0 y 127; y los mensajes de información se establecen a partir de 128 a 255 (IEFT, 2006).

4.2.4.1 Estructura de un mensaje ICMPv6

Tipo (8 bits): Indica el tipo de mensaje, el formato de la información a recibir.

Tabla 9. Mensajes de error ICMPv6

Tipo	**Mensaje ICMPv6**
1	“Host” inaccesible
2	Paquete muy grande
3	Tiempo Excedido
4	Problema de Parámetro
100	Uso Privado de Pruebas
101	Uso Privado de Pruebas
127	Reservado para la expansión de mensajes de error ICMPv6

Fuente: **Internet Control Message Protocol versión 6** California: RFC 4443, 1998.

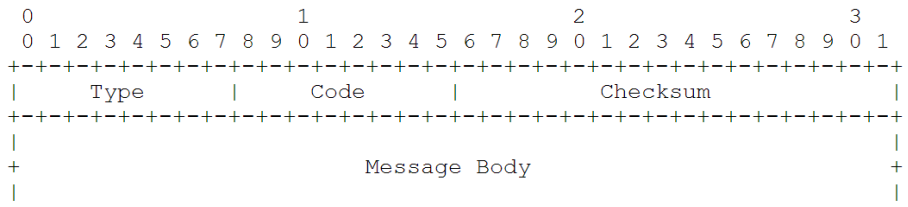
Tabla 10. Mensajes de Información ICMPv6

Tipo	Mensaje ICMP
128	Echo Request
129	Echo Reply
200	Uso Privado de Pruebas
201	Uso Privado de Pruebas
255	Reservado para la expansión de mensajes de información ICMPv6

Fuente: **Internet Control Message Protocol versión 6** California: RFC 4443, 1998.

- **Código (8 bits)**: Dependiendo del tipo de mensaje es empelado para generar un nuevo subnivel de clasificación.
- **Checksum (16 bits)**: Detecta los errores en mensajes ICMP y en para algunos de los mensajes IPv6.

Figura 6. Formato mensaje ICMPv6



Fuente: **Internet Control Message Protocol versión 6** California: RFC 4443, 1998.

4.2.5 Estrategias de Convivencia entre Protocolos IPv4 e IPv6

Las redes de datos y a su vez de dispositivos que requieren del protocolo de internet están forjando la necesidad de evolución para dicho protocolo y aunque el protocolo IPv6 existe desde la década de 1990, el cambio en los protocolos de internet no puede darse de forma radical debido a la necesidad del mundo actual por Internet.

Por lo cual es necesario que ambos protocolos coexistan durante algún tiempo con el fin de implementar IPv6 de una forma gradual (Ávila Mejía, 2011); para permitir la convivencia de ambos protocolos en las redes IPv4 se han desarrollado técnicas que estimulen la migración progresiva, entre las cuales se tienen:

- **Doble Pila (Dual Stack)**: Solución de nivel IP (RFC 4213), esta estrategia de transición requiere mantener dos pilas de los protocolos (IPv4 e IPv6) con el fin que funcionen de manera simultánea pero de forma paralela, permitiendo a los dispositivos (PC, routers, servidores) trabajar a través de ambos protocolos (Waddington, 2012).

Cada nodo en la red que utilice doble pila tendrá que definir dos direcciones (IPv4 y otra con IPv6) representado una desventaja debido a que se requiere de más procesamiento al tener actualizada las tablas de enrutamiento de ambas pilas (Ávila Mejía, 2011), (Waddington, 2012).

- **Túnel**: Consiste en un encapsulamiento de paquetes IPv6 sobre IPv4, es decir enviar (datagramas IPv6 encapsuladas) a un nodo de destino IPv6 sobre una infraestructura de red IPv4, por lo cual se genera una conexión punto a punto entre dos nodos IPv6 (Waddington, 2012).

Esta técnica se caracteriza por la diversidad de métodos que se emplean en la encapsulación en los nodos para establecer la dirección a la salida del túnel.

- **Traducción (Translation)**: Este método se utiliza de tal forma que un host que únicamente soporte IPv4 requiere establecer comunicación con uno que implante exclusivamente IPv6. Su funcionamiento se basa en la traducción de los campos comunes de las cabeceras de ambos protocolos (Ávila Mejía, 2011).

A continuación se muestra los mecanismos que utiliza cada método de transición y convivencia entre IPv4 e IPv6.

Tabla 11. Mecanismos usados en la convivencia entre IPv4 e IPv6

Nombre	Conectividad	Tipo
Doble Pila	4-to-4 over 4, 6-to-6 over 6	Doble Pila
SIIT (Stateless IP/ICMP Translation)	6-to-4, 4-to-6	Traducción
Bump-in-Stack (BIS)	4-to-6	Traducción
Bump-in-API (BIA)	4-to-6	Traducción
NAT-PT	6-to-4, 4-to-6	Traducción
MTP	4-to-6,4-to-6 (multicast)	Traducción
TRT	6-to-4	Traducción
SOCKS64	4-to-6, 4-to-6	Traducción
6over4	6-to-6 over 4	Túnel
ISATAP	6-to-6 over 4	Túnel
DSTM	4-to-4 over 6	Túnel
Configuración IP-en-IP	6-to-6 over 4, 4-to-4 over 6	Túnel
6to4	6-to-6 over 4	Túnel

Fuente: **Chang, Daniel G. Waddington - Fangzhe**. Realizing the Transition to IPv6. Bell Research Laboratories, 2012.

4.3 ENRUTAMIENTO

Un protocolo de encaminamiento proporciona técnicas para buscar un camino y guiar la información, con el objetivo de completar y actualizar su tabla de enrutamiento con las mejores rutas para intercambiar información con otras redes (Universidad de Alcalá, 2012).

4.3.1 OSPF (Open Shortest Path First)

Protocolo de enrutamiento interno basado en el algoritmo Short Path Firts para calcular la ruta más idónea. Esto significa que distribuye la información de enrutamiento entre los routers que pertenecen a un mismo sistema autónomo, fue desarrollado por Internet Engineering Task Force (Force Internet Engineering Task, 1998).

Se denomina costo a su medida métrica, donde se tienen presente factores externos como ancho de banda y comportamiento de los enlaces en cuanto a la cogestión que presenten. El funcionamiento se basa en el estado de los enlaces

existentes dentro de un plano de red donde se tienen a disposición todos los nodos.

4.3.1.1 Tipos de Mensajes

OSPF constantemente actualiza el mapa de enrutamiento de los nodos existentes dentro de una misma red y además los datos de información de estado-enlace entre estos. Esta difusión se establece por medio de los siguientes paquetes:

- **Paquetes Hello**: Cada uno de los Routers envía un paquete diciendo que vecinos tiene y la relación que mantiene con cada uno (Gil).

- **Paquetes Database Description, DBD**: Se desarrolla en el cambio de bases de datos entre nodos, además entrega información acerca de los registros contenidos (Gil).

- **Paquetes Link State Advertisements (LSA)**: Cambios en los estados de un router se notifican por medio de los mensajes LSA, a partir de esto se desarrollan las versiones que maneja cada uno de los routers (Gil).

4.3.1.2 Tipos de Áreas

Al disponer de sistemas autónomos grandes se dificulta su administración, OSPF permite fragmentarlos en áreas numeradas, siendo éstas un conjunto de redes vecinas o subredes. Cada área dispone de un número. El área 0 se conecta al Backbone que se vincula con otras áreas e integra a los demás sistemas autónomos (Cisco CCNA - CCNP, 2003).

- **Área Backbone**: Área que esta permanente en una red OSPF (núcleo de una red OSPF) estableciendo una conexión físico-lógica con otras áreas (Cisco CCNA - CCNP, 2003).
- **Área Stub**: Las áreas de este tipo no permiten información de rutas que se encuentran por fuera del sistema autónomo, como por ejemplo rutas que no son de origen OSPF, al requerirse establecer comunicación con éstas áreas

se empela una ruta por defecto (0.0.0.0/0) (Cisco CCNA - CCNP, 2003), (Redes Cisco.NET).

- **Área not-so-stubby**: Compone un tipo de área stub permitiendo importar rutas externas de sistemas autónomos y ser recibidas en el backbone debido a que se establece conexión con otro protocolo de enrutamiento (RIP, EIGRP, etc) (Cisco CCNA - CCNP, 2003) (Redes Cisco.NET).

4.3.2 OSPFv3 (Open Shortest Path First version 3)

OSPFv3 (RFC 5340), protocolo de enrutamiento IPv6 de estado de enlace, muestra una descripción de la interfaz y su manera de relacionarse con los demás dispositivos de red vecinos, dicha interfaz incluye el prefijo IPv6, la máscara de red, el tipo de red que está conectado, entre otros (CISCO).

Gran parte de OSPFv3 se compone de características similares que presenta la versión anterior, aunque se expande para proporcionar apoyo a los prefijos de enrutamiento IPv6, como por ejemplo las direcciones IP (en IPv6 son más grandes).

En OSPFv3, cada interfaz debe ser habilitada con un comando en el modo de configuración de la interfaz. Esto lo diferencia de OSPF donde las interfaces quedan automáticamente habilitadas en modo de configuración global.

En IPv6, se puede configurar diversos prefijos de direcciones en una interfaz. En OSPFv3, todos los prefijos de direcciones en una interfaz se incluyen por defecto. No es posible seleccionar algunos prefijos de direcciones para ser importados en OSPFv3; es decir todos o ninguno los prefijos de direcciones de una interfaz son importados (CISCO).

4.3.3 MPLS (Multiprotocol Label Switching)

Estándar IP de conmutación de paquetes del IEFT, reduce significativamente el procesamiento que se requiere en un paquete que ingresa al router, mejorando el desempeño de dicho dispositivo y de la red en general. Se originó con el fin de

agrupar el servicio de transporte de datos para las redes establecidas en circuitos y en paquetes. Trabaja entre la capa de enlace de datos y la capa de red del modelo OSI. Las capacidades más relevantes de dicho protocolo son: el soporte de calidad de servicio (QoS), ingeniería de tráfico, soporte para Redes Privadas Virtuales (VPNs) y soporte multiprotocolo (Moya, 2002), (Tapasco, 2008).

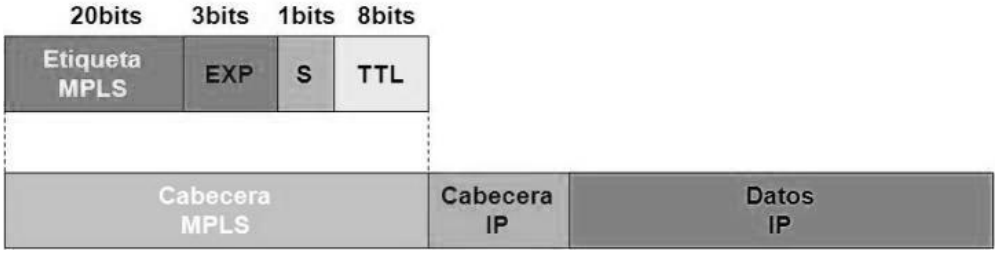
4.3.3.1 Funcionamiento de MPLS

Una red MPLS consiste de un conjunto de routers LSR (Label Switching Router) que tienen la capacidad de conmutar y enrutar paquetes en base a la etiqueta que se le ha añadido. Cada etiqueta define un flujo de paquetes entre dos puntos finales y cada flujo denominado FEC (Forwarding Equivalence Class) es diferente, tiene un camino específico a través de los LSR de la red. Así mismo los FEC, además de la ruta de paquetes contienen una serie de caracteres que definen los requerimientos de QoS del flujo. Los routers de la red MPLS no necesitan examinar ni procesar el encabezado IP, sólo es necesario reenviar cada paquete dependiendo el valor de su etiqueta. Dicha etiqueta es establecida al paquete basada en su dirección de destino, los parámetros del tipo de servicio, la pertenencia a una VPN o algún otro criterio (Moya, 2002), (Tapasco, 2008).

4.3.3.2 Cabecera MPLS

- **Label (20 bits)**: Es el valor actual, esta etiqueta determinará el próximo salto del paquete (Moya, 2002).
- **Cos o Ex (3bits)**: Influye a los algoritmos de descarte de paquetes y de mantenimiento de colas en nodos intermedios, indica la QoS del paquete mejorando el rendimiento de un tipo de tráfico (Moya, 2002).

Figura 7. Cabecera MPLS



Fuente: **Álvaro Rendón**. MPLS. Sistemas de Comunicación. Universidad del Cauca. Facultad de Ingeniería Electrónica y Telecomunicaciones. 2012.

- **Stack (1bit)**: Establece si se presentan más etiquetas, debido a que las cabeceras MPLS actúan como si estuviera una encima de otras, por lo cual de atenderá siempre la que esté más alto en la pila (Moya, 2002).

- **TTL (8bits)**: Facilita la funcionalidad de tiempo de vida del paquete permitiendo atenuar el efecto de posibles bucles que se presenten (Moya, 2002).

4.3.3.3 Elementos de una Red MPLS

- **LSR (Label Switch Router)**: Elemento que está implicado en el proceso de distribución de etiquetas permitiendo el reenvío de paquetes en el corazón de la red MPLS, para el desarrollo de este proceso consiste en instruir sobre las LSR que divulguen sus enlaces de etiquetas a otros LSRs dentro de la red (Jesús, 2002), (Dorado, 2004).

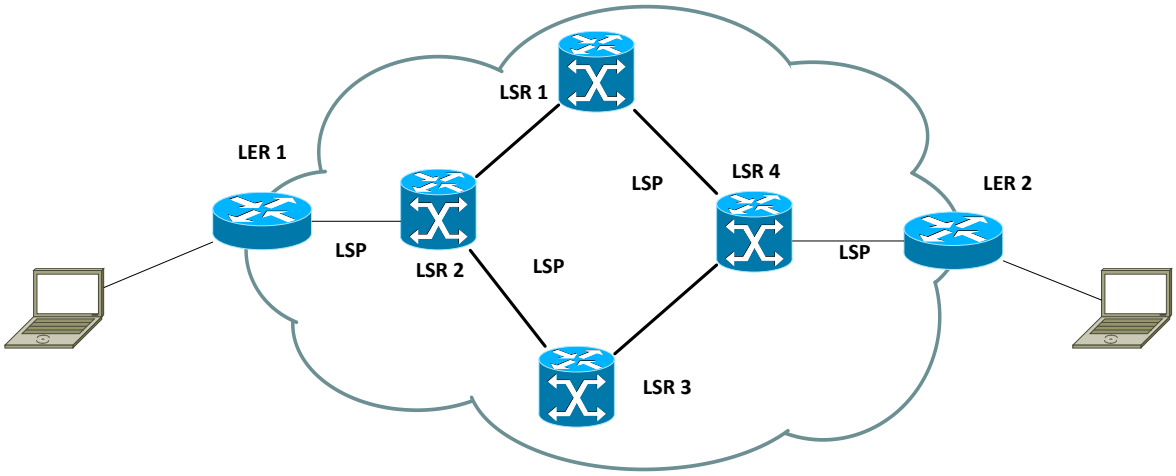
- **LER (Label Edge Router)**: Es un dispositivo que opera en el borde de la red de acceso y del dominio MPLS, el cual se encarga de insertar las etiquetas basándose en la información de enrutamiento (Jesús, 2002), (Dorado, 2004).

Soporta múltiples puertos conectados a redes distintas y envía este tráfico a través de la red MPLS después de haber establecido un camino LSP camino

utilizando un protocolo de distribución de etiquetas, también se encarga de retirar las etiquetas y distribuir el tráfico a las redes de salida.

- **LSP (Label Switched Path)**: Se llama así a cada uno de los caminos unidireccionales que un paquete toma para ir desde un LER de entrada a un LER de salida, pasando por uno o varios LSRs, es decir es un circuito virtual que se establece por la red para todos los paquetes sujetos a la misma FEC (Jesús, 2002), (Dorado, 2004).

Figura 8. Componentes del Dominio MPLS



Fuente: **Rogelio Alvez.** Fundamentos de MPLS/VPN. CERTuy

- **FEC (Forward Equivalent Class**): Conjunto de paquetes que tienen los mismos requerimientos para su transporte y son transmitidos por una misma ruta, dichos paquetes reciben un mismo trato en MPLS, un FEC está formado por todos los paquetes a los que se le puede aplicar una etiqueta específica y sólo se hace cuando un paquete ingresa a la red (Jesús, 2002), (Dorado, 2004).

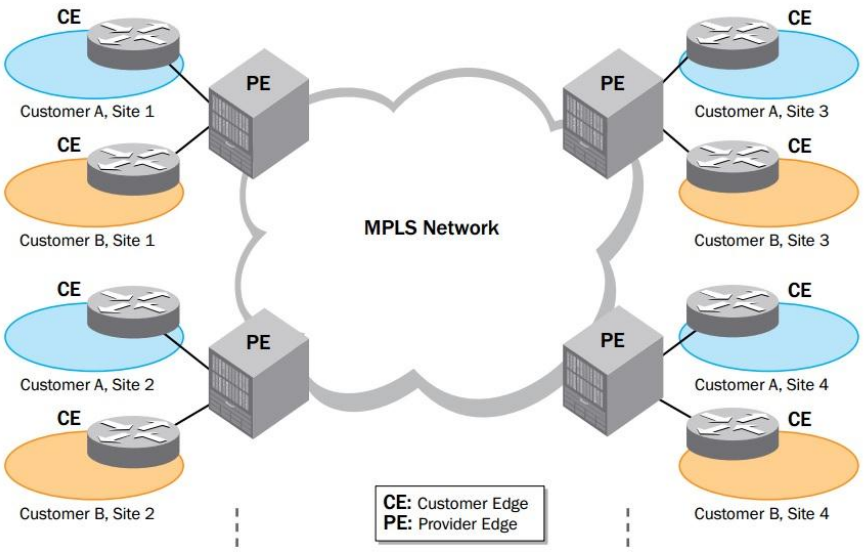
4.3.3.4 VPN (Virtual Private Network)

Durante los últimos años se ha presentado un mayor interés en el constante desarrollo de sistemas de seguridad y protección de información, entre los cuales se tiene las VPNs. Por lo cual en la capa 2 (enlace de datos) sobre MPLS se basan dos tecnologías, las “Virtual Private LAN Service” (VPLS) y las “Virtual Leased Line” (VLL) ofreciendo servicios escalables de punto-multipunto y punto a punto respectivamente.

- **VPLS (Virtual Private LAN Service):** Se define como un mecanismo para ofrecer servicios Ethernet multipunto a través de una infraestructura MPLS. Para lograr esto, en la red se simula el comportamiento de un puente IEEE 802.1D. Permite a sitios dispersos geográficamente compartir un dominio de difusión Ethernet (BROCADE Communications Systems, 2009).
- **VLL (Virtual Leased Line):** Es un servicio de VPN punto a punto, que emula el comportamiento de una línea dedicada entre dos puntos, transporta tráfico Ethernet sobre un túnel MPLS a través de una red troncal IP, es el escenario ideal para empresas que están ubicadas geográficamente distantes donde predomina un tipo de tráfico establecido (BROCADE Communications Systems, 2009).

Así mismo MPLS (como se define en el RFC 2547) dispone de estándares y proyectos relacionados que proporcionan VPNs en capa 3 (capa de red) denominadas “Virtual Routing and Forwarding” VRF.

Figura 9. Red MPLS- VRF



Fuente: **Technical Brief: Offering Scalable Layer 2 Services with VPLS and VLL**. Brocade Communications Systems. 2009

En una VPN de capa 3, cada dispositivo Provider Edge (PE) actúa como un conjunto de routers virtuales, uno por VPN. El proveedor de la red configura los miembros de la VPN de cada interfaz del router PE. El puerto de la red se limita a las redes privadas virtuales de las que es miembro y no puede abordar a dispositivos fuera de ese entorno. En el enrutamiento IP convencional es la interfaz entre cliente Edge (CE) y los dispositivos de PE (CISCO).

- **VRF (Virtual Routing y Forwarding)**: Es una tecnología IP que permite a múltiples instancias de una tabla de enrutamiento coexistir en el mismo router al mismo tiempo, estableciendo segmentos de red sin el uso de múltiples dispositivos. Debido a que las instancias de enrutamiento son independientes, pueden llegar a ser las mismas sobreponiéndose una dirección IP con otra pero sin generar conflicto alguno. VRF también aumenta la seguridad de la red y puede eliminar la necesidad de cifrado y autenticación (CISCO).

Es válido tener presente que para efectos de nomenclatura entre fabricantes y configuraciones de VPNs en capa 3. En Cisco Systems conserva el mismo nombre (VRF) y todas las políticas de tipo túnel que están aplicadas en Huawei Technologies se reconocen como VPN-Instance siendo ésta última utilizada en el desarrollo de este proyecto con respecto a la realización y obtención de resultados de carácter real.

4.4 GSN3 (GRAPHICAL NETWORK SIMULATOR)

Software (bajo licencia GPL) de código abierto que simula redes complejas lo más real posible. Todo esto sin disponer del hardware de dispositivos de red tales como routers y switches (GNS3).

GNS3 realmente utiliza los siguientes emuladores para ejecutar los mismos sistemas operativos como en las redes reales:

- **Dynamips:** Es un emulador IOS de routers Cisco (Cisco Internetwork Operating Systems) montando las plataformas 1700, 2600, 3600, 3700 y 7200, y ejecuta imágenes de IOS estándar (Cádiz, 2012).
- **VirtualBox:** Permite implementar diversos dispositivos (PC, servidores) bajo sistemas operativos Windows, Linux, MacOSX así como Juniper JunOS.
- **Qemu:** Es un emulador con código abierto, realizado para máquinas virtuales, este emulador se ejecuta sobre Xen utilizando un módulo de Linux donde se puede virtualizar diferentes máquinas de x86, servidores, etc.

4.5 ONF (OPEN NETWORKING FOUNDATION)

Para el desarrollo de una nueva arquitectura y tecnología a nivel mundial se fundó la ONF (Open networking Foundation), organización sin ánimo de lucro para un comercio mutuo, es decir, fundada por diferentes empresas para mejorar la creación de SDN (Software Define Networking) y dar la estandarización del protocolo OPENFLOW (OFN, 2014).

Esta organización fue fundada por Deutsche Telekom, Facebook, Google, Microsoft, Verizon y Yahoo. Hoy en día tiene más de 120 empresas afiliadas con diferentes proveedores de equipos que usan esta tecnología. Dicha fundación tiene como iniciativa acelerar e innovar a través del software las redes de telecomunicaciones, a nivel inalámbrico y en la nube (OFN, 2014).

4.6 SDN (SOFTWARE DEFINE NETWORKING)

Dentro de las Arquitecturas de red que hoy en día se trabajan y ayudan a los operadores en cuanto a la gestión de red, velocidad y QoS (Quality and service) surge una nueva tecnología, dicha tecnología inicia a principios del 2008 en la Universidad de Stanford en California y la Universidad de Berkley las dos ubicadas en EE.UU.

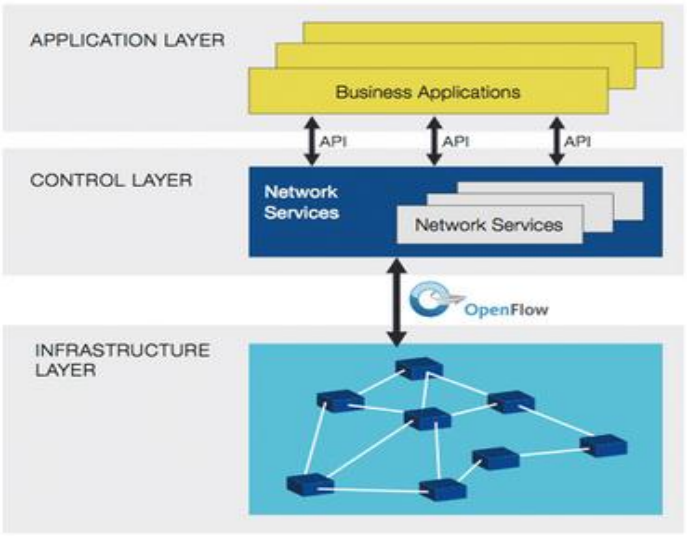
Esta tecnología es denominada SDN y posee una arquitectura que surge con la necesidad de manejar y distribuir los paquetes de forma dinámica, manejable y adaptable a diferentes redes. Estas características ayudan a manejar un gran ancho de banda, lo cual destaca el desacoplamiento de funciones de la red y el control de la misma para ser directamente programable desde el gestor de la misma, es decir una separación del sistema que toma las decisiones acerca de dónde se envía tráfico (plano de control) de los sistemas subyacentes para el reenvío de tráfico al destino seleccionado (plano de datos). Los inventores y vendedores de estos sistemas sostienen que esto simplifica la creación de redes. (OFN, 2012).

4.6.1 Arquitectura de Red

Dicha arquitectura esta desarrolla para ser programable y tener cambios en tiempo real, lo cual implica tener más aplicaciones y mayor rentabilidad para una empresa, además SDN permite la implementación de diferentes servicios en cuestión de minutos (lo que hoy en día se hace en meses), obteniendo un enrutamiento, multidifusión, seguridad, control de acceso, gran ancho de banda y mayor optimización de la red.

- **Directamente programable**: El control de la red se puede manejar y programar de diferentes formas para así acoplarse a la necesidad de la red (ONF, 2013).
- **Ágil**: Se adapta fácilmente al flujo de tráfico de toda la red.
- **Gestión de forma centralizada**: Es una red inteligente basada en los controladores de SDN, en donde el switch juega un papel fundamental para la lógica y programación de las aplicaciones (ONF, 2013).
- **Programación configurada**: Dentro de la red, SDN permitirá que no sólo se configuren los paquetes, sino que además se administren, aseguren y se optimicen recursos de red haciéndola más rentable (ONF, 2013).
- **Basados en estándares abiertos e independientes del proveedor**: Dicha arquitectura se implementa con estándares abiertos los cuales simplifican el diseño de la red (ONF, 2013).

Figura 10. Arquitectura de red definida por software**.**



Fuente: **ONF**. Open Networking Foundation, SDN Definition. 2014

- **Aplicación SDN (SDN App)**: Son programas que de manera explícita, directa, y mediante programación comunican sus requerimientos de red y el comportamiento de la red que desee para el controlador SDN.

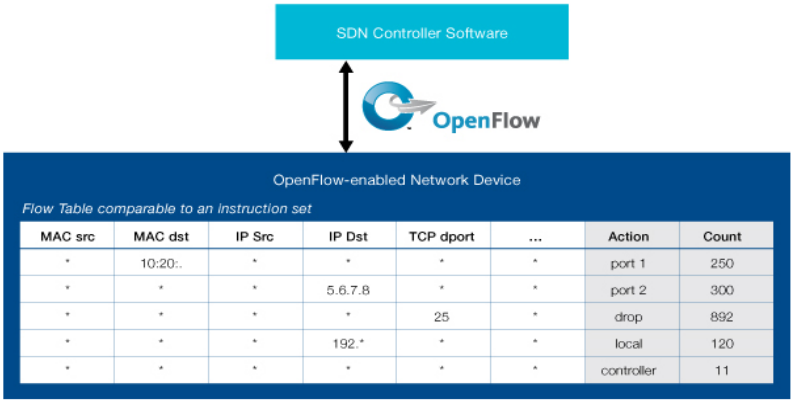
- **Controlador SDN**: Es la entidad lógica a cargo de la conversión de los requisitos de la capa de aplicación hasta los Datapaths. Un controlador SDN ni establece ni impide detalles de implementación con otros controladores, conexión jerárquica de otros controladores e interfaces de comunicación.
- **SDN Datapath**: Comprende la Capa de Infraestructura. Son los múltiples dispositivos de red lógicos. Un SDN Datapath dispone de un conjunto de uno o más motores de reenvío de tráfico y de funciones de procesamiento.

4.6.2 OpenFlow

Es un estándar para el control y las capas de reenvió de una arquitectura SDN, este permite el acceso directo a la manipulación y el plano del reenvió de dispositivos como router y switches (OFN, 2014).

Este protocolo especifica primitivas básicas que se pueden utilizar por una aplicación de software externo para la programación de los paquetes y de los dispositivos de red, es similar a una CPU, por ejemplo permitirle a un servidor establecer la ruta de reenvío de paquetes que corresponden en una red de switches, las medidas que involucren el movimiento de paquetes están conjuntadas permitiendo a la red ser programada arbitrariamente de los switches. (ONF, 2013), (OFN, 2014).

Figura 11. Instrucciones de Openflow



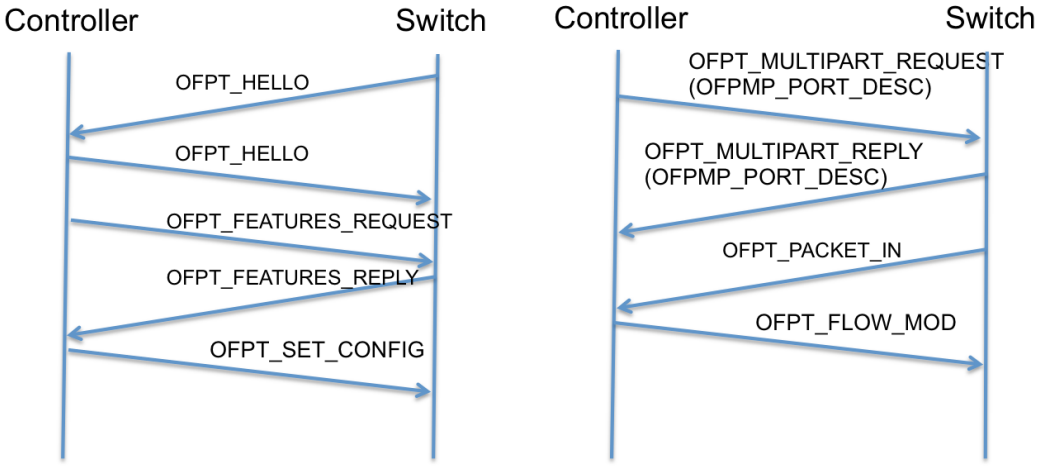
Fuente: **ONF**. Open Networking Foundation, SDN Definition. 2014

Dentro de este protocolo se utiliza un componente vital para cualquier arquitectura de red, el switch, éste se ocupa hoy en día con tablas y flujos de datos, para así tener una estadística de cada uno, trabajando en cuanto a firewalls y Qos. OpenFlow lo que busca es tener un switch más inteligente, es decir, en capa 3 para permitir la visibilidad y apertura en la red, darle un mejor camino a los paquetes y flujo de datos, por tal motivo, el tema de enrutamiento y VLANs ya no serán utilizadas, haciendo la red más eficaz en cuanto a velocidad y ancho de banda (OFN, 2014) (Cisco). Ahora bien un switch OpenFlow maneja tabla de flujos indicando cómo será el proceso de cada uno estos mensajes o paquetes, un canal con una seguridad remota permitiendo que tanto comandos y paquetes se envíen en el switch utilizando dicho protocolo, por último se tiene el controlador el cual añade o elimina flujos para poder experimentar dentro de la red, dicho controlador es el gestor de toda la red (OFN, 2014) (Cisco).

4.6.2.1 Mensajes OpenFlow

Al tener una red establecida en SDN se presenta un conjunto de mensajes entre el controlador OpenFlow y el switch, a continuación se muestra la descripción de cada uno de ellos.

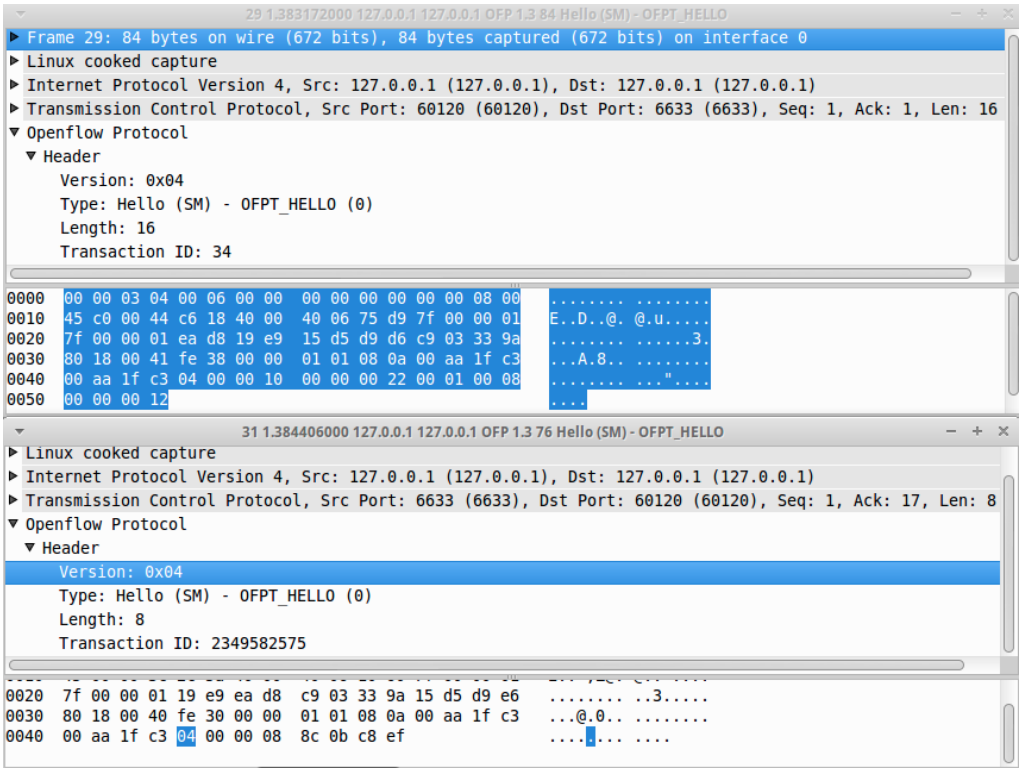
Figura 12. Estructura de mensajes OpenFlow



Fuente: **SDN Hub**. OpenFlow. Tutorial. Understanding OpenFlow Messages

- **Configuración de la conexión:** El switch inicia una conexión TCP estándar (o TLS) al controlador. Cuando se establece una conexión OpenFlow, cada entidad debe enviar un mensaje de OFPT_HELLO con la versión del protocolo establecido soportado por el remitente (SDN Hub, 2014). En la Figura 14 se aprecia la negociación del protocolo OpenFlow.

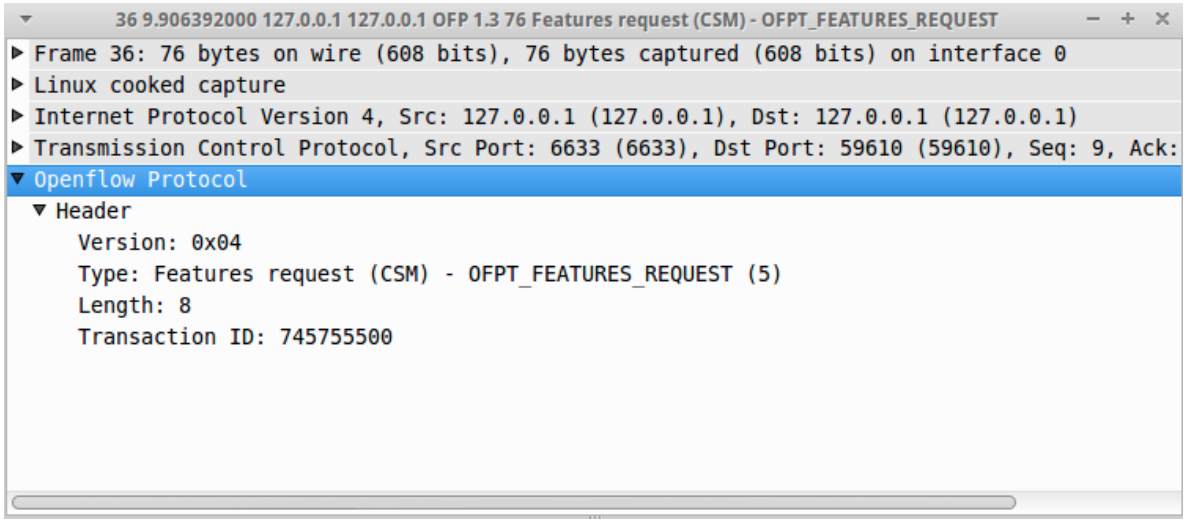
Figura 13. Negoción del Protocolo OpenFlow



Fuente: **SDN Hub**. OpenFlow. Tutorial. Connection Setup

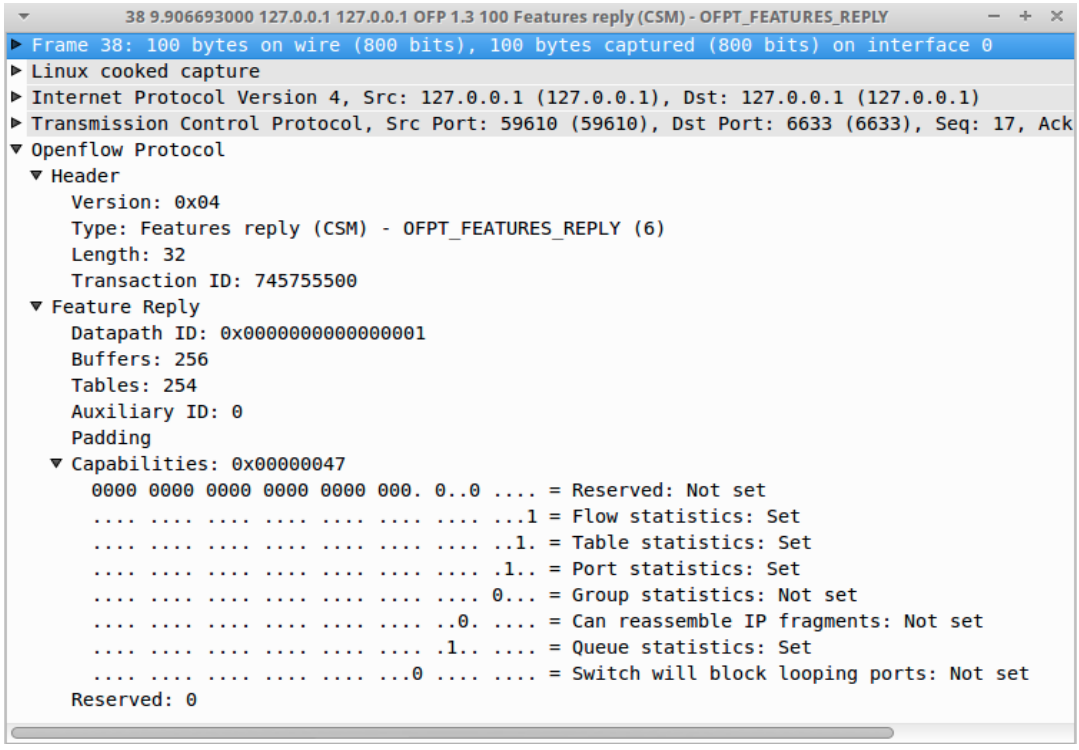
- **Pedido de Funciones (Reply – Request)**: Después de establecer una sesión, el controlador envía un mensaje de OFPT_FEATURES_REQUEST. Este mensaje sólo contiene un encabezado. Así mismo el switch responde con un mensaje de OFPT_FEATURES_REPLY (SDN Hub, 2014).

Figura 14. Mensaje Feature Request



Fuente: **SDN Hub**. OpenFlow. Tutorial. Feature Request

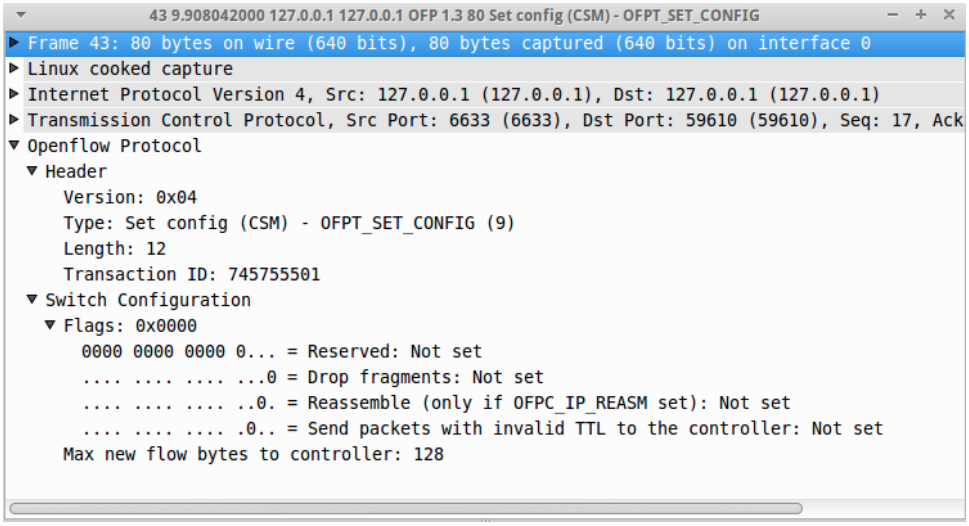
Figura 15. Mensaje Feature Reply



Fuente: **SDN Hub**. OpenFlow. Tutorial. Feature Reply

- **Configuración Set**: El controlador envía el mensaje OFPT_SET_CONFIG al switch. Incluye el conjunto de banderas y el máximo de bytes del paquete.

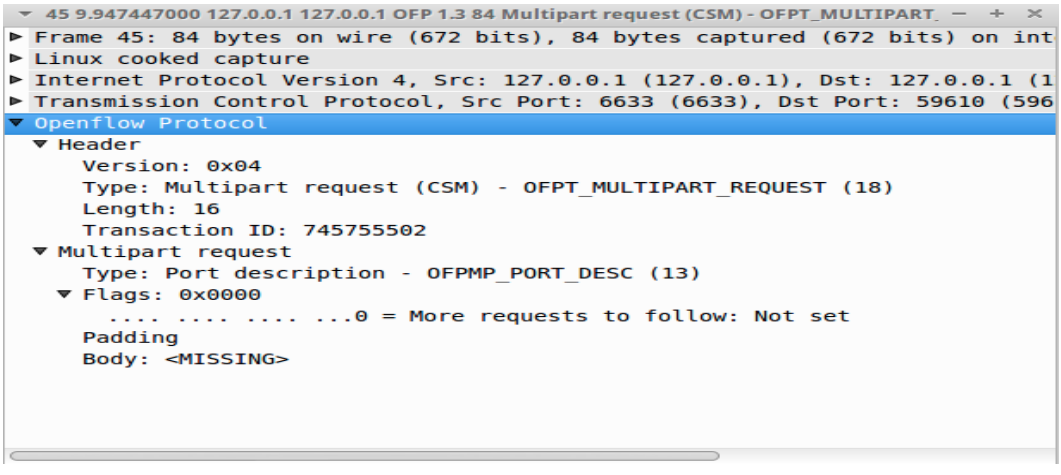
Figura 16. Mensaje Set Configuration



Fuente: **SDN Hub**. OpenFlow. Tutorial. Set Configuration

- **Multipart Request**: El controlador puede solicitar el estado actual de la ruta de datos usando el mensaje OFPT_MULTIPART_REQUEST.

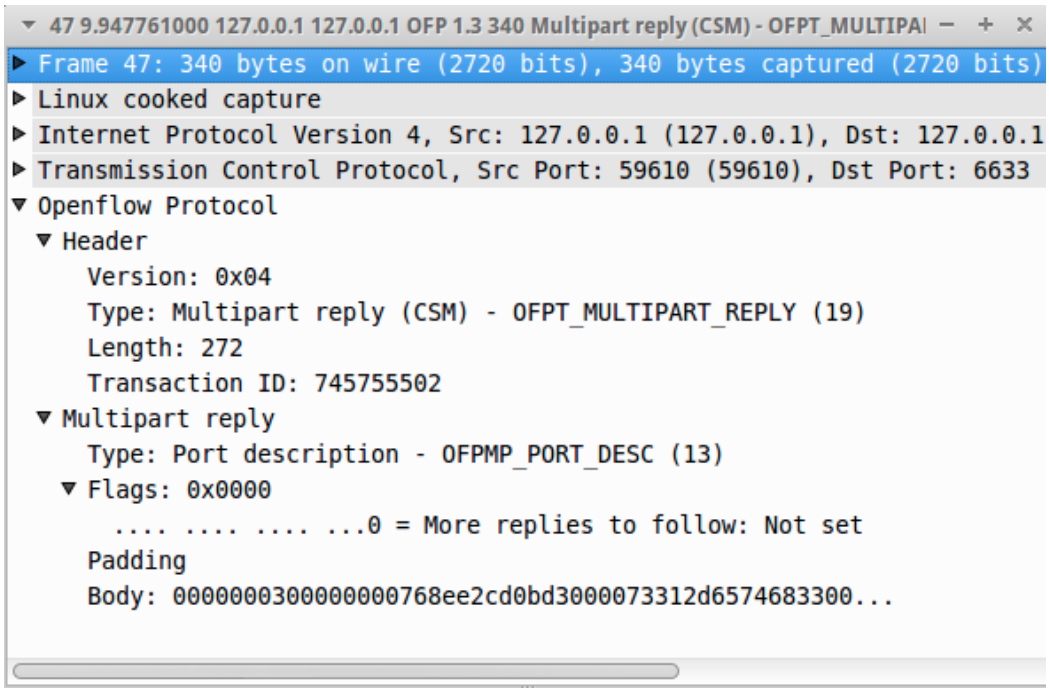
Figura 17. Mensaje Multipart Request



Fuente: **SDN Hub**. OpenFlow. Tutorial. Multipart Request

- **Multipart Reply:** El switch responde un mensaje de todos los puertos activos en él por medio del FEATURE_REPLY.

Figura 18. Mensaje Multipart Reply

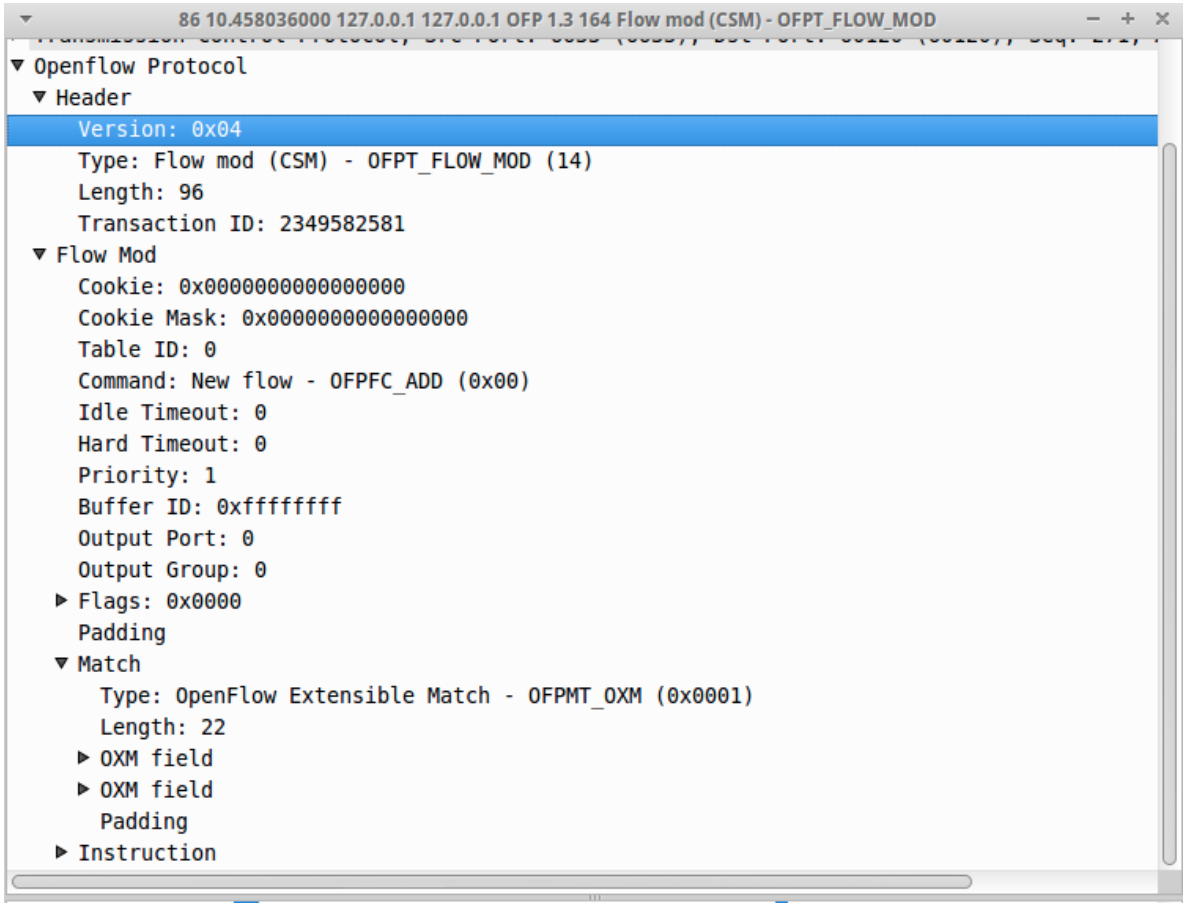


Fuente: **SDN Hub**. OpenFlow. Tutorial. Multipart Reply.

- **Flow Mod**: Los flujos pueden ser de manera proactiva (flujos de pre-instalación) o de manera reactiva (para mensajes packet_in) enviados desde el controlador.

En el siguiente caso, el controlador instala un nuevo flujo, lo que demuestra que aparte del conjunto de parámetros como prioridad 1.0 (idle_timeout, etc) también aparece en la estructura de instrucciones los nuevos parámetros especificados en 1.3 (OF_Flow_Mod). Ver Figura 20.

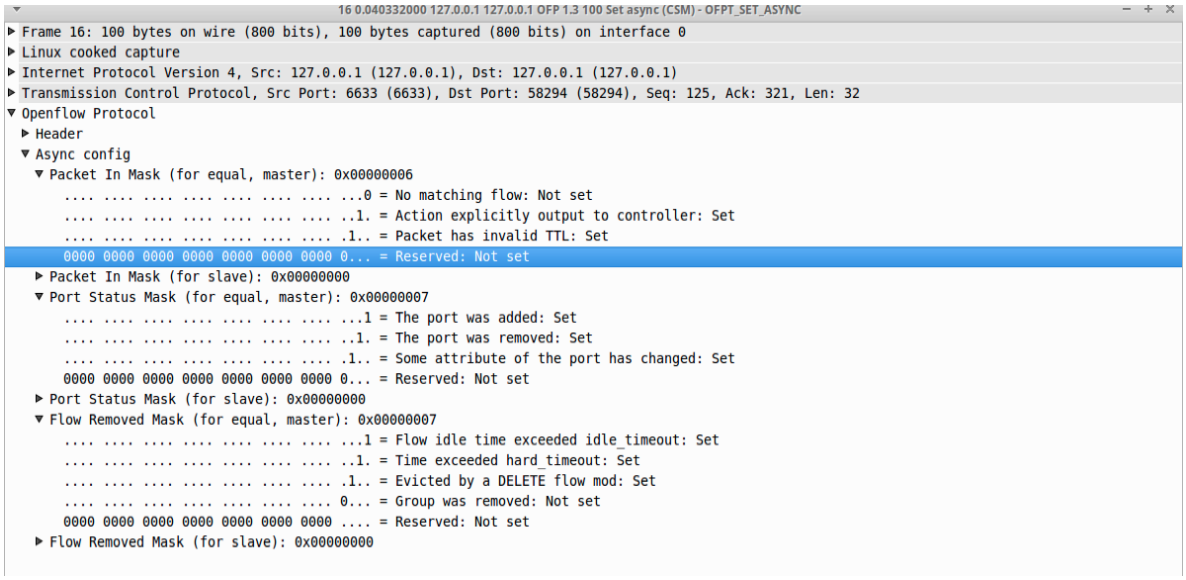
Figura 19. Mensaje Flow Mod



Fuente: **SDN Hub**. OpenFlow. Tutorial. Flow Mod.

- **Set Async Configuration Message**: Los mensajes asíncronos se envían desde un switch en el controlador. El conjunto de mensajes soportados por el protocolo OpenFlow incluye mensajes “Packet-Ins, Flow-Removed, Port-Status or Error”. Cuando el switch se conecta al controlador, el controlador puede establecer el tipo de mensajes que desea recibir en su canal de OpenFlow.

Figura 20. Mensaje Set Async Configuration



Fuente: **SDN Hub**. OpenFlow. Tutorial. Set Async Configuration Message.

4.6.3 Software y Controladores

Dentro del trabajo realizado para la implementación de este proyecto se tienen diferentes alternativas, dentro de estas se encontraron algunas de software:

- **Open vSwitch**: Este es un software que utiliza varias capas virtuales, está bajo la licencia del código de apache 2.0. Dicho software está diseñado para permitir la automatización de la red, por otro lado apoya las interfaces de gestión y protocolos como Netflow, sFlow, SPAN entre otros (Open vSwitch).
- **Índigo**: Esta es una implementación de OpenFlow con código abierto el cual funciona con características similares a las de los de los switches (OpenFlowHub, 2010).
- **Pantou**: Es una plataforma de software abierto para cualquier tipo de hardware de conmutación, este también convierte un router en un punto de acceso inalámbrico; el desarrollo de Openflow esta implementado en cuanto a una aplicación llamada OpenWrt la cual es una distribución de Linux para estos dispositivos (OpenFlow, 2011).

- **Nettle**: Este es un paquete que proporciona datos que modelan los mensajes en el protocolo OpenFlow, también tiene funciones que implementan serialización entre tipos de datos y sus representaciones binarias en el protocolo, se puede decir que es una biblioteca OpenFlow escrita en Haskell (hackage and Cabal 1.20.0.1).

En cuanto a las plataformas, existen diversas variedades entre las cuales resaltan las más importantes:

- **POX**: Es una plataforma de rápido desarrollo en cuanto a la creación de software de control de una red, dicho elemento trabaja con lenguaje Python (Nox).
- **Iris:** Es un controlador OpenFlow que tiene como características, la escalabilidad horizontal de la red, alta disponibilidad en cuanto a fallos presentados y multidominio soportado con una red basada en OpenFlow (IRIS:).
- **MUL:** Controlador basado en lenguaje C, diseñado para obtener un rendimiento y fiabilidad acorde a la red (Open Mul, 2012).

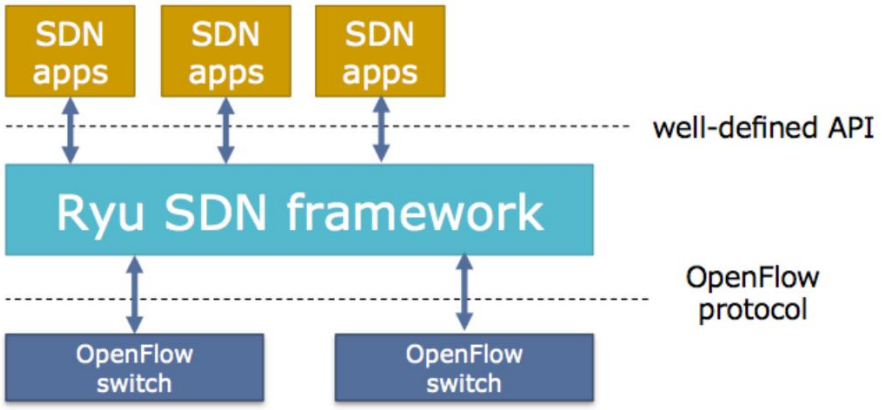
4.6.3.1 Plataformas a usar para el Desarrollo del Proyecto

Teniendo en cuenta las características de las plataformas y controladores (descritos anteriormente), para el desarrollo de este proyecto se muestran con mayor detalle las que se ejecutarán para el desarrollo de este proyecto

- **POX:** Es una plataforma para el rápido desarrollo y creación de prototipos de software de control de red usando Python. Es un nivel muy básico que presenta un creciente número de marcos (incluyendo NOX, Reflector, Trema, etc) para ayudar a escribir un controlador OpenFlow. Además de ser un marco para la interacción con switches OpenFlow (Nox). Es ideal para la navegación en SDN usando Python en Windows, Mac OS o Linux.

- **RYU:** “Flujo” Es un software basado para la estructura y control de redes SDN. Dispone de componentes API (Application Programming Interface) que hacen que sea fácil para los desarrolladores crear nuevas aplicaciones de gestión y control de red (Network Starup Resource Center).

Figura 21. Estructura de RYU



Fuente: **RYU** OpenFlow Controller. University of Oregon. NSRC

RYU es compatible con varios protocolos para la gestión de dispositivos de red, tales como OpenFlow, Netconf, OF-config, etc. Todo el código está disponible gratuitamente bajo la licencia Apache 2.0 (Ruy SDN Framework Community, 2014).

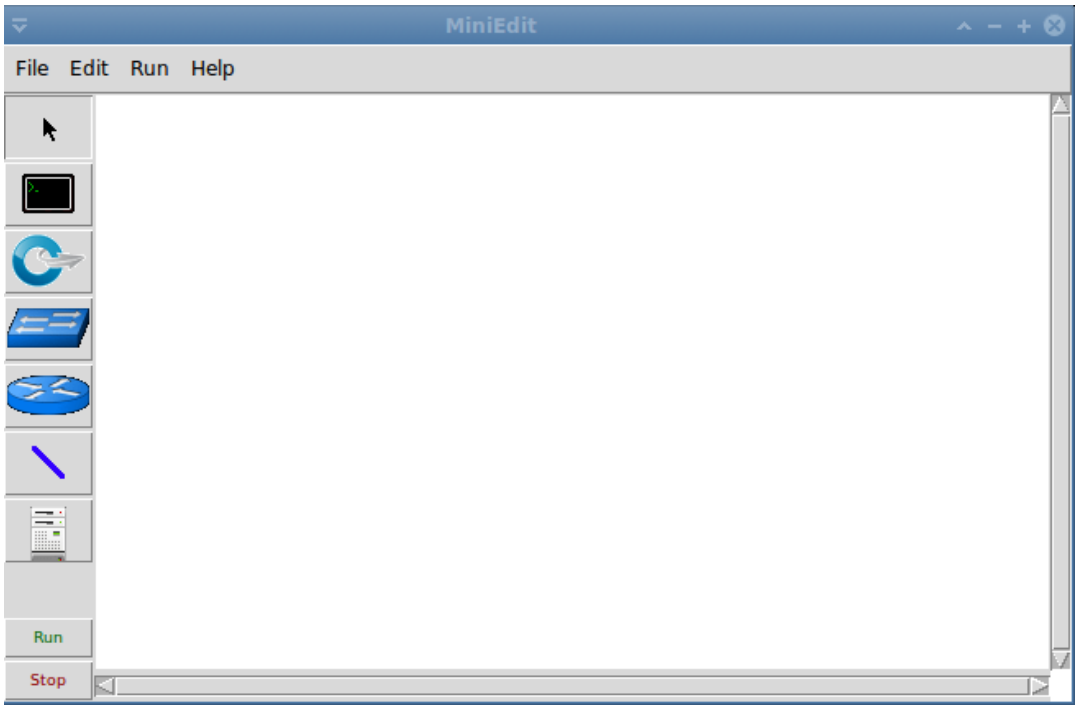
- **OpenDaylight:** Open Source Programmable Networking Platform, es una plataforma abierta para la programación de redes SDN y NFV (Network Functions Virtualization) sin importar su tamaño y escala. Es una combinación de componentes que incluye un controlador totalmente conectable, interfaces, protocolos de plug-ins y aplicaciones. Actualmente empresas proveedoras de servicios, de equipos y la academia trabajan en el despliegue y comercialización de soluciones basadas SDN y NFV (Linux Foundation, 2014).

- **Open vSwitch:** Es una implementación para la producción de un switch virtual multicapa, licenciado bajo el código abierto Apache 2.0. Está diseñado para permitir la automatización de la red y su expansión por medio de la programación, sin dejar el soporte a las interfaces estándar de gestión y protocolos (NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag). Además está orientado para permitir su distribución a través de múltiples servidores físicos similares a vNetwork switch virtual distribuido por VMware o Nexus 1000V de Cisco (Open vSwitch).
- **MiniNet:** Permite la creación de una red virtual con características reales. Se ejecuta como un conjunto de sistemas principales finales, switches, routers y enlaces en un sólo núcleo de Linux. Utiliza una virtualización ligera logrando hacer un sólo sistema que se asemeje a una red completa que emplea el mismo kernel y el código de usuario (Mininet Team , 2014) (Bob Lantz, Nikhil Handigol, Brandon Heller, and Vimal Jeyakumar, 2014).

Los programas que se ejecutan pueden enviar paquetes a través de lo que parece ser un verdadero entorno de interfaz Ethernet, con una velocidad de enlace y retardo así mismo los paquetes son procesados por un switch Ethernet, con una cantidad de colas dada (Bob Lantz, Nikhil Handigol, Brandon Heller, and Vimal Jeyakumar, 2014).

MiniNet además permite usar de una aplicación gráfica llamada **MiniEdit**, siendo ésta un entorno gráfico para la creación de una red SDN, dispone de dispositivos como routers, switches (OpenFlow), conector de red y el controlador.

Figura 22. Aplicación grafica de MiniNet (MiniEdit)



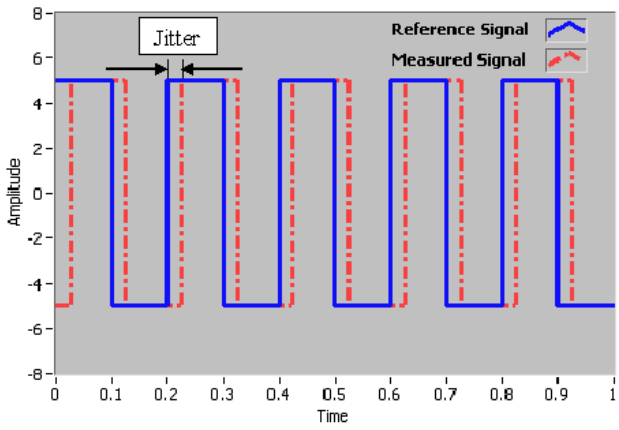
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

4.7 VARIABLES A CONTROLAR

4.7.1 Jitter

El jitter es la variación en cortos periodos de los instantes significativos de una señal digital con respecto a su posición ideal en el tiempo, está comprendido entre los paquetes de datos. La medición del jitter se realiza en segundos y expresa la distorsión que ocurre entre el patrón original de la señal o datos enviados sobre la calidad de los llegados (Damw). La cantidad de jitter reconocido depende en gran medida de la aplicación empleada.

Figura 23. Jitter Measuremet



Fuente: **An Introduction to Signal Generation**. National Instruments. 2006.

En cuanto a la medida del jitter se destacan diferentes maneras:

- Periodo del Jitter
- Ciclo periódico del jitter
- Jitter a largo plazo
- Fase de Jitter
- Intervalo de Error de Tiempo (TIE) (SITIME).

Se le conoce latencia a la suma de los retardos en la red. Están formados por el retardo de propagación y el de transmisión (depende el tamaño del paquete), retardo por el procesamiento (causado porque los switches o routers envían el paquete luego de ser totalmente diligenciados en una memoria buffer) y el retardo de procesamiento (indispensable para reconocer encabezados, errores, direcciones, etc) (Calidad de Servicio en Redes IP).

4.7.2 Delay

Se define como el intervalo de tiempo de transmisión de datos de un emisor hasta un receptor (Mejía F. , Análisis de Parametros de QoS, 2011), el proceso de medida de dicho elemento se utiliza por medio de ping, este se puede utilizar para calcular la latencia de ida y vuelta.

En contexto de redes es el momento gastado por propagación a través del medio de red y el hardware del adaptador, así como los tiempos de ejecución del software (QLogic).

En telecomunicaciones una de las causas de este retardo se da cuando ésta fluye a través de una gran cantidad de componentes y subsistemas de comunicación situados en el sistema receptor así como en la red. Cada uno de esos componentes puede ser caracterizado por su velocidad de procesamiento y por la capacidad de almacenamiento donde los datos esperan para ser procesados (Mejía F. , Análisis de Parametros de QoS, 2011).

Por otro lado para medir la latencia en una red se puede realizar de varias maneras:

- **Netperf**: Se utiliza en la capa de Aplicación en un sistema remoto.
- **Ping**: Medidas de latencia entre pares existentes por medio de protocolos.
- **Reenvió de IP**: Contribución de latencia de red, hardware, fimware, capa IP (RFC 2544).

5. DESARROLLO DEL PROYECTO

Este proyecto de grado tiene como objeto analizar el desempeño de SDN con respecto a redes IP soportadas en protocolos de enrutamiento convencionales. Para cumplir con este objetivo se utilizará una metodología experimental enfocada en tomar datos experimentales de Delay y Jitter de diferentes entornos de red.

Estos entornos de red se basarán en los Routes con los que cuenta la Universidad y con base en el número de equipos, se plantearán escenarios de simulación para tener herramientas de comparación y posterior análisis.

Teniendo en cuenta lo anterior se procede a elegir que herramientas de simulación se utilizarán para el desarrollo del proyecto.

Según lo mencionado en el marco teórico para el desarrollo de este proyecto de grado se decidió optar por el simulador GNs3 y utilizar OpenVswitch y los controladores POX y ovs-controller los cuales tienen ciertas ventajas que a continuación serán mencionadas.

Tabla 12. Simuladores de Redes IP, GNs3 de CISCO y eNSP De HUAWEI.

SIMULADORES DE REDES IP CONVENCIONALES	
GNs3	**eNSP**
Software que proporciona una interfaz gráfica de usuario donde se pueden realizar diferentes entornos de red con referencias de routers reales CISCO de cualquier serie simulando funciones reales.	Simula algunas funciones y características de Huawei AR routers y series x7 interruptores .Simula las computadoras personales, las nubes y funciones de configuración de dispositivo, utiliza tarjetas de red real para conectarse a dispositivos de red reales. Tiene Problemas para al realizar protocolos MPLS en IPv4 e IPv6, BGP y demás.

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Tabla 13. Software y Controladores de SDN (Software Defined Networking).

SDN (SOFTWARE DEFINED NETWORKING)					
Software			**Controladores**		
	Lenguaje	**Descripción**		**Lenguaje**	**Descripción**
OpenVswitch	C/Python	Utiliza varias capas virtuales, licencia del código de apache 2.0. Diseñado para permitir la automatización de la red, apoya las interfaces de gestión y protocolos como Netflow, sFlow, SPAN entre otros	**POX**	Python	Plataforma para el desarrollo y creación de prototipos de software de control de red. Ayuda a escribir un controlador OpenFlow. Además de ser un marco para la interacción con switches OpenFlow. Ideal para la navegación en SDN usando Python en Windows, Mac OS o Linux.
Pica8:	C	Plataforma de software abierto para el interruptor de hardware de conmutación chips que incluye desarrollo de L3 L2 y soporte para OpenFlow	**IRIS:**	Java	Controlador OpenFlow que tiene escalabilidad horizontal de la red, alta disponibilidad en cuanto a fallos presentados y multidominio soportado con una red basada en OpenFlow
Indigo:	C	implementación de OpenFlow con código abierto el cual funciona con características similares a las de los de los switches	**MUL**	C	Controlador diseñado para obtener un rendimiento y fiabilidad acorde a la red
Pantou:	C	plataforma de software abierto para cualquier tipo de hardware de conmutación, el desarrollo de Openflow esta implementado en cuanto a una aplicación llamada OpenWrt la cual es una distribución de Linux.	**NOX:**	C++/Python	Fue el primer controlador OpenFlow solo viene para una versión, hasta el momento.
OpenFaucet:	Python)	Aplicación Python pura con OpenFlow 1, se puede utilizar solo dos interruptores.	**Floodlight**	Java	El controlador OpenFlow se basa en Java. Originario desarrollado por David Erickson en Stanford

OpenFlowJ:	Java	Desarrollado en Java, solo contiene una versión del OpenFlow hasta la fecha	**Maestro:**	Java	Es un tipo de sistema de manejo sobre OpenFlow para orquestar las aplicaciones de control de red.
Oflib-node:	Javascript	Es una biblioteca de protocolo OpenFlow para el nodo. Convierte entre los mensajes del protocolo OpenFlow en objetos de Javascript.	**Ryu:**	Python	Ryu es un sistema operativo de red de código abierto (NOS) que apoya OpenFlow.
Nettle:	Haskell)	OpenFlow escrito en Haskell	**ovs-controller**	C	controlador empaquetado con Open vSwitch

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Además con planteamiento anterior se dan a conocer los requerimientos necesarios para identificar las prestaciones SDN con respecto a redes IP convencionales, logrando cumplir con los objetivos propuestos. Por otro lado se desarrollará un entorno de red en el cual se utilizaran los equipos HUAWEI de la Universidad Santo Tomás, trabajando protocolos como OSPF y MPLS (desarrollados en IPv4 e IPv6). Por otro lado, SDN se desarrolló mediante el protocolo OpenFlow (IPv4 e IPv6). Por último se realizara dicha red en simulación por medio del software GNS3 y una maquina Virtualizada de Linux para SDN, donde también se desarrollarán las pruebas pertinentes para probar esta nueva tecnología.

5.1 RECURSOS DE TOPOLOGÍA

5.1.1 Recursos Físicos

- Cuatro Routers Huawei Serie AR1220.
- Dos Routers Huawei Serie AR2220.
- Dos computadores con sistema Operativo MS Windows 7.
- Dos computadores virtualizados con sistema Operativo Ubuntu 14.04
- Un computador con memoria RAM de 3GB, procesador Intel(R) Core(TM) i3-2320M CPU@ 2.10 GHz, Sistema Operativo 32 Bits.

- Un computador con memoria RAM de 8GB, procesador Intel(R) Core(TM) i7-2670QM CPU@ 2.20 GHz, Sistema Operativo 64 Bits.

5.1.2 Recursos De Software

- Simuladores GNS3 y SDN
- OpenVswitch, POX, RYU.
- Routers CISCO serie 7206VXR.
- Routers CISCO serie 3725 (R7000).
- Sniffer Wireshark

5.2 DISEÑO DE EXPERIMENTOS

Cumpliendo los objetivos de este proyecto se realizaron pruebas y experimentos analizando las variables jitter y delay en diferentes escenarios de red, divididas en pruebas físicas y simuladas con la misma topología encontradas en el capítulo 7 análisis de resultados.

Dado que se tiene una metodología experimental a continuación se muestra el diseño de una serie de experimentos (limitados por él número de equipos con el que cuenta la Universidad) para obtener datos de Delay y Jitter en diferentes escenarios de red, reales y simulados para posteriormente realizar el análisis de resultados.

5.2.1 PRUEBAS FÍSICAS

Para el desarrollo de las pruebas físicas se realiza el siguiente direccionamiento tanto en IPv4 como en IPv6:

Tabla 14. Direccionamiento IPv4 para los Routers físicos HUAWEI.

Interfaz/Rx	R1	R2	R3	R4	R5	R6
Gb 0/0	192.168.0.2/24	10.0.1.1/8	10.0.1.2/8	10.0.3.1/8	10.0.3.2/8	192.168.1.2/24
Gb 0/1	10.0.0.1/8	10.0.0.2/8	10.0.2.1/8	10.0.2.2/8	10.0.4.1/8	10.0.4.2

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Tabla 15. Direccionamiento IPv6 para los Routers físicos HUAWEI.

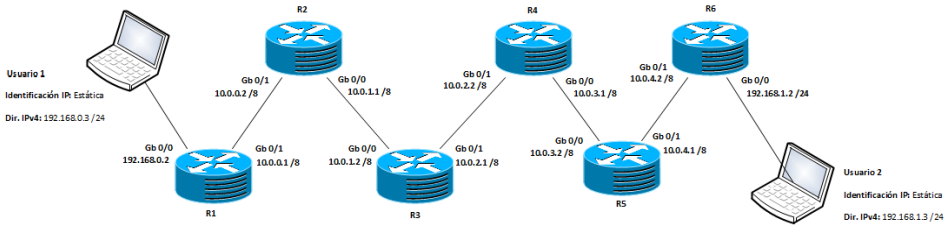
Interfaz/Rx	R1	R2	R3	R4	R5	R6
Gb 0/0	fc00:1::1/64	fc00:3::1/64	fc00:3::2/64	fc00:4::2/64	fc005::2/64	fc00:7::1/64
Gb 0/1	fc00:2::1/64	fc00:2::2/64	fc00:4::1/64	fc00:5::1/64	fc00:6::1/64	fc00:6::2/64

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

5.2.1.1 Experimento 1: Enrutamiento OSPF (IPv4)

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red sobre el protocolo OSPF en IPv4 utilizando un direccionamiento de clase A y C, para medir jitter y delay.

Figura 24. Topología de Red con direccionamiento de IPv4



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

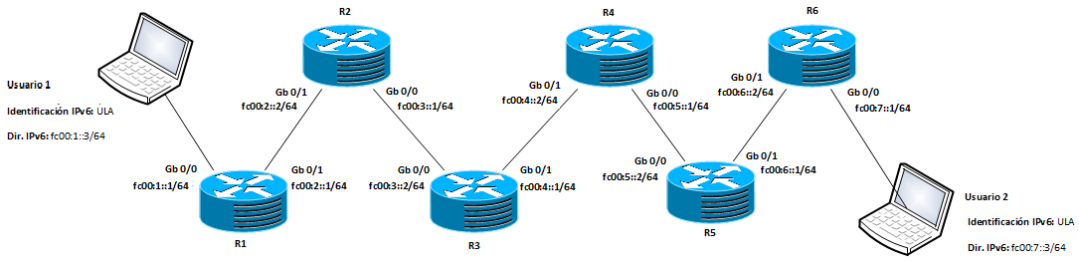
Se utilizaron routers Huawei de la serie AR2200 para los extremos y routers de la serie AR1220 para los internos. Este esquema maneja mascara de red de 8 bits, se desarrolló con enrutamiento dinámico mediante el protocolo OSPF para las

interfaces internas de los routers 1 al 6, por otro lado para las interfaces de borde de los routers de los extremos se cuenta con máscara de red de 24 bits en donde la interface Gb 0/0 del router 1 es utilizada para realizar 4 pruebas diferentes, 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una, las cuales llegaran hasta la interfaz Gb 0/0 del router 6 siendo monitoreadas por el software wireshark.

5.2.1.2 Experimento 2: OSPFv3 (IPv6)

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red sobre el protocolo OSPF en IPv6 utilizando direcciones tipo ULA, para medir jitter y delay.

Figura 25. Topología de Red con Direccionamiento IPv6



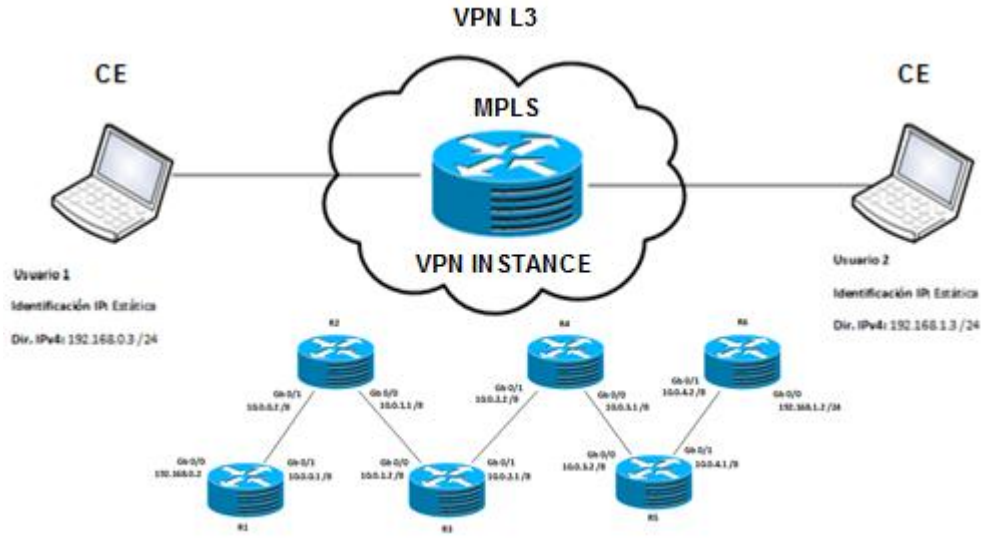
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron routers Huawei de la serie AR2200 para los extremos y routers de la serie AR1220 para los internos. El direccionamiento IPv6 que se implementó para OSPFv3 cuenta con direcciones Unicast de tipo ULA, este proceso se da por medio de IPv6 nativo en los routers de la serie AR220 Y AR1220. Para las interfaces Gb 0/0 de los routers 1 y 6 fueron utilizadas para realizar 4 pruebas diferentes, 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una, las cuales llegaran hasta la interfaz Gb 0/0 del router 6 siendo monitoreadas por el software wireshark.

5.2.1.3 Experimento 3: MPLS (IPv4)

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red sobre el protocolo MPLS en IPv4 realizando una VPN en capa 3 utilizando el direccionamiento del protocolo OSPF para medir jitter y delay.

Figura 26. Esquema Red MPLS (VPN) en IPv4



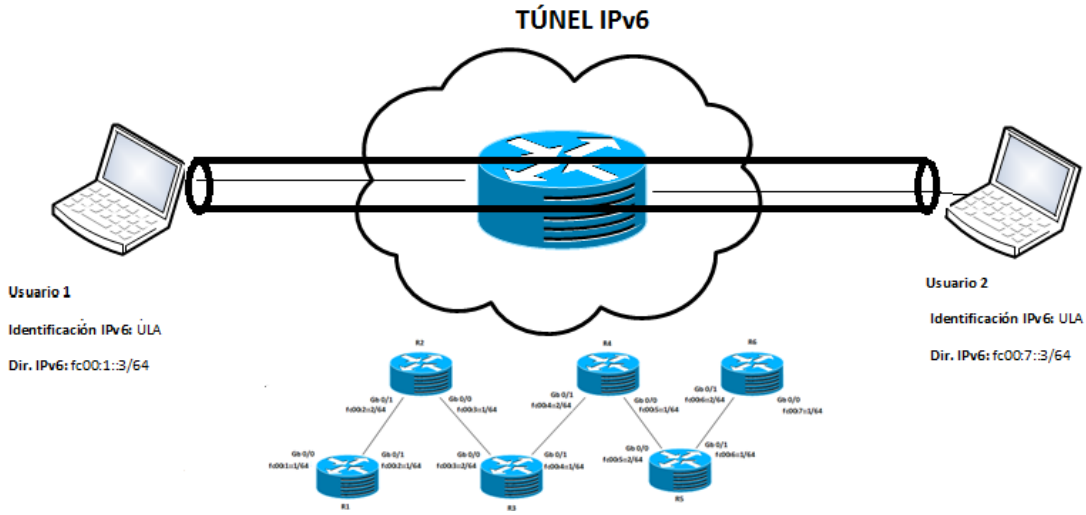
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron routers Huawei de la serie AR2200 para los extremos y routers de la serie AR1220 para los internos. Dentro de lo trabajado en MPLS se realizó una VPN en capa tres (Huawei, 2003), en donde se configura una VPN-instace en los equipos PE para así poder pasar trafico ICMP sobre MPLS entre los equipos CE y poder realizar las pruebas correspondientes a esta topología de red.}

5.2.1.4 Experimento 4: MPLS IPv6

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red sobre el protocolo MPLS en IPv6 (soportado sobre OSPF IPv6) realizando un túnel ya que los routers no soportan L3 VPN para IPv6 midiendo jitter y delay.

Figura 27. Esquema Red MPLS (VPN) en IPv6.



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron routers Huawei de la serie AR2200 para los extremos y routers de la serie AR1220 para los internos. Esta topología de red se elabora debido a que en los equipos físicos de Huawei no está activado IPv6 sobre MPLS por lo tanto se realiza un túnel IPv6 sobre IPv4 en el cual se comprueba el trafico ICMP sobre MPLS entre las interfaces de los extremos.

5.2.2 PRUEBAS SIMULADAS

Para el desarrollo se utilizó la misma topología de red con los protocolos OSPF y MPLS realizadas en GNs3, por otro lado SDN se implementó en una máquina virtual Linux Ubuntu 12.04, para dichas pruebas simuladas se utilizó un computador con las siguientes características: memoria RAM de 8GB, procesador Intel(R) Core (TM) i7-2670QM CPU@ 2.20 GHz y Sistema Operativo 64 Bits.

El direccionamiento en IPv4 e IPv6 que se realizó para las pruebas simuladas en el software GNs3 muestra a continuación:

Tabla 16. Direccionamiento IPv4 para los Routers simulados en el software GNs3.

Interfaz/Rx	R1	R2	R3	R4	R5	R6
Gb 1/0	192.168.0.1/24	10.0.0.2/8	10.0.1.2/8	10.0.2.2/8	10.0.3.2/8	10.0.4.2
Gb 2/0	10.0.0.1/8	10.0.1.1/8	10.0.2.1/8	10.0.3.1/8	10.0.4.1/8	192.168.1.2/24

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Tabla 17. Direccionamiento IPv6 para los Routers simulados en el software GNs3.

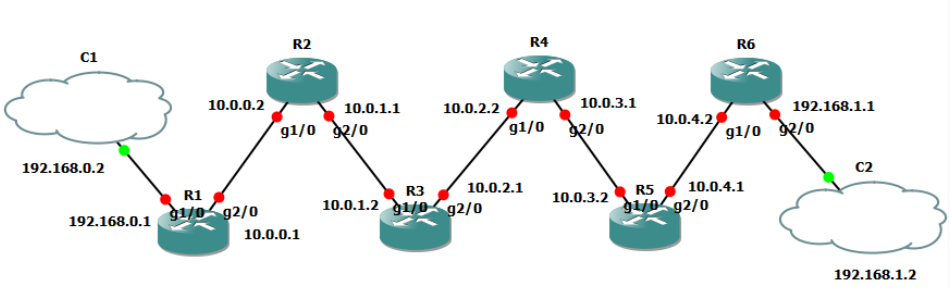
Interfaz/Rx	R1	R2	R3	R4	R5	R6
Gb 1/0	fc00:1::1/64	fc00:2::2/64	fc00:3::2/64	fc00:4::2/64	fc005::2/64	fc00:6::2/64
Gb 2/0	fc00:2::1/64	fc00:3::1/64	fc00:4::1/64	fc00:5::1/64	fc00:6::1/64	fc00:7::1/64

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

5.2.2.2 Experimento 5: OSPF IPv4 GNS3

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red simulado en el software GNs3 con el protocolo OSPF en IPv4 utilizando un direccionamiento de clase A y C, para medir jitter y delay.

Figura 28. Topología de Red OSPF en IPv4 (Simulado)



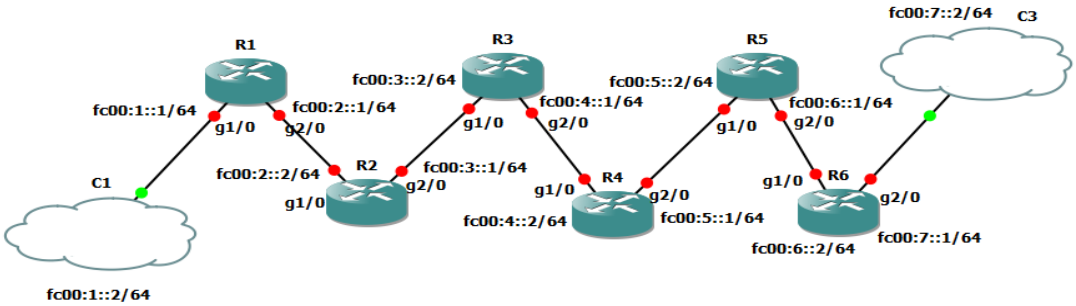
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron routers CISCO de la serie 7206VXR, Versión 12.4 (15) T14. Este esquema de red se desarrolla con la misma topología, para este diseño es necesario crear dos nubes independientes, cada una de ellas conectadas a dos máquinas virtuales de Linux realizando 4 pruebas de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una, siendo monitoreadas por el software wireshark.

5.2.2.3 Experimento 6: OSPFv3 (IPv6) GNS3

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red simulado en el software GNs3 con el protocolo OSPF en IPv6 para medir jitter y delay.

Figura 29. Topología de Red OSPF en IPv6 (Simulado)



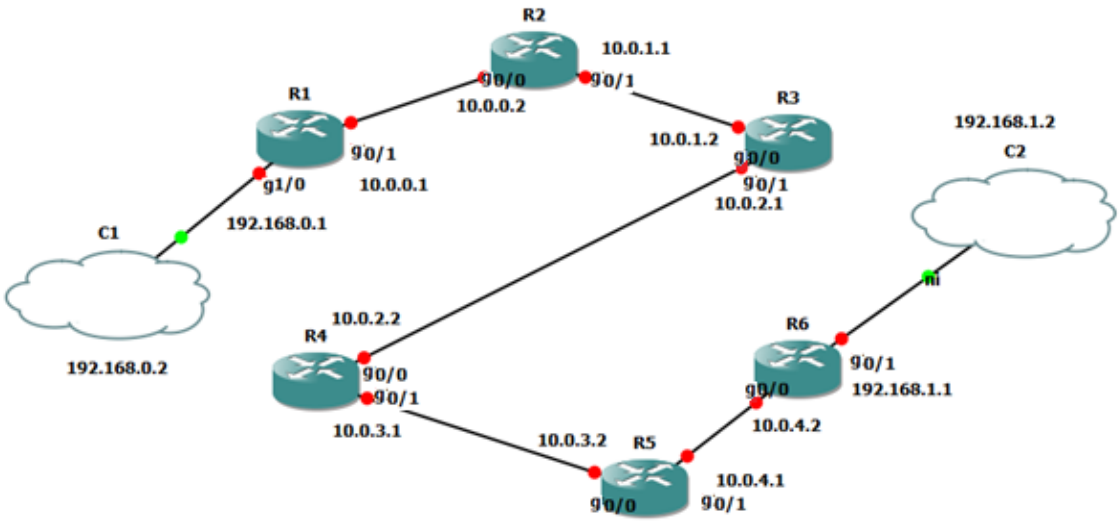
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron routers CISCO de la serie 7206VXR, Versión 12.4 (15) T14 y se desarrolla la misma red de OSPF IPv4 simulado con la misma topología de la física, creando dos nubes independientes, cada una de ellas conectadas a dos máquinas virtuales de Linux configurando la interfaz de red en IPv6 realizando 4 pruebas de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una siendo monitoreadas por el software wireshark.

5.2.2.4 Experimento 7: MPLS GNS3 IPv4

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red simulado en el software GNs3 con el protocolo MPLS en IPv4 utilizando el direccionamiento del protocolo OSPF para medir jitter y delay.

Figura 30. Esquema Red MPLS en IPv4 simulado.



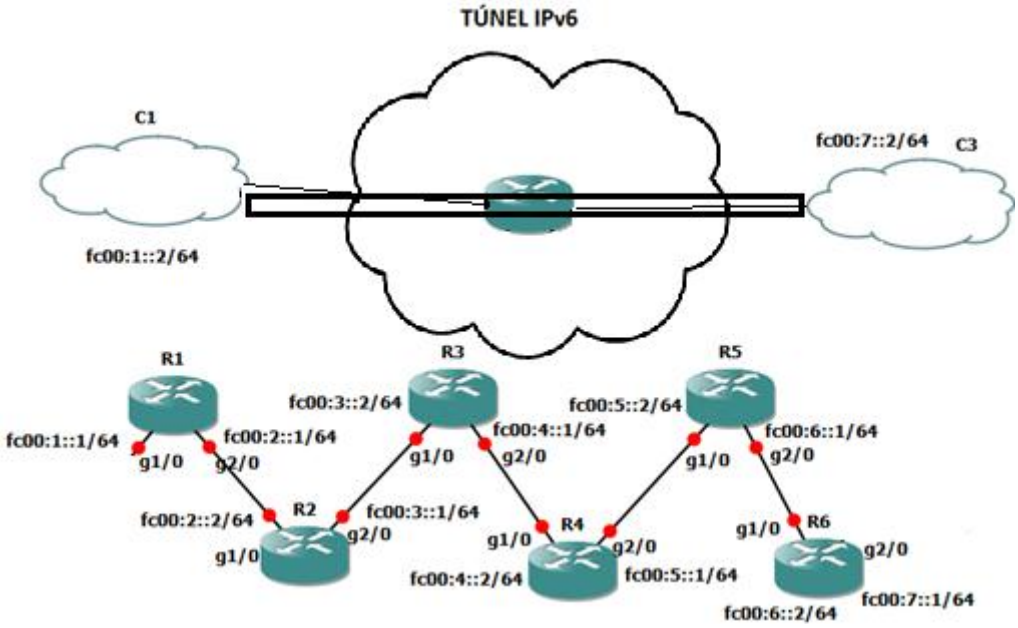
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron routers CISCO de la serie 7206VXR, Versión 12.4 (15) T14 creando la misma red con la misma topología de la física, creando dos nubes independientes, cada una de ellas conectadas a dos máquinas virtuales de Windows efectuando 4 pruebas de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una siendo monitoreadas por el software wireshark.

5.2.2.5 Experimento 8: MPLS GNS3 IPv6

El objetivo de este experimento es realizar cuatro pruebas mediante un entorno de red simulado en el software GNs3 con el protocolo MPLS en IPv6 (soportado sobre OSPF IPv6) realizando un túnel ya que los routers no soportan L3 VPN para IPv6 midiendo jitter y delay.

Figura 31. Esquema Red MPLS en IPv6 (Simulado).



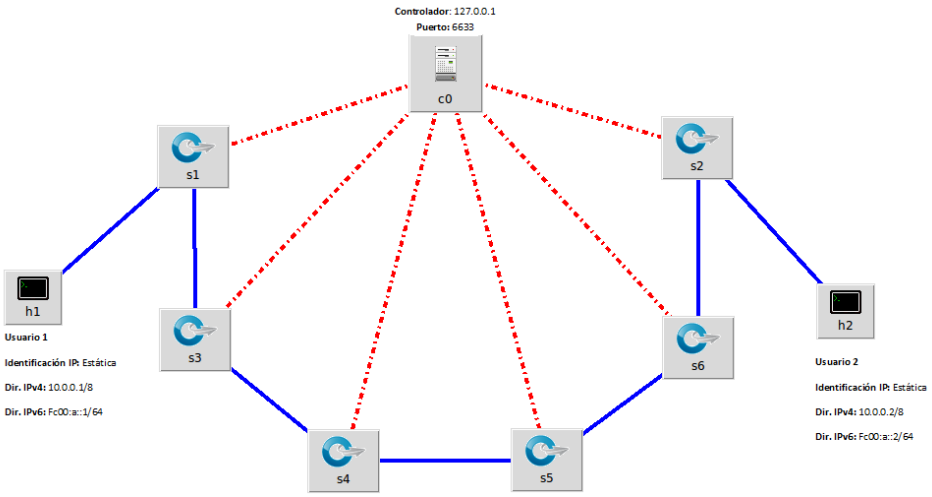
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se utilizaron CISCO de la serie 7206VXR, Versión 12.4 (15) T14, creando dos nubes independientes, cada una de ellas conectadas a dos máquinas virtuales de Windows, realizando un túnel IPv6 sobre IPv4 con las direcciones Link local de cada pc y así poder mantener el trafico ICMP sobre MPLS, también se hacen las pruebas de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una siendo monitoreadas por el software wireshark.

5.2.2.6 Experimento 9: SDN

El objetivo de este experimento es realizar ocho pruebas (cuatro de IPv4 y cuatro para IPv6), mediante una máquina virtual con sistema operativo Linux Ubuntu 12.04, además el entorno de red esta emulado con el ambiente grafico de Mininet desarrollado con el controlador y el protocolo OpenFlow y sus respectivos switches OpenFlow por lo tanto en el host 2 virtual se medirá jitter y delay.

Figura 32. Topología de Red en SDN



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se trabajó en el software Linux Ubuntu 12.04 desarrollando la misma arquitectura de red que se hiso para OSPF y MPLS, para este caso se tienen dos host virtuales con direcciones IP clase A para IPv4 y direcciones ULA tipo UNICAST para IPv6, para las interfaces de salida de los Switches de extremo se realizaron 4 pruebas diferentes de 10, 100, 6000 y 30000 Bytes con 200 paquetes tanto para IPv4 como para IPv6, tomando la captura del sniffer Wireshark en la interfaz de salida del Switch 2 (Host 2).

5.3 IMPLEMENTACIÓN Y DESARROLLO DE EXPERIMENTOS.

Dentro del desarrollo de esta investigación se realizan cuarenta pruebas las cuales estarán evidenciadas más adelante mostrando los resultados de cada una de ellas, además dichas pruebas se realizaron con el protocolo ICMP y se capturaron por medio del software Wireshark. Este proceso de pruebas se divide en dos fases:

Fase 1: Se realizan las pruebas con routers físicos en el entorno de red, configurando cada uno de los routers HUAWEI mediante los protocolos OSPF y MPLS, cada uno de ellos en IPv4 e IPv6.

Fase 2: Se realizan las pruebas simuladas teniendo en cuenta los mismos protocolos OSPF y MPLS estos trabajados en GNs3 simulando routers CISCO. Por otro lado SDN se desarrolla con el ambiente grafico de Mininet desarrollado con el controlador POX y el protocolo OpenFlow y sus respectivos switches OpenFlow. Para ambos casos se trabajara IPv4 e IPv6.

5.3.1 Configuración de Interfaces de Red para OSPF en Routers Huawei

Para la configuración de las interfaces de red de los routers HUAWEI se tendrá en cuenta la figura 24 y los siguientes pasos:

- Entrar en modo habilitado utilizando el comando *System-view*.
- Entrar a la interfaz que se quiere configurar mediante el comando *interface GigaEthernet 0/0/X.*
- Configurar las IP asignadas en la tabla 14 para cada interfaz de los routers mediante el comando *ip adress x.x.x.x 255.x.x.x*.
- Configurar el área en cada router, entrando al modo OSPF mediante el comando *ospf* seguidamente ejecutar *area x,* la red y la wildcard mediante *network X,X.X.X 0.0.X.X.*

5.3.1.1 Configuración Genérica para los Routers

- Configuración de dirección IP en interfaces:

```
[R1] interface GigabitEthernet0/0/0
[R1] ip address 192.168.0.2 255.255.255.0
[R1] interface GigabitEthernet0/0/1
[R1] ip address 10.0.0.1 255.255.255.0
```

- Configuración del Área en Router.

```
[R1] ospf 1
[R1-ospf-1] area 0.0.0.0
[R1-ospf-1] network 10.0.0.0 0.0.0.255
[R1-ospf-1] network 192.168.0.0 0.0.0.255
```

5.3.2 Configuración de Interfaces de Red para OSPFv3 en Huawei

Para la configuración de las interfaces de red en IPv6 se tendrá en cuenta la figura 25 y los siguientes pasos:

- Entrar en modo habilitado utilizando el comando *System-view*.
- Habilitar IPv6 mediante el comando *ipv6.*
- Habilitar ospfv3 con el comando *ospfv3* y mediante el comando router-id X.X.X.X habilitar este proceso, por ultimo especificar el área a la cual pertenecerá la interfaz.
- Entrar a la interfaz que se quiere configurar mediante el comando *interface GigaEthernet 0/0/X* y habilitar nuevamente IPv6 en cada interfaz.
- Configurar las IP asignadas en la tabla 15 para cada interfaz de los routers mediante el comando *ipv6 adress fc00:x::x 64.*
- Crear una interfaz de loopback con el comando *interface loopback 0*.
- A continuación se habilita IPv6 y se asigna una dirección IPv6 a la loopback en este caso ira de la 1111::1/128 hasta la 6666::6 /128 y finalmente se especifica el área creada anteriormente.

5.3.2.1 Configuración Genérica para los Routers

- Configuración del Área en Router.

```
[R1] ipv6
[R1] ospfv3 1
[R1- ospfv3 1] router-id 1.1.1.1
```

- Configuración de dirección IP en interfaces:

```
[R1] interface GigabitEthernet0/0/0
[R1- interface GigabitEthernet0/0/0] ipv6 enable
[R1- interface GigabitEthernet0/0/0] ipv6 address FC00:1::1/64
[R1- interface GigabitEthernet0/0/0] ospfv3 1 area 0.0.0.1
```

- Configuración de la interfaz de loopback0

[R1] interface LoopBack0

[R1-interface LoopBack0] ipv6 enable

[R1-interface LoopBack0] ipv6 address 1111::1/128

[R1-interface LoopBack0] ospfv3 1 area 0.0.0.1

5.3.3 Configuración de Interfaces de Red MPLS en Huawei

Para el desarrollo, se realizó una VPN en capa 3 en los routers de borde para poder enviar tráfico ICMP entre los PC´s y se tendrá en cuenta la figura 26. A continuación se mostrara los pasos y la configuración para routers internos y de borde.

5.3.3.1 Configuración Genérica de los Routers de PE.

Para la configuración de los routers de borde es necesario tener el protocolo OSPF en toda la red como base de MPLS y LDP operativo y una vez hecho esto realizar un VPN instance, realizando los siguientes pasos:

- Entrar en modo habilitado utilizando el comando *System-view*.
- Habilitar en cada interfaz MPLS con el comando *mpls* y *mpls ldp*.
- Habilitar ip vpn-instance *vpn-instance-name.*
- Por medio del protocolo BGP se realizara la configuración para el destino de la interfaz del otro extremo, esto con el siguiente comando *peer x.x.x.x as-number 100* y *con peer x.x.x.x connect-interface loopback0*.
- En seguida configurar el RD (Router distigusher) el cual se realizara con el comando *route-distinguisher*.
- Se habilitan los atributos de VPN instance mediante los comandos *vpn-target vpn-target &<1-0> export-extcommunity* / *import-extcommunity*.
- Por ultimo con el comando *import route-policy policy-name* y el *export route-policy policy-name*.

A continuación se presenta la configuración en detalle para los routers de borde:

- Configuración de VPN instance.

```
[R1] ip vpn-instance tesis
[R1] ipv4-family
[R1] route-distinguisher 100:101
[R1] vpn-target 100:101 export-extcommunity
[R1] vpn-target 100:101 import-extcommunity
```

- Configuración de MPLS en el área del router

```
[R1] router id 50.50.50.50
[R1] mpls lsr-id 50.50.50.50
[R1] mpls
[R1] mpls ldp
```

- Configuración de MPLS en las interface GigabitEthernet de borde.

```
[R1] interface GigabitEthernet0/0/2
[R1] ip binding vpn-instance tesis
[R1] ip address 192.168.0.1 255.255.255.0
```

- Configuración de interface Loopback0.

```
[R1] interface LoopBack0
[R1] ip address 50.50.50.50 255.255.255.255
```

- Configuración del protocolo BGP.

```
[R1] bgp 100
[R1] peer 60.60.60.60 as-number 100
[R1] peer 60.60.60.60 connect-interface LoopBack0
[R1] ipv4-family unicast
[R1] undo synchronization
[R1] undo peer 60.60.60.60 enable
[R1] ipv4-family vpn-instance tesis
[R1] import-route direct
```

5.3.3.2 Configuración Genérica para los Routers Internos

Para la configuración de los routers internos solo se habilita *mpls* y *mpls ldp* en el router y en cada una de las interfaces de red.

- Configuración de MPLS.

```
[R1] interface GigabitEthernet0/0/0
[R1] ip address 10.10.12.1 255.255.255.252
[R1] mpls
[R1] mpls ldp
[R1] interface GigabitEthernet0/0/1
[R1] ip address 10.10.16.1 255.255.255.252
[R1] mpls
[R1] mpls ldp
```

5.3.4 Configuración de Interfaces de Red MPLS IPv6 en Huawei

Para la configuración de MPLS IPv6 se desarrolló un Túnel IPv6 sobre IPv4 y se tendrá en cuenta la figura 27 configurando las interfaces de red de cada uno de los pc de la siguiente manera:

- Se entra al cmd y se ejecuta el comando *netsh.*
- Se habilita las interfaces de red en IPv6 con el siguiente comando *interface ipv6*.
- Ya en la interfaz con el siguiente comando *reset* se restablece la interfaz IPv6.
- Por último se ejecuta el comando con las direcciones de los host y se le da un nombre al túnel teniendo así el comando *add v6v4tunnel tesis 192.168.1.2 192.168.0.2* y se comprueba conexión por medio del ping.

5.3.4.1 Configuración Genérica para PC

- Configuración de túnel IPv6 sobre IPv4

```
C:\Windows\system32>netsh

        netsh>interface ipv6
```

netsh interface ipv6>reset

Interfaz se restableció correctamente.

Dirección de unidifusión se restableció correctamente.

Ruta se restableció correctamente.

Reinicie el equipo para completar esta acción.

netsh interface ipv6>add v6v4tunnel tesis 192.168.1.2 192.168.0.2

netsh interface ipv6>ping -6 fc00:1::3

5.3.5 Configuración de Interfaces de Red para OSPF GNS3 en Cisco

Para la configuración de las interfaces de red de los routers CISCO en GNs3 se tendrá en cuenta la figura 28 y los siguientes pasos:

- Entrar en modo habilitado utilizando el comando *configure Terminal*.
- Entrar a la interfaz que se quiere configurar mediante el comando *interface GigaEthernet 0/0/X.*
- Configurar las IP asignadas para cada interfaz de los routers mediante el comando *ip adress x.x.x.x 255.x.x.x*.

- Configurar la loopback0 mediante el comando *interface Loopback0* y se asigna la dirección IP que tendrá *ip address 1.1.1.1 255.255.255.0*
- Configurar el área en cada router, entrando con *ip routing* seguidamente *router ospf 1* y por último la red con la wildcard por medio del comando *network X,X.X.X 0.0.X.X.area 0.*

5.3.5.1 Configuración Genérica para los Routers

- Configuración de dirección IP en interfaces:

[R1-config]# interface GigabitEthernet1/0

[R1-config]# ip address 192.168.0.1 255.255.255.0

[R1-config]# interface GigabitEthernet2/0

[R1-config]# ip address 10.0.0.1 255.255.255.0

- Configuración de Interface de Loopback 0

  [R1-config]# Interface Loopback0

  [R1-config]# ip address 1.1.1.1 255.255.255.0

- Configuración de OSPF.

  [R1-config]# router ospf 1

  [R1-config]# network 10.0.0.0 0.0.0.255 area 0

  [R1-config]# network 192.168.0.0 0.0.0.255 area 0

5.3.6 Configuración de Interfaces de Red OSPF Ipv6 GNS3 en Cisco

Para la configuración de OSPF IPv6 en las interfaces de red de los routers CISCO en GNs3 se tendrá en cuenta la figura 29 y los siguientes pasos:

- Entrar en modo habilitado utilizando el comando *Configure Terminal*.
- Habilitar IPv6 mediante el comando *ipv6 unicast-routing,*
- Habilitar ospf con el comando *ipv6 router ospf NAME.* y mediante el comando *router-id xx_NAME.*
- Entrar a la interfaz que se quiere configurar mediante el comando *interface GigaEthernet 0/0/X*.
- Configurar las IP asignadas para cada interfaz de los routers mediante el comando *ipv6 adress fc00:x::x 64* y a cada interfaz asignar cada área mediante *ipv6 ospf 1 area x.*
- Crear una interfaz de loopback con el comando *interface loopback 0* y asiganarle una Ip mediante *ip address* x.x.x.x x.x.x.0.

5.3.6.1 Configuración Genérica para los Routers

- Configuración de IPV6.

  [R1-config]# ipv6 unicast-routing

  [R1-config]# ipv6 router ospf 1

  [R1-config]# router-id 1.1.1.1

- Configuración de dirección IPv6 en interfaces y IPv6 OSPF:

[R1-config]# interface GigabitEthernet1/0

[R1-config]# ipv6 address FC00:1::1/64

[R1-config]# ipv6 ospf 1 area 0

[R1-config]# interface GigabitEthernet2/0

[R1-config]# ipv6 address FC00:2::1/64

[R1-config]# ipv6 ospf 1 area 0

- Configuración de Interface de Loopback 0

[R1-config]# Interface Loopback0

[R1-config]# ip address 1.1.1.1 255.255.255.0

5.3.7 Configuración de Interfaces de Red para MPLS GNS3 en Cisco

Para la configuración de MPLS en CISCO es necesario tener ya configurado los protocolos de routing y tener en cuenta la figura 30. A continuación se mostrara los pasos y la configuración para routers internos y de borde.

5.3.7.1 Routers de Borde

Para la configuración de MPLS en los routers CISCO en GNs3 se tendrá en cuenta los siguientes pasos:

- Entrar en modo habilitado utilizando el comando *Configure Terminal*.
- Habilitar el CEF mediante el comando *ip cef* y se comprueba con *show ip cef summary.*
- Entrar a la interfaz que se quiere configurar mediante el comando *interface GigaEthernet 0/0/X*.
- Configurar las IP asignadas para cada interfaz de los routers mediante el comando *ip adress x.x.x.x x.x.x.x.*
- Activar el protocolo LDP solo para la interfaces internas que manejaran MPLS con *mpls lp* y *mpls label protocolo ldp.*
- Para verificar con los comandos *show mpls interfaces, show mpls ldp parameters* y *show mpls ldp neighbor.*

A continuación se presenta en detalle la configuración:

- Configuración de Interfaz de borde del router 1.

```
[R1-config]# ip cef
[R1-config]# mpls label protocol ldp
[R1-config]# interface GigabitEthernet0/0
[R1-config]# ip address 192.168.0.1 255.255.255.0
```

- Configuración de Interfaz Interna del Router 1 de Borde.

```
[R1-config]# interface GigabitEthernet 0/1
[R1-config]# ip address 10.0.0.1 255.255.255.0
[R1-config]# mpls label protocol ldp
```

5.3.7.2 Routers Internos

Para la configuración de los routers internos se habilita *mpls* y *mpls ldp* en el router y en cada una de las interfaces de red.

- Entrar en modo habilitado utilizando el comando *Configure Terminal*.
- Habilitar el CEF mediante el comando *ip cef* y se comprueba con *show ip cef summary.*
- Entrar a la interfaz que se quiere configurar mediante el comando *interface GigaEthernet 0/0/X*.
- Configurar las IP asignadas para cada interfaz de los routers mediante el comando *ip adress x.x.x.x x.x.x.x.*
- Activar el protocolo LDP en todas las interfaces que se van a utilizar con *mpls lp* y *mpls label protocolo ldp.*
- Para verificar con los comandos *show mpls interfaces, show mpls ldp parameters* y *show mpls ldp neighbor.*

A continuación se presenta en detalle la configuración:

- Configuración de las Interfaces internas del router 2.

```
[R1-config]# ip cef
[R1-config]# mpls label protocol ldp
[R1-config]# interface GigabitEthernet0/0
[R1-config]# ip address 10.0.0.2 255.255.255.0
[R1-config]# mpls label protocol ldp
[R1-config]# mpls ip
 [R1-config]# interface GigabitEthernet 0/1
[R1-config]# ip address 10.0.1.1 255.255.255.0
[R1-config]# mpls label protocol ldp
[R1-config]# mpls ip
```

5.3.8 Configuración de Interfaces de Red para MPLS IPv6 GNS3 en Cisco

Para la configuración de MPLS IPv6 se desarrolló un Túnel IPv6 sobre IPv4 configurando las interfaces de red de cada uno de los pc de la siguiente manera:

- Se entra al cmd y se ejecuta el comando *netsh.*
- Se habilita las interfaces de red en IPv6 con el siguiente comando *interface ipv6*.
- Ya en la interfaz con el siguiente comando *reset* se restablece la interfaz IPv6.
- Por último se ejecuta el comando con las direcciones de los host y se le da un nombre al túnel teniendo así el comando *add v6v4tunnel tesis 192.168.1.2 192.168.0.2* y se comprueba la conexión por medio del ping.

5.3.8.1 Configuración Genérica para PC

- Configuración de Túnel IPv6 sobre IPv4

```
C:\Windows\system32>netsh
    netsh>interface ipv6
    netsh interface ipv6>reset
```

Interfaz se restableció correctamente.

Dirección de unidifusión se restableció correctamente.

Ruta se restableció correctamente.

Reinicie el equipo para completar esta acción.

netsh interface ipv6>add v6v4tunnel tesis 192.168.1.2 192.168.0.2

netsh interface ipv6>ping -6 fc00:1::3

5.3.9 Configuración de Simulación de SDN (Software Defined Networkinging)

Para la configuración de SDN se instala en una máquina virtual sistema operativo Linux Ubuntu 14.04 de 32 bits con OpenVswitch 2.1.0 con soporte OpenFlow versión 1.3 y Mininet, controladores SDN como OpenDaylight, POX, RYU.

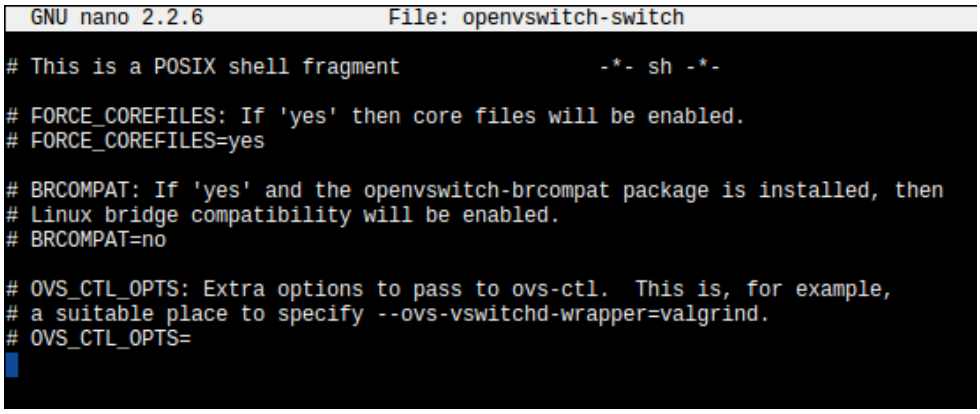
Para que el sistema maneje Open Vswitch con OpenFlow se descargan los paquetes entrando como súper usuario de la siguiente manera:

```
Sudo su
apt-get purge network-manager
apt-get install openvswitch-datapath-source bridge-utils
module-assistant auto-install openvswitch-datapath
apt-get install openvswitch-brcompat openvswitch-common
```

Se habilita la compatibilidad puente para cambiar el BRCOMPAT (SDN Hub, 2014) de no a yes por medio del comando nano.

```
/etc/default/openvswitch-switch
```

Figura 33. Configuración de BRCOMPAT para OpenvSwitch



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se reinicia el OpenVswitch con el comando:

/etc/init.d/openvswitch-switch restart

Se añade un puente y después una interfaz física realizando un puente virtual de la siguiente manera:

ovs-vsctl add-br br-int

ovs-vsctl add-port br-int eth0

ifconfig eth0 0

A la interfaz eth0 se le configura un puente de la siguiente manera

ifconfig br-int 192.168.1.208 netmask 255.255.255.0

Y se cambiara la ruta que viene por defecto

route add default gw 192.168.1.1 br-int and route del default gw 192.168.1.1 eth0

Ahora bien para los controladores SDN se instalara el POX de la siguiente forma:

sudo apt-get install git

git clone http://github.com/noxrepo/pox

```
cd pox

./pox.py forwarding.l2_learning

NOX&amp;gt;
```

Se puede observar que por defecto se une al puerto 6633 y los demás.

Por último se tendrá que adjuntar el OpenVswitch al controlador por medio del comando

```
ovs-vsctl set-controller br-int tcp:192.168.1.208:6633
```

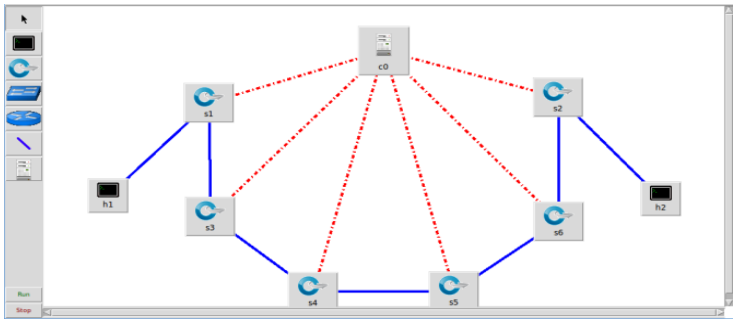
Realizado estos pasos ya se tendrá la maquina funcionando con OpenVswitch y con el controlador POX.

Una vez realizado estos pasos se inicializa mininet por medio de la interfaz gráfica miniedit con el comando:

```
Sudo sumininet/examples/miniedit.py
```

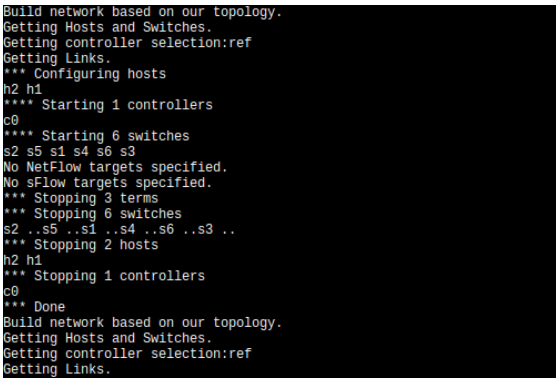
Y se observara la interfaz gráfica y como se crea la topología en la terminal.

Figura 34. Configuración de Topología de Red en SDN, Switches OpenVswitch



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Figura 35. Construcción de Topología de MiniNet

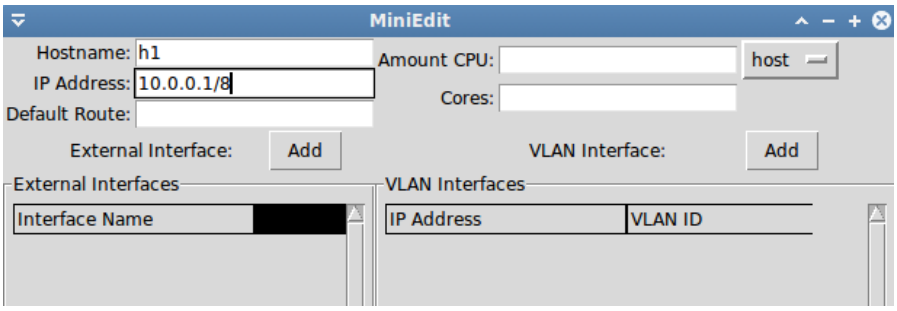


Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se procede a configurar la red de la siguiente manera:

Con click derecho mantenido en el host 1, se entra a propiedades y se configura las IP en IPv4 para cada uno de los Host virtuales creados.

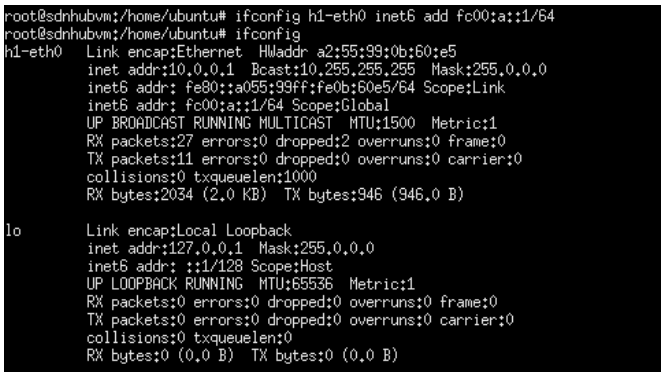
Figura 36. Propiedades del host 1 con dirección IPv4



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

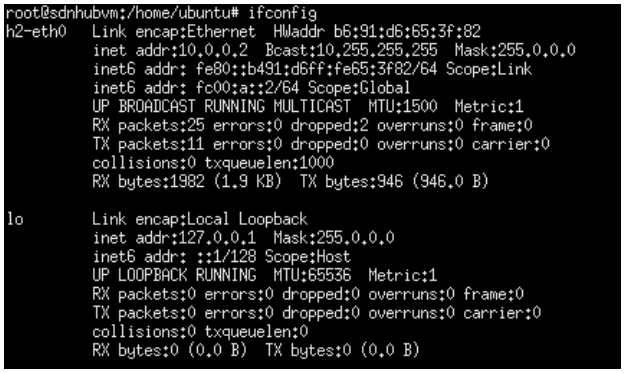
Para el desarrollo de IPv6 se ejecuta por comando ya que la versión del Mininet no está implementado en modo grafico por lo tanto lo se realizara de la siguiente manera con el comando `ifconfig h1-eth0 inet6 add fc00:a::1/64` tanto para host 1 como para host2.

Figura 37. Configuración de host 1 por comando con dirección IPv6



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

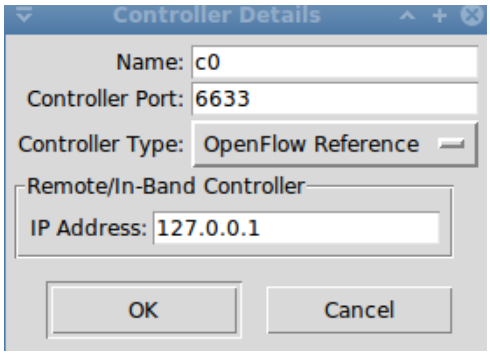
Figura 38. Configuración del hots 2 con direccionamiento IPv6



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Por otro lado se configura el controlador con el puerto por defecto 6633, y el tipo de controlador en este caso OpenFlow Reference y la IP de loopback 127.0.0.1.

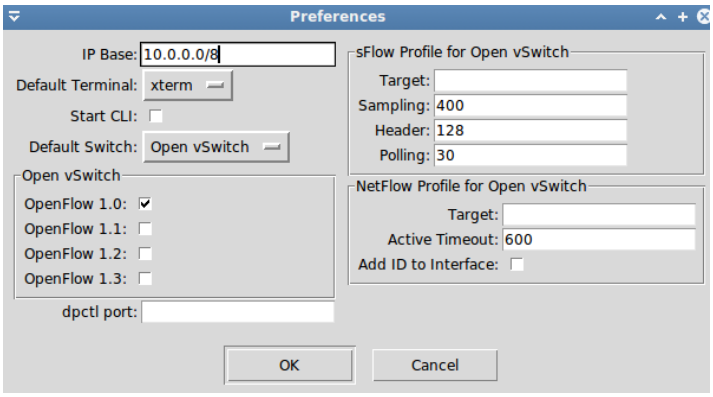
Figura 39. Configuración del Protocolo OpenFlow



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Por último en editar se configura las preferencias de la siguiente manera:

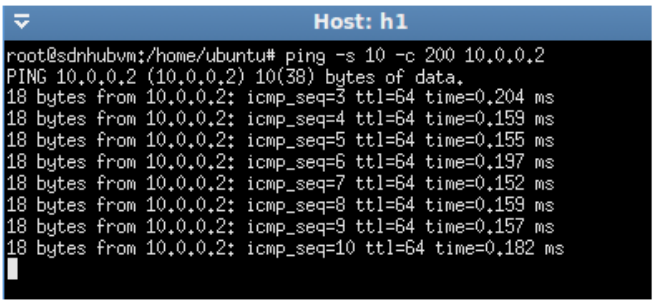
Figura 40. Configuración de las Preferencias de la Topología



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Para finalizar se procede a realizar las pruebas de 10, 100, 6000 y 30000 Bytes con 200 paquetes cada una.

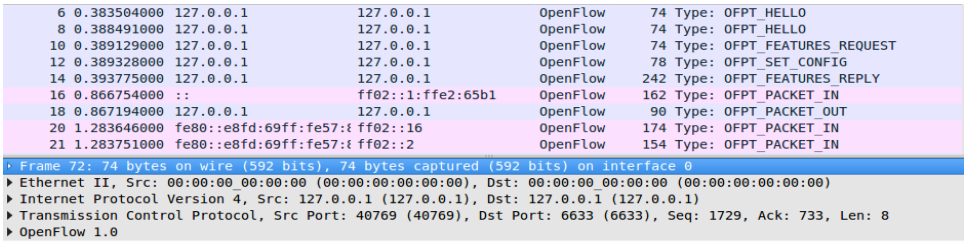
Figura 41. Ping del host 1 al host 2 (10 Bytes 200 paquetes)



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

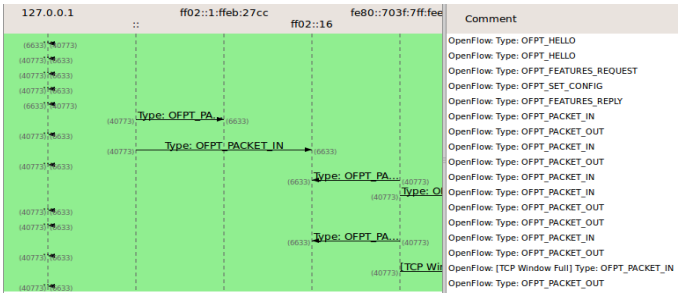
Ya realizados estos últimos pasos se observa el protocolo OpenFlow y sus mensajes de respuesta.

Figura 42. Captura del Protocolo OpenFlow



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Figura 43. Estructura de Mensajes OpenFlow



Fuente: Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera. Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.

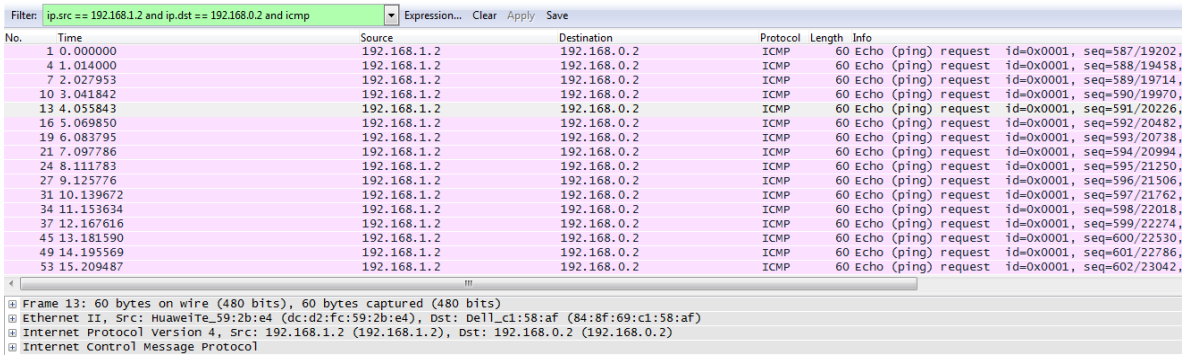
6. RESULTADOS DE EXPERIMENTOS

6.1 OBTENCIÓN DE DATOS DE DELAY Y JITTER A PARTIR DE LAS CAPTURAS

Para la obtención de los datos de cada una de las pruebas se realizó por medio del software Wireshark (Orzach, December 2013) importando los datos del protocolo ICMP acorde a cada topología de red. A continuación se mostrara el desarrollo genérico para cualquier prueba.

Con la captura de cada una de las pruebas, se realiza un filtro especificando la ip de origen, la ip de destino y el protocolo que en este caso será ICMP.

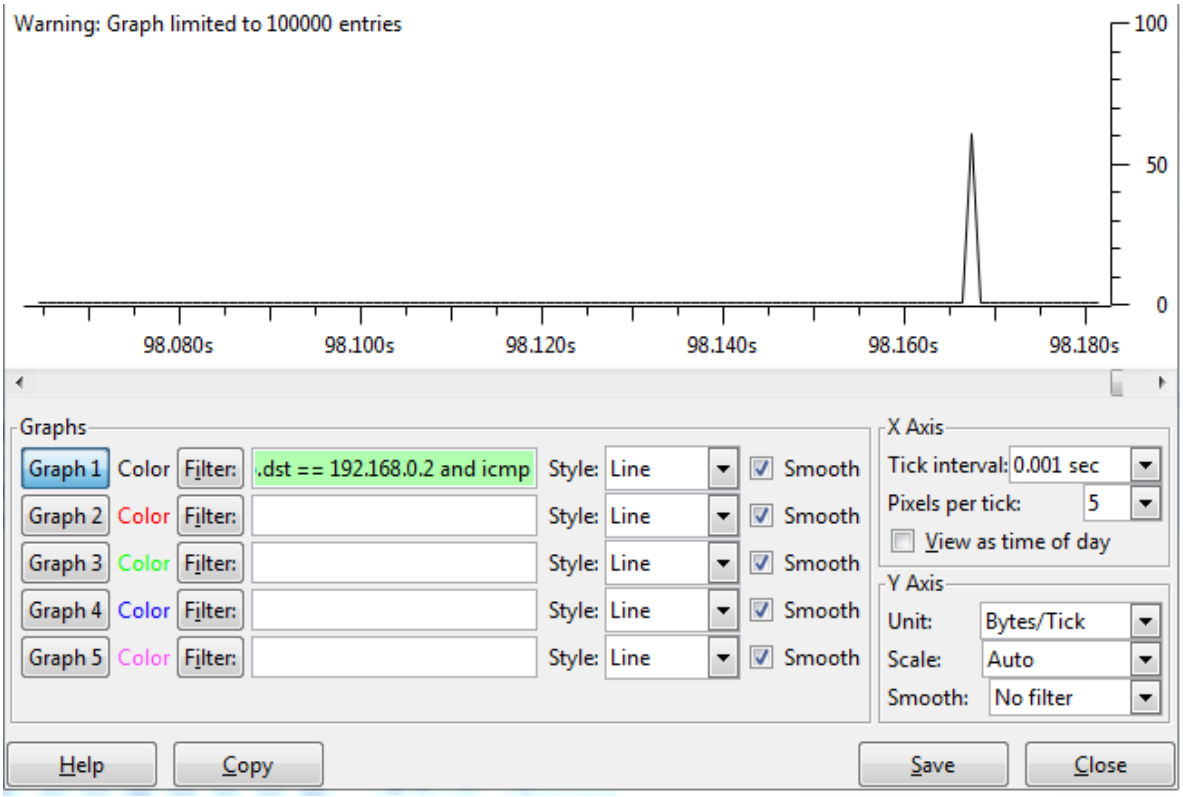
Figura 44. Filtro utilizado para las pruebas de IPv4 en el software Wireshark.



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Aplicado el filtro en las opciones de “Statistics – IO graph” se encontrara una gráfica que muestra los picos de la prueba, también hay que tener en cuenta que para opción se debe realizar el mismo filtro utilizado en la captura.

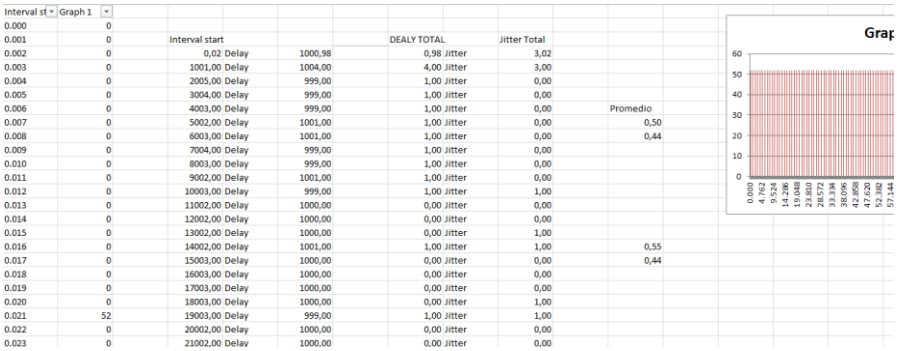
Figura 45. Grafica de la prueba de IPv4 en el software Wireshark.



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Para terminar se debe importar estos datos a Excel o alguna herramienta de software matemático, para el desarrollo de este proceso se tienen todos los valores y se debe filtrar solo los valores en los cuales se encuentran bytes, para posteriormente obtener el delay y el jitter.

Figura 46. Datos importados en Excel para la prueba de IPv4 del software Wireshark.



Interval st	Graph 1								
0.000	0								
0.001	0	Interval start			DEALY TOTAL	Jitter Total			
0.002	0	0,02	Delay	1000,98	0,98	Jitter	3,02		
0.003	0	1001,00	Delay	1004,00	4,00	Jitter	3,00		
0.004	0	2005,00	Delay	999,00	1,00	Jitter	0,00		
0.005	0	3004,00	Delay	999,00	1,00	Jitter	0,00		
0.006	0	4003,00	Delay	999,00	1,00	Jitter	0,00	Promedio	
0.007	0	5002,00	Delay	1001,00	1,00	Jitter	0,00	0,50	
0.008	0	6003,00	Delay	1001,00	1,00	Jitter	0,00	0,44	
0.009	0	7004,00	Delay	999,00	1,00	Jitter	0,00		
0.010	0	8003,00	Delay	999,00	1,00	Jitter	0,00		
0.011	0	9002,00	Delay	1001,00	1,00	Jitter	0,00		
0.012	0	10003,00	Delay	999,00	1,00	Jitter	1,00		
0.013	0	11002,00	Delay	1000,00	0,00	Jitter	0,00		
0.014	0	12002,00	Delay	1000,00	0,00	Jitter	0,00		
0.015	0	13002,00	Delay	1000,00	0,00	Jitter	1,00		
0.016	0	14002,00	Delay	1001,00	1,00	Jitter	1,00	0,55	
0.017	0	15003,00	Delay	1000,00	0,00	Jitter	0,00	0,44	
0.018	0	16003,00	Delay	1000,00	0,00	Jitter	0,00		
0.019	0	17003,00	Delay	1000,00	0,00	Jitter	0,00		
0.020	0	18003,00	Delay	1000,00	0,00	Jitter	1,00		
0.021	52	19003,00	Delay	999,00	1,00	Jitter	1,00		
0.022	0	20002,00	Delay	1000,00	0,00	Jitter	0,00		
0.023	0	21002,00	Delay	1000,00	0,00	Jitter	0,00		

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

En los siguientes datos obtenidos de los experimentos se resumen las tablas que mostraron el comportamiento de delay y el jitter con 10, 100, 6000 y 30000 Bytes cada una con 200 paquetes.

6.2 RESULTADOS DE PRUEBAS SIMULADAS

Las siguientes tablas muestran el comportamiento de las pruebas simuladas en IPv4 para SDN, OSPF y MPLS teniendo el resultado de Delay y el jitter promedio para cuatro tamaños de paquetes diferentes.

Tabla 18. Pruebas de Simulación en IPv4.

RED\ VARIABLE	**PRUEBA SIMULADA**				
	IPv4				
	Tamaño (Bytes)				
		10	**100**	**6000**	**30000**
SDN	**Delay (ms)**	0,55	0,76	2,04	2,64
	Jitter (ms)	0,44	0,6	1,78	1,74
OSPF	**Delay (ms)**	12,38	10,68	11,53	9,23
	Jitter (ms)	12,94	11,32	10,43	7,75
MPLS	**Delay (ms)**	4,10	3,59	7,18	12,55
	Jitter (ms)	4,98	3,49	12,08	10,39

Fuente: Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

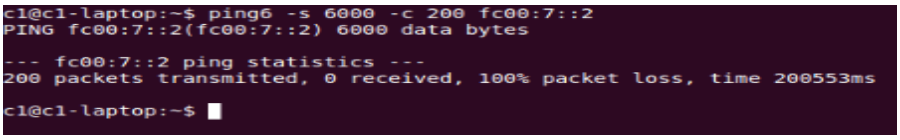
La siguiente tabla muestra el desarrollo de la prueba simulada en IPv6 para SDN, OSPF y MPLS teniendo el resultado de Delay y Jitter para cuatro valores diferentes. Para las pruebas de OSPF con 6000 y 30000 Bytes no fueron exitosas dado que la capacidad de procesamiento del computador no fue suficiente.

Tabla 19. Prueba de Simulación en IPv6.

RED\ VARIABLE	PRUEBA SIMULADA				
	IPv6				
	Tamaño (Bytes)				
		10	100	6000	30000
SDN	Delay (ms)	2,55	1,9	3,58	7,18
	Jitter (ms)	2,43	0,85	3,4	5,35
OSPF	Delay (ms)	485,75	20,57	X	X
	Jitter (ms)	970,36	36,86	X	X
MPLS	Delay (ms)	4,08	3,99	6,23	16,2
	Jitter (ms)	5,05	4	5,41	14,21

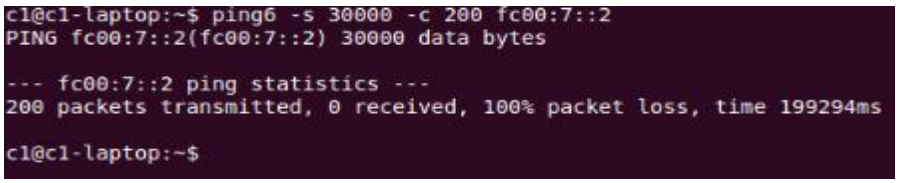
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Figura 47. Prueba OSPF 6000 Bytes en IPv6 Simulado



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Figura 48. Prueba OSPF 30000 Bytes en IPv6 Simulado



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

6.3 RESULTADOS DE PRUEBAS CON ROUTERS HUAWEI

Las siguientes tablas muestra el desarrollo de las pruebas físicas en IPv4 e IPv6 para OSPF y MPLS teniendo el resultado de Delay y Jitter para cuatro valores diferentes.

Tabla 20. Prueba con Routers Huawei en IPv4.

RED\ VARIABLE	PRUEBA CON ENRUTADORES HUAWEI				
	IPv4				
	Tamaño (Bytes)				
		10	100	6000	30000
OSPF	Delay (ms)	2,53	2,78	7,39	2,79
	Jitter (ms)	0,48	0,57	5,69	0,76
MPLS	Delay (ms)	12,74	13,65	4,93	5,12
	Jitter (ms)	2,62	5,45	4,28	4,44

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Tabla 21. Prueba con Routers Huawei en IPv6.

RED\ VARIABLE	PRUEBA CON ENRUTADORES HUAWEI				
	IPv6				
	Tamaño (Bytes)				
		10	100	6000	30000
OSPF	Delay (ms)	1,87	2,07	2,55	3,73
	Jitter (ms)	0,93	0,56	0,77	2,42
MPLS	Delay (ms)	135,19	89,36	44,86	44,28
	Jitter (ms)	267,45	176,73	87,89	87,04

Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

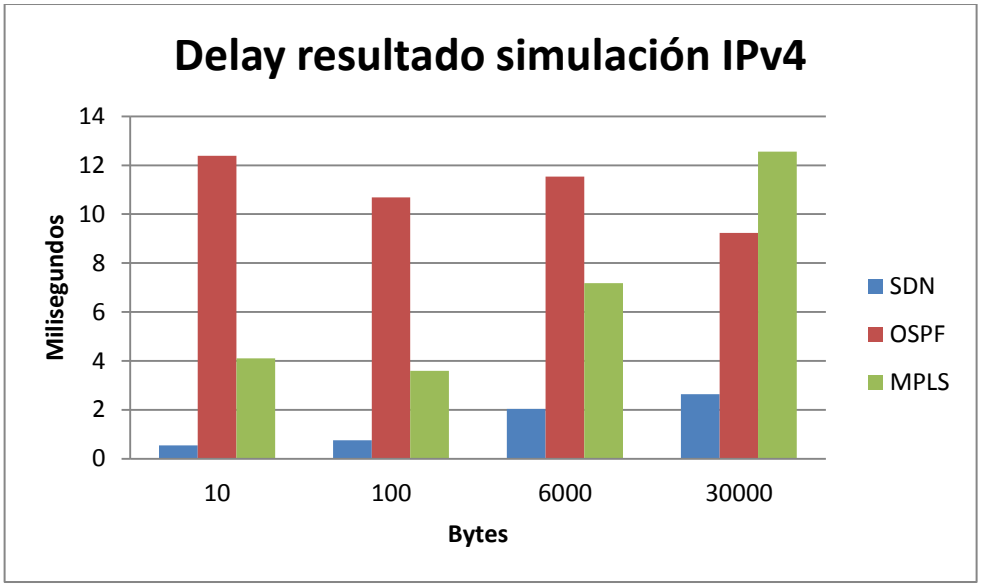
7. ANÁLISIS DE RESULTADOS

Los análisis a continuación muestran el rendimiento de cada uno de los entornos de red trabajados por medio de OSPF y MPLS con routers Huawei y routers CISCO simulado, también el rendimiento de SDN emulado encontrando diferencias entre cada una de las tecnologías mostradas en este proyecto.

7.1 PRUEBAS DELAY SIMULADAS EN IPv4

Para las pruebas de delay realizadas en IPV4 para SDN, OSPF y MPLS se obtuvieron los siguientes resultados:

Figura 49. Delay en SDN, OSPF y MPLS en IPv4



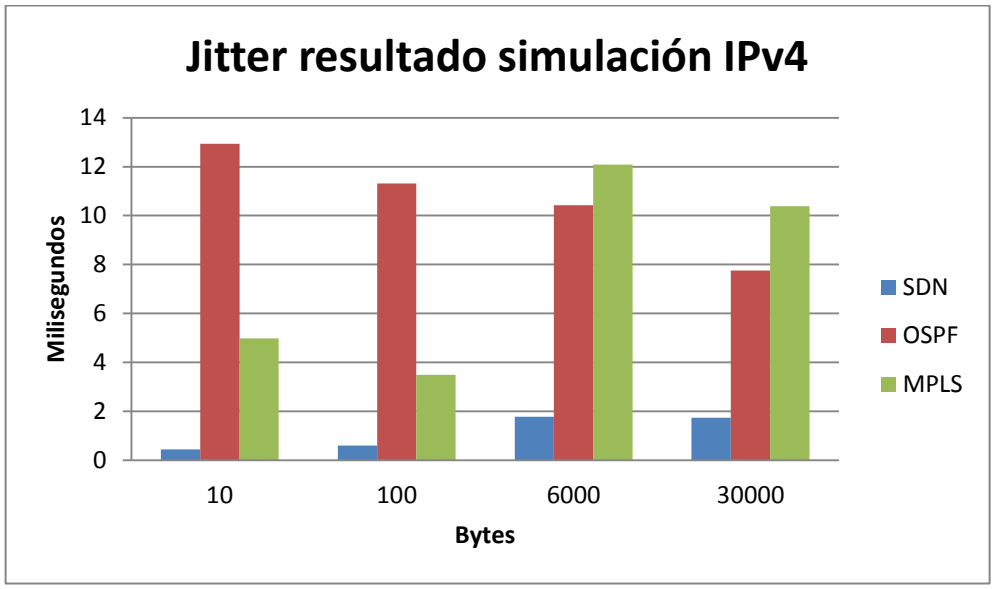
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Las pruebas correspondientes muestran que el delay es bajo para SDN con respecto a OSPF y MPLS en CISCO de 10, 100, 6000 y 30000 Bytes cada una, observando que para OSPF se mantiene el delay entre 13 y 9 ms, tanto que para MPLS va ascendiendo a medida que aumentan los paquetes.

7.2 PRUEBAS DE JITTER SIMULADAS EN IPv4

Para las pruebas de Jitter en IPV4 para SDN, OSPF y MPLS se obtuvieron los siguientes resultados:

Figura 50. Jitter de SDN, OSPF y MPLS en IPv4



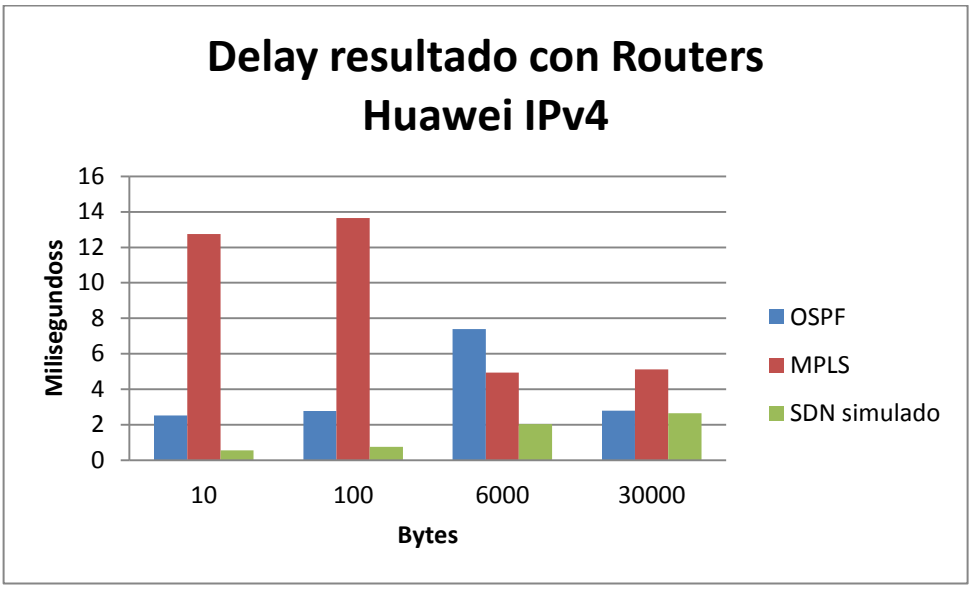
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Las diferencias de estas tecnologías muestran que SDN tiene un jitter menor con respecto OSPF y MPLS, manteniendo las mismas proporciones de Delay entre 13 y 9 ms.

7.3 PRUEBAS DELAY CON LOS ROUTERS HUAWEI EN IPv4

Las pruebas con los routers Huawei en IPV4 se realizaron para OSPF y MPLS obteniendo los siguientes resultados:

Figura 51. Delay de OSPF y MPLS en IPv4 (Routers Huawei)



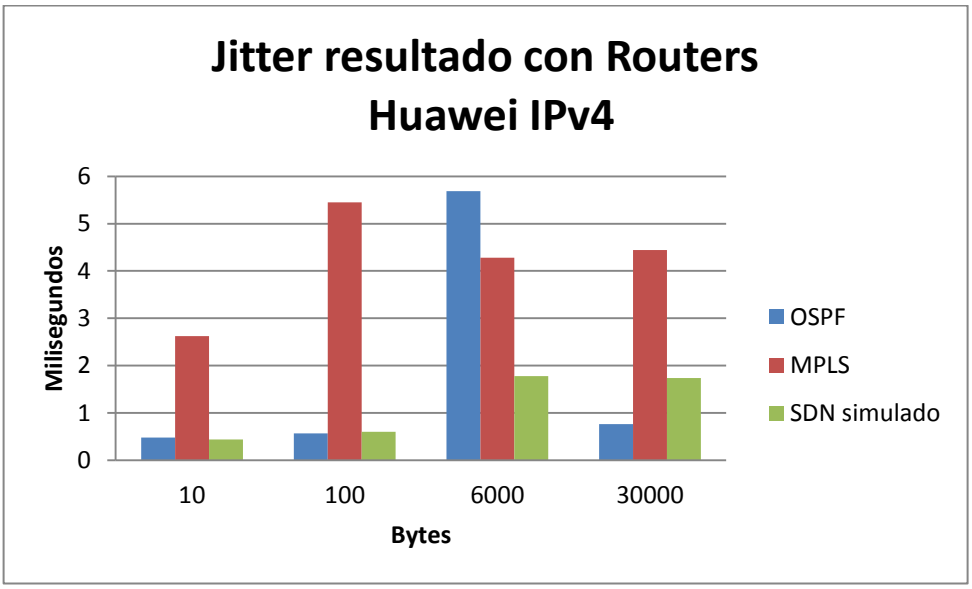
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Observando estas tecnologías en los routers Huawei se puede ver que el Delay se comporta similar al simulado en cuanto a los valores manejados en milisegundos, por otro lado varía para las pruebas de OSPF las cuales son más bajas. Para MPLS las pruebas de 10 y 100 Bytes aumentaron, tanto que para las de 6000 y 30000 Bytes bajaron comportándose más rápido para valores grandes.

7.4 PRUEBAS DE JITTER CON ROUTERS HUAWEI EN IPv4

Las pruebas con los routers Huawei en IPV4 se realizaron para OSPF y MPLS obteniendo los siguientes resultados:

Figura 52. Jitter de OSPF y MPLS en IPv4 (Routers Huawei)



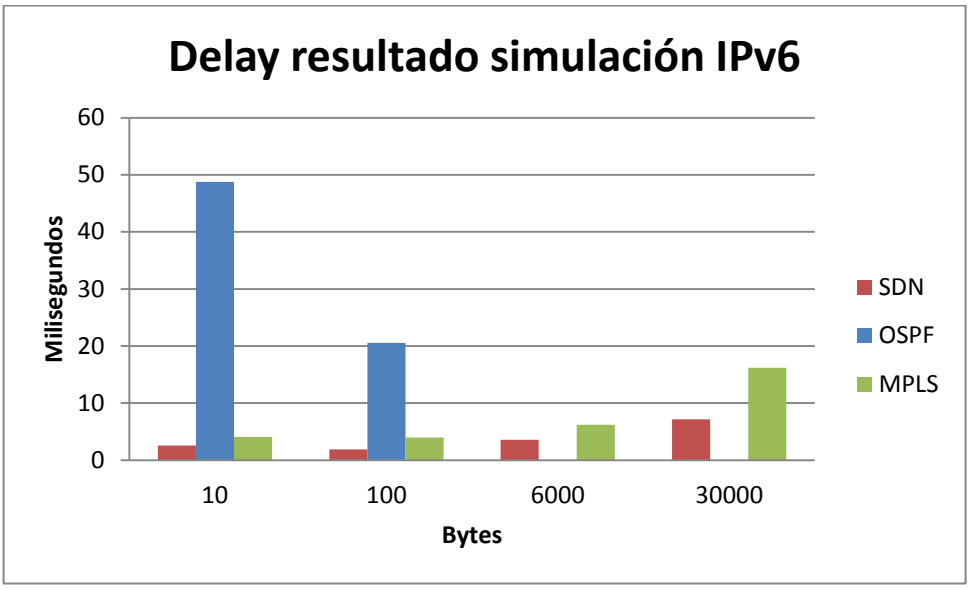
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

El Jitter en los routers Huawei es menor que simulado, encontrando unos picos de 4 a 6 ms en las pruebas de 100 Bytes para MPLS y de 6000 Bytes para OSPF y MPLS. También se observa que SDN sigue teniendo un Jitter bajo para tramas más pequeñas.

7.5 PRUEBAS DE DELAY SIMULADAS EN IPv6

Para las pruebas de Delay realizadas en IPV6 para SDN, OSPF y MPLS se obtuvieron los siguientes resultados:

Figura 53. . Delay de SDN, OSPF y MPLS en IPv6



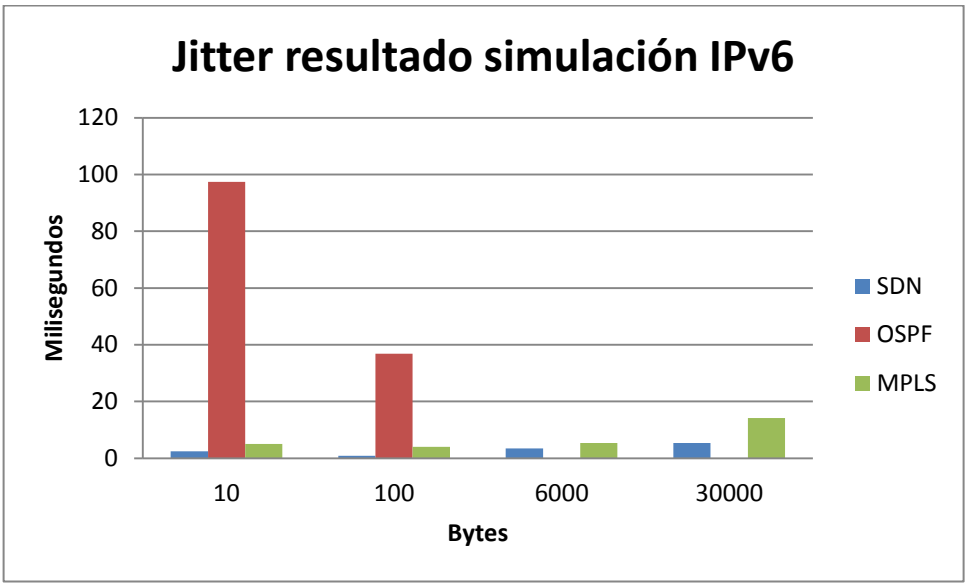
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Con respecto al Delay en OSPF IPv6 simulado, se obtuvieron resultados elevados para 10 y 100 Bytes, cabe destacar que para las demás pruebas no se realizaron ya que había pérdidas de paquetes evidenciándolas en las figuras 48 y 49, por último la diferencia entre MPLS y SDN es poca oscilando entre 2 y 3 ms, excepto para la prueba de 30000 Bytes.

7.6 PRUEBAS DE JITTER SIMULADAS EN IPv6

Para las pruebas de Jitter realizadas en IPV6 simuladas para SDN, OSPF y MPLS se obtuvieron los siguientes resultados:

Figura 54. Jitter de SDN, OSPF y MPLS en IPv6



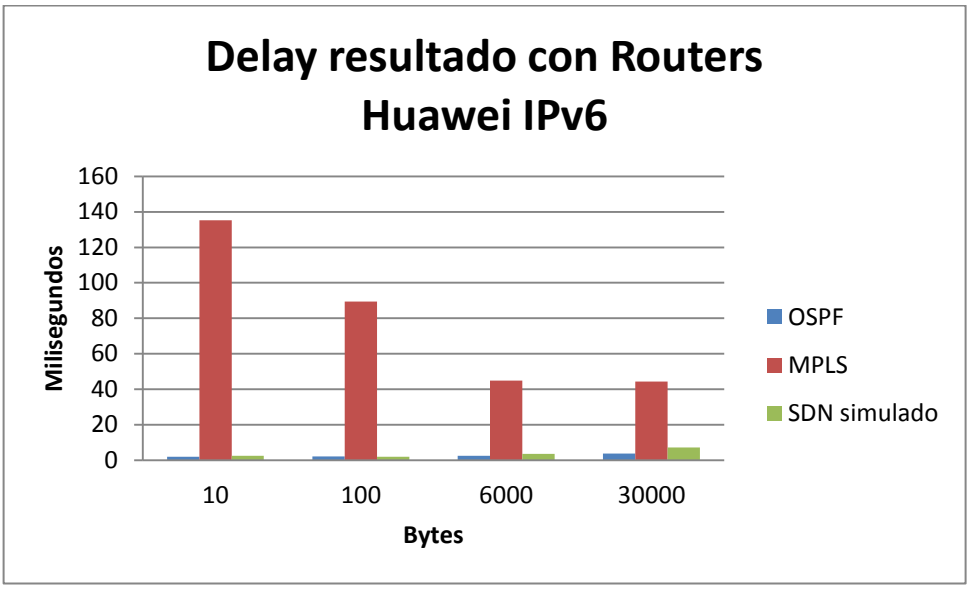
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

El Jitter en las pruebas de IPv6 simulado, se observa que para OSPF se obtuvo un comportamiento igual que el Delay, siendo elevado para 10 y 100 Bytes, tanto que para MPLS se mantiene durante las pruebas de 10, 100 y 6000 Bytes, por otro lado, para la de 30000 Bytes se observa que aumenta ya que la realización de un túnel IPv6 sobre IPv4 hace que tenga más procesamiento el PC y no se pueda observar bien la simulación.

7.7 PRUEBAS DE DELAY CON LOS ROUTERS HUAWEI EN IPv6

Para las pruebas de Delay realizadas en IPV6 para OSPF y MPLS se obtuvieron los siguientes resultados:

Figura 55. Delay de OSPF y MPLS en IPv4 (Routers Huawei)



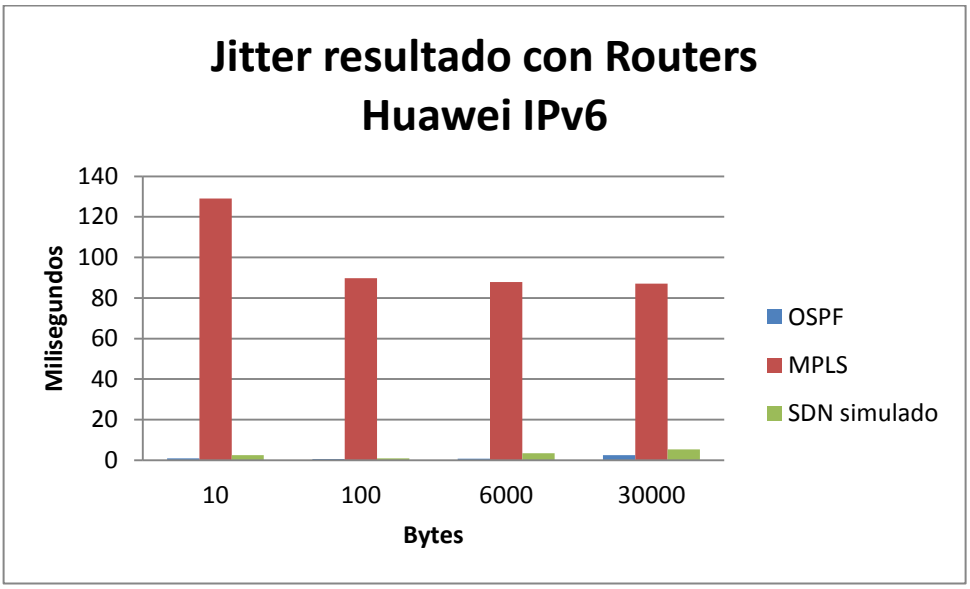
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Para las pruebas con los routers Huawei en IPv6, los valores son elevados para MPLS, además cabe recordar que se realizó un Túnel IPv6 sobre IPv4, por ultimo para OSPF (routers HUAWEI) y SDN emulado los valores son muy bajos.

7.8 PRUEBAS DE JITTER SIMULADAS EN IPv6

Para las pruebas de Jitter realizadas en IPV6 para OSPF y MPLS se obtuvieron los siguientes resultados:

Figura 56. Jitter de OSPF y MPLS en IPv4 (Routers Huawei)



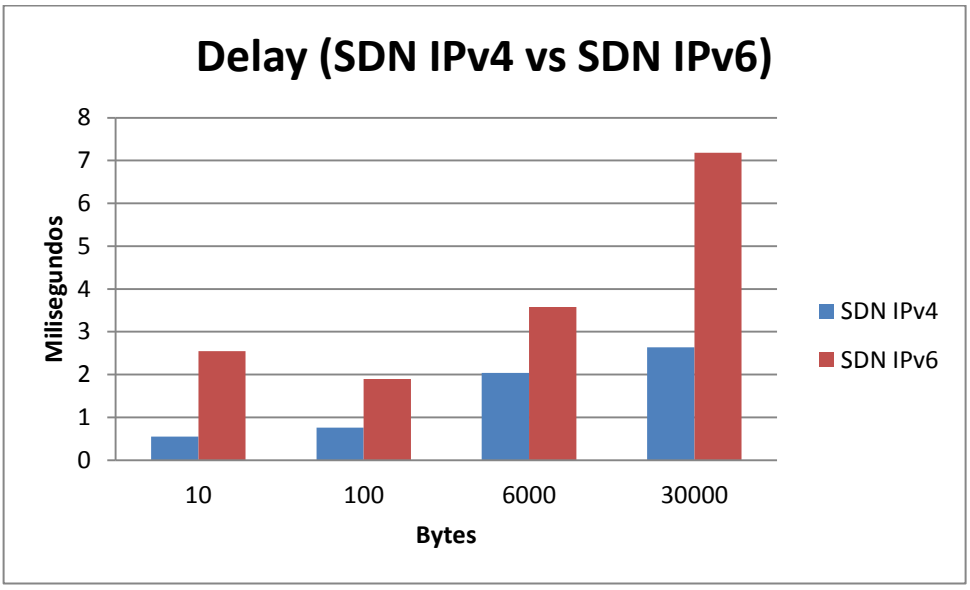
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se obtienen los mismos resultados para MPLS con valores muy elevados, además para OSPF y SDN emulado los valores son bajos.

7.9 DELAY DE SDN (IPv4 VS IPv6)

Para las pruebas de Delay de SDN para IPv4 e IPV6 se obtuvieron los siguientes resultados:

Figura 57. Delay (SDN IPv4 vs IPv6)



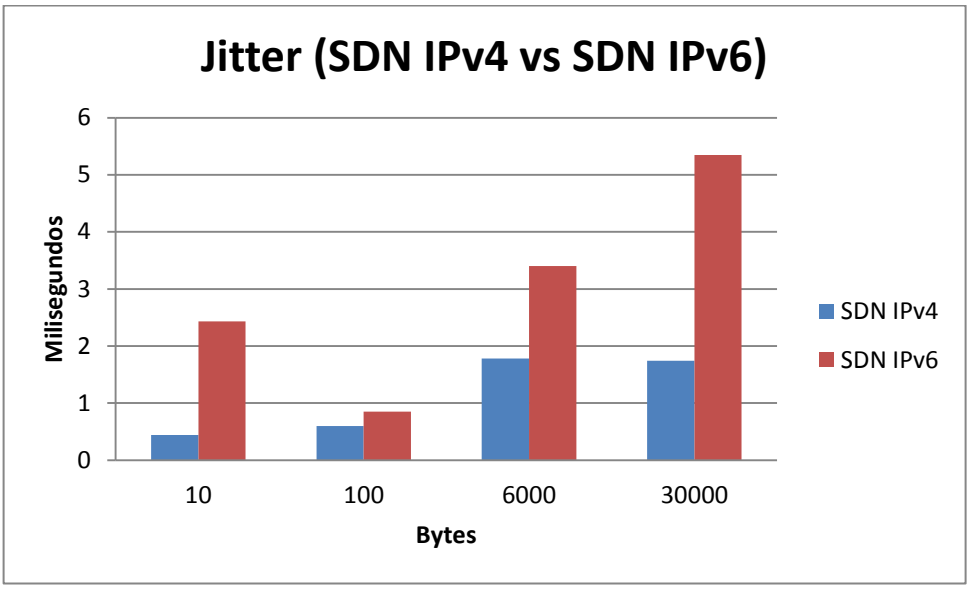
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se obtuvo que para las pruebas de IPv4 el Delay va en orden ascendente, para IPv6 se observa un valor pequeño para 100 Bytes para las demás los valores se encuentran entre 3 y 8 ms.

7.10 JITTER DE SDN (IPv4 VS IPv6)

Para las pruebas de Jitter de SDN para IPv4 e IPV6 se obtuvieron los siguientes resultados:

Figura 58. Jitter de SDN simulado IPv4 vs IPv6



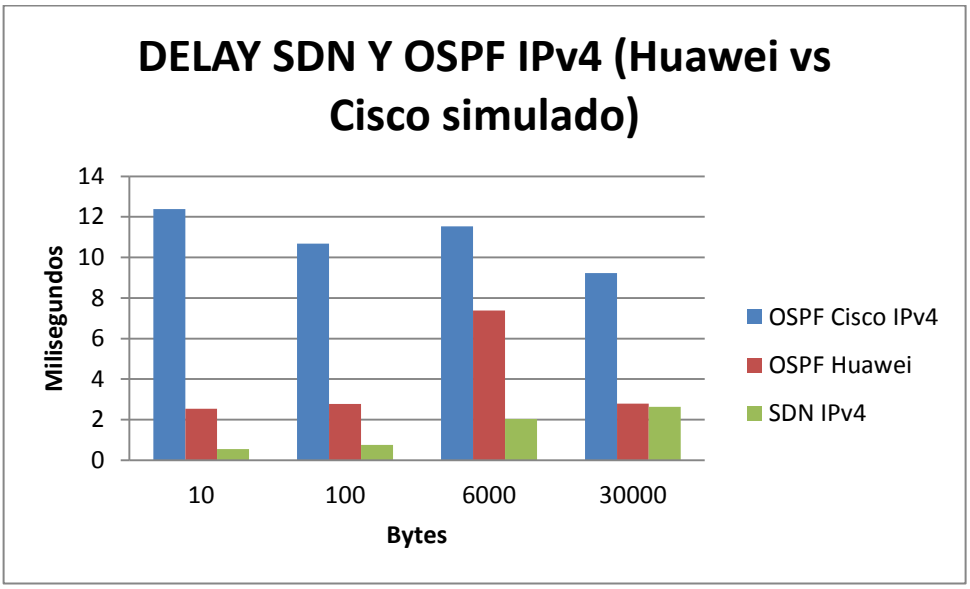
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Los valores de Jitter para SDN se comportan de la misma manera que el Delay siendo más rápido en IPv4.

7.11 DELAY DE SDN Y OSPF IPv4 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de SDN y OSPF para IPv4 en Huawei y Cisco Simulado se obtuvieron los siguientes resultados:

Figura 59. Delay de SDN y OSPF IPv4 (Huawei vs CISCO simulado)



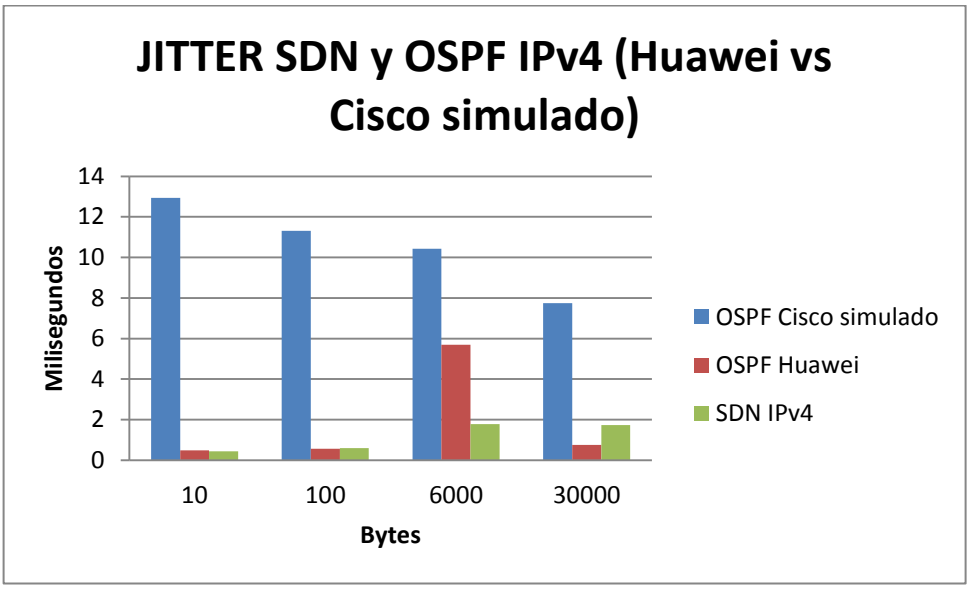
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Los valores de OSPF en Cisco simulado oscilan entre 12 y 9 ms siendo más elevados que los de las pruebas físicas con los routers de los equipos Huawei, donde solo hay una variación cercana a 8 ms, los demás valores oscilan entre 2 y 3 ms. Para SDN los valores son bajos para 10, 100, y 6000, en cuanto que a 30000 el valor es muy parecido que al de OSPF en Huawei.

7.12 JITTER DE SDN Y OSPF IPv4 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Jitter y Delay en SDN y OSPF para IPv4 en Huawei y Cisco Simulado se obtuvieron los siguientes resultados:

Figura 60. Jitter de SDN y OSPF IPv4 (Huawei vs CISCO simulado)



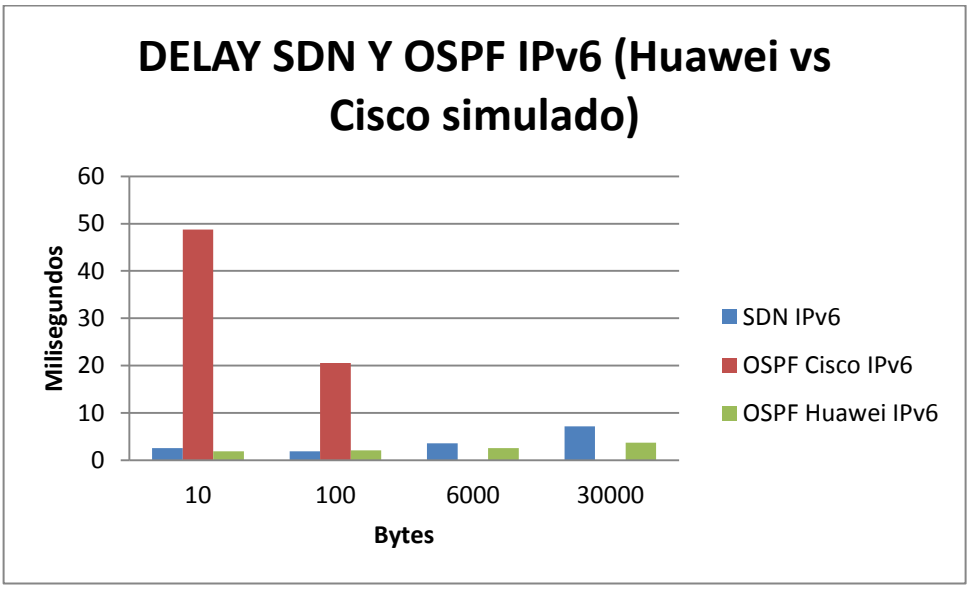
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Los valores de Jitter para OSPF en CISCO simulado son altos con respecto a OSPF en Huawei, donde los valores van de 0,8 a 1 ms incluso siendo más bajos de SDN emulado.

7.13 DELAY DE SDN Y OSPF IPv6 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Delay en SDN y OSPF para IPv6 en Huawei y Cisco Simulado se obtuvieron los siguientes resultados:

Figura 61. Delay de SDN y OSPF IPv6 de (Huawei vs CISCO simulado)



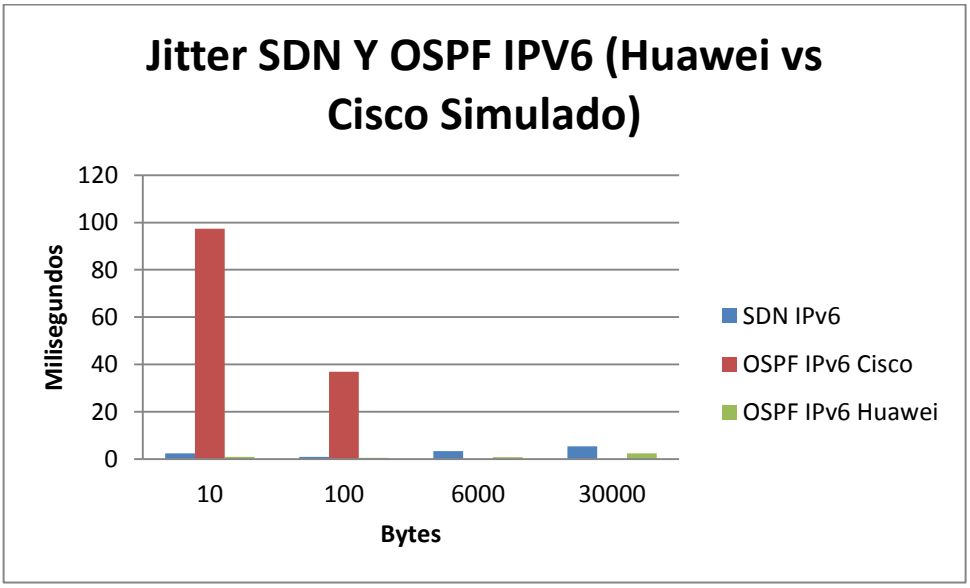
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

El Delay de OSPF en IPv6 de CISCO es alto para 10 y 100 Bytes, en cuanto a OSPF en Huawei en IPV6 los valores son más bajos que SDN en IPv6 emulado.

7.14 JITTER DE SDN Y OSPF IPv6 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Jitter de SDN y Delay de OSPF para IPv6 en Huawei y Cisco Simulado:

Figura 62. Jitter de SDN y OSPF IPv6 de Huawei vs CISCO simulado



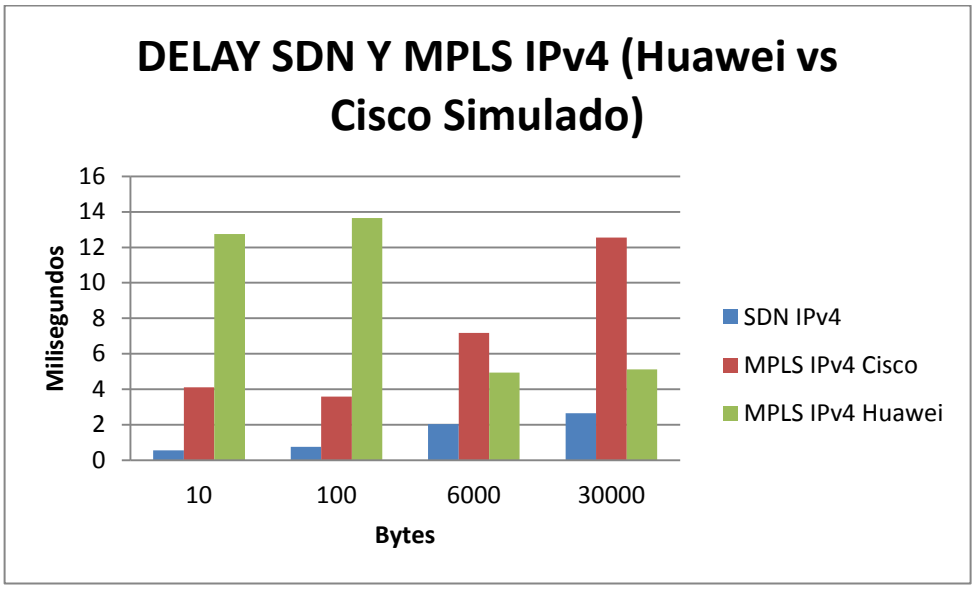
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera**. *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

El Jitter de OSPF en IPv6 de CISCO es muy elevado para valores de 10 y 100 Bytes, en SDN y OSPF en Huawei para IPv6 oscilado entre 0 y 8 ms.

7.15 DELAY DE SDN Y MPLS IPv4 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Delay en SDN y MPLS para IPv4 en Huawei y Cisco Simulado:

Figura 63. Delay de SDN y MPLS IPv4 (Huawei vs CISCO simulado)



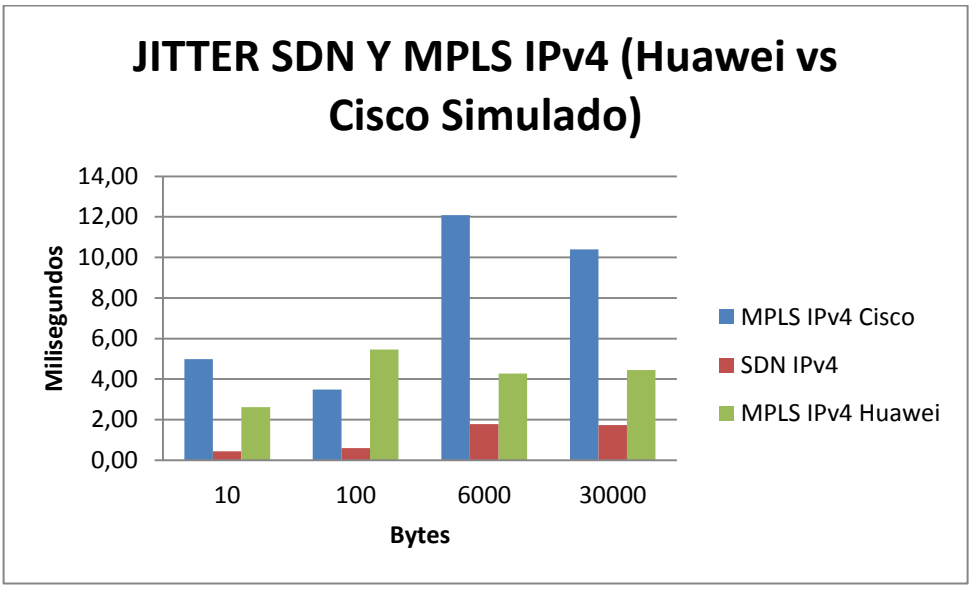
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

Se observa que para el Delay de MPLS en IPv4 es elevado ya que se realizó una VPN en capa 3, pero para 6000 y para 30000 Bytes los valores bajan, siendo estos, los valores más altos para MPLS IPv4 en CISCO. SDN emulado continua siendo más rápido que los anteriores.

7.16 JITTER DE SDN Y MPLS IPv4 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Jitter de SDN y Delay de MPLS para IPv4 en Huawei y Cisco Simulado:

Figura 64. Jitter de SDN y MPLS IPv4 de (Huawei vs CISCO simulado)



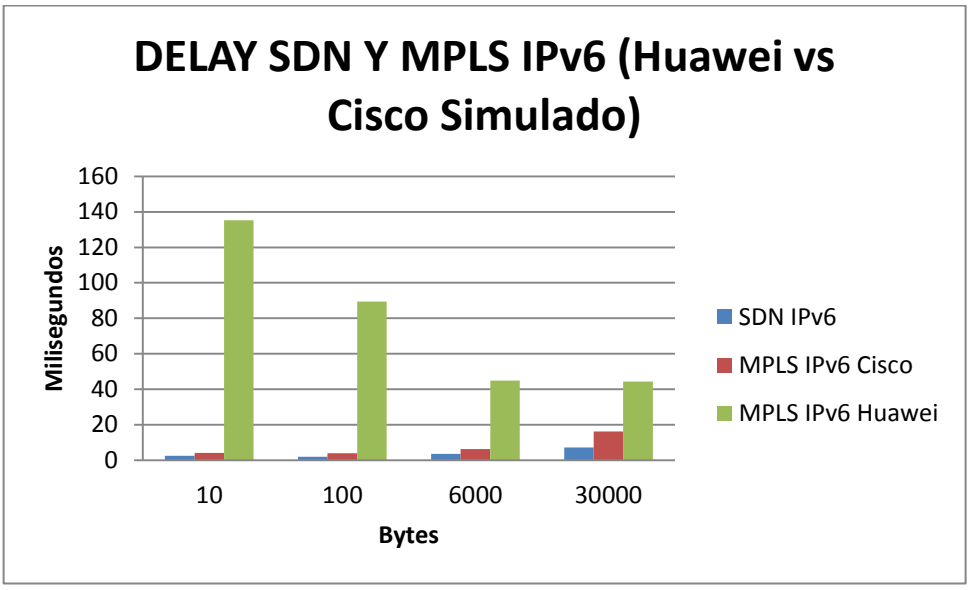
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

MPLS en HUAWEI como en CISCO tienen grandes cambios en este caso para 6000 y 30000 Bytes son los valores más elevados para MPLS de CISCO, en HUAWEI se observa que el valor más elevado esta entre 4 y 5 ms, por otro lado SDN sigue dominando entre estas variables.

7.17 DELAY DE SDN Y MPLS IPv6 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Delay en SDN y MPLS para IPv6 en Huawei y Cisco Simulado:

Figura 65. Delay de SDN y MPLS IPv6 (Huawei vs CISCO simulado)



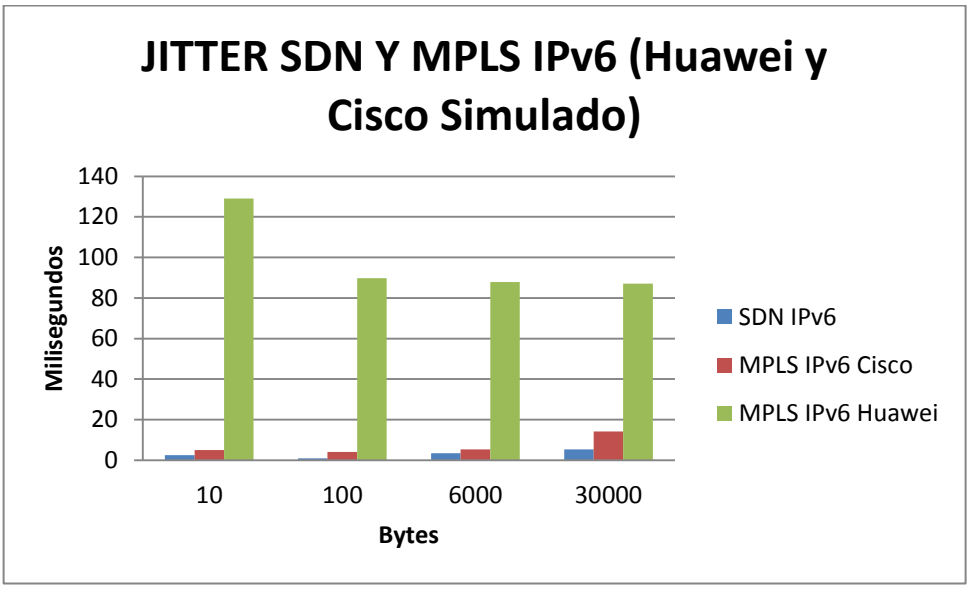
Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

En el proceso del túnel IPv6 sobre IPv4, se evidencio que esta lleva más desarrollo y es más demorada por lo tanto los valores de MPLS IPv6 en HUAWEI son muy altos con respecto a SDN y MPLS de CISCO simulado en IPv6.

7.18 JITTER DE SDN Y MPLS IPv4 (HUAWEI vs CISCO SIMULADO)

Para las pruebas de Jitter en SDN y Delay en MPLS para IPv6 en Huawei y Cisco Simulado:

Figura 66. Jitter de SDN y MPLS IPv6 (Huawei vs CISCO simulado)



Fuente: **Julián Camilo Sombredero Alfonso, Erick Ferney Silva Herrera.** *Análisis comparativo de prestaciones entre SDN (Software Defined Networking) y redes IP convencionales, 2014.*

El Jitter tiene un comportamiento similar que el Delay en estas variables, ya que el túnel IPv6 sobre IPv4 requiere más procesamiento en los routers de HUAWEI y por lo tanto tendrá más bits en la cabecera de los paquetes enviados.

8. RECOMENDACIONES

Durante el desarrollo de este proyecto se probaron distintas soluciones para implementar la topología de red mostrada en la figura 24, encontrando problemas al realizar los experimentos.

En OSPF se tiene inconvenientes con el entorno simulado, ya que para las pruebas sobre IPv6 con 6000 y 30000 Bytes no se pudieron realizar por la pérdida completa de paquetes.

Para MPLS con routers Huawei en IPv4 e IPv6 surgió la dificultad de no tener actualizado el software de estos, teniendo como impedimento la no realización de una VPN L2, por lo tanto se realizó una VPN Instance donde solo funcionó para IPv4, como consecuencia para IPv6 se realizó un túnel IPv6 sobre IPv4.

Por ultimo en SDN surgieron problemas con la virtualización de las máquinas de Linux y la instalación de los paquetes, así mismo no se pudo realizar por completo el entorno de red con un computador bajo una máquina virtual de 64 Bits, ya que por comandos y por la interfaz gráfica presentaba problemas al implementar switches OpenFlow.

En consecuencia las razones anteriores impulsan a escribir estas sugerencias a tener en cuenta para los lectores:

- Tener computadores con suficiente memoria RAM mayores a 8 Gb y como mínimo un procesador Core I5.
- Actualizar el debido software de los routers a trabajar para no tener inconvenientes a futuro.
- Preferiblemente tener en los PC un sistema operativo Windows 7 (Opcional)
- Disponer de Switches físicos SDN para obtener un mejor rendimiento en las pruebas.

9. FUNDAMENTACIÓN HUMANÍSTICA

Dentro de la misión de la Universidad Santo Tomás se destaca la formación integral y el carácter ético, creativo e investigativo que tiene que tener cada uno de los profesionales, ya que el hombre como un ser humano sin perfección puede llegar a esta mediante sus estudios y su educación. Para ello el aprendizaje es un proceso que transforma estados, imperfecto a perfecto, a lo cual Santo Tomás lo llama estado de virtud, donde las aptitudes perfeccionan aquellas facultades de vida y las capacidades técnicas y profesionales.

Además para la formación integral es importante tener un proceso de investigación, docencia y de proyección social ayudando a las necesidades en las que vive el país, desarrollando técnicas y herramientas esenciales para la solución de problemas. Por otro lado para la Facultad de Ingeniería de Telecomunicaciones tiene como objetivo formar ingenieros que tengan un buen impacto positivo en la sociedad con fundamentos éticos y morales buscando el bien común, principio de Santo Tomás.

Desde este punto de vista, el desarrollo de este proyecto hace uso de los anteriores procesos formativos, teniendo en cuenta las capacidades en la integración de una nueva tecnología a las actuales, por lo tanto este proceso de Software Defined Networking conlleva a la implementación y su progreso para los operadores de telecomunicaciones ayudando a la disminución de recursos económicos para ellos y unos menores costos para el usuario.

10. CONCLUSIONES

Este documento manifestó las metodologías y procedimientos que se requieren para medir el Delay y Jitter realizando una topología de red en IPv4 e IPv6 trabajada en OSPF, MPLS y SDN, siendo esta última el futuro de las telecomunicaciones, evidenciando la investigación, desarrollo y análisis de los objetivos cumplidos, para así poder llegar a exponer las siguientes conclusiones del estudio realizado:

- Según las pruebas realizadas en las variables en IPv4 con routers Huawei, Software Defined Networking emulado tiene un delay menor en un 78,27% que OSPF para la prueba de 10 Bytes, además se observa que para las pruebas de 100 y 6000 bytes SDN tiene un delay menor en un 72,6 % que OSPF. Ahora bien para MPLS, SDN tiene un delay menor en 95,69% y 94,35% con 10 y 1000 Bytes respectivamente, tanto que para 6000 y 30000 Bytes SDN tiene un delay menor en 58,64% y un 48,44% que MPLS.

- Al realizar las prueba con los routers físicos HUAWEI sobre IPv6, OSPF tiene un Delay y Jitter más menor que SDN y MPLS, cabe destacar que en MPLS para IPv6 se realizó un túnel IPv6 sobre IPv4 por lo tanto la transmisión de tantos bytes en el header conllevara a que el retardo será mucho mayor que en MPLS IPv4 con la VPN.

- En la implementación de SDN se observó que durante la instalación de paquetes y su ejecución no se podía realizar ninguna topología de red para una máquina virtual de 64 bits por consiguiente se debió a realizarlo con un sistema de 32 bits, teniendo éxito en las pruebas sobre SDN.

- Al realizar todas las pruebas se percibió que para 6000 y 30000 Bytes, MPLS tiene un Delay y Jitter menor para los routers HUAWEI físicos, cabe destacar

que se transportaron más Bytes en su cabecera ya que se implementó una VPN instance y tiene más procesamiento, por lo tanto la diferencia de delay entre la prueba de 6000 es de 11,94% y para la de 30000 es de un 9,18%.

- Las pruebas de las variables en IPv6, demostraron que Software Defined Networkinging emulado, tiene un Delay y Jitter más bajo que los protocolos estudiados en CISCO (simulado) como en HUAWEI (con routers físicos). Por lo tanto también se contempla que MPLS tuvo un tiempo de respuesta demasiado largo ya que al realizar un túnel IPv6 sobre IPv4 su necesidad de procesamiento aumenta.

- OSPF para IPv6 tiene menor Delay y Jitter que IPv4 pues aunque IPv6 tiene un encabezado más grande, este tiene menos campos que procesar, por lo tanto se cumple lo que está especificado en la RFC 2460 para IPv6 (IEFT, 2006)y la RFC 791 para IPv4 (IETF. Mariana de Rey).

BIBLIOGRAFÍA

(3GGP), 3. G. (2006-6). *IP Multimedia Subsystem(IMS); Stage 2 (Release 5).*

ÁVILA MEJÍA, O. (2011). *Migración del Protocolo IPv4 a IPv6.* Cuidad de México: Uiversidad Autónoma de México.

Big Switch Networks. (2014). *Big Cloud Fabric*. Obtenido de www.bigswitch.com/sdn-products/big-cloud-fabric

Big Switch Networks. (2014). *Open SDN*. Obtenido de www.bigswitch.com/products/open-sdn

Bob Lantz, Nikhil Handigol, Brandon Heller, and Vimal Jeyakumar. (2014). *Introduction to Mininet* . (GitHub) Obtenido de https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

BROCADE Communications Systems. (2009). *Technical Brief: Offering Scalable Layer 2 Services .*

Cádiz, U. d. (2012). *Guía: Actividad 1.5.1 en GNS3.* España: Escuela Superior de Ingeniería .

Calidad de Servicio en Redes IP. (s.f.). Obtenido de http://www.spw.cl

Calidad de Servicio en Redes IP. (s.f.). Obtenido de http://www.spw.cl/08oct06_ra/doc/REDES%20WAN%20IP-ATM/CalidaddeservicioenredesIP.pdf

CISCO. (Marzo de 2013). *Software-Defined Networks and OpenFlow*. Obtenido de www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/161_sdn.html

Cisco CCNA - CCNP. (2003). *Capítulo 11, Capa 3 Protocolos.*

Cisco. (s.f.). *Cisco Networking Academy Program. CCNA 1 and 2.* Version 3.1.

CISCO. (s.f.). *IPv6 Routing: OSPFv3*. Obtenido de http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/15-sy/iro-15-sy-book/ip6-route-ospfv3.html

CISCO. (s.f.). *Soft-ware-Defined Networks and OpenFlow*. (The Internet Protocol Journal, Volume 16, No. 1) Obtenido de http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/161_sdn.html

CISCO. (s.f.). *Virtual Routing and Forwarding.* Technology Support and Information Model Reference Manual .

Damw, W. (s.f.). *Introducción al Jitter.* United States: Wireles Telecom Group.

Dorado, M. D. (2004). *Soporte de Garantías de Servicio (GoS) sobre MPLS mediante Técnicas Activas.* España.

FERNÁNDEZ, A. (2010). *Tutorial de IPv6.* México: Universidad Nacional Autónoma de México.

Force Internet Engineering Task. (1998). *OSPF Version 2.* RFC 2328.

Gil, F. H. (s.f.). *Introduction al OSPF.*

GNS3. (s.f.). *Introduction to GNS3*. Obtenido de http://www.gns3.net/

hackage and Cabal 1.20.0.1. (s.f.). *http://hackage.haskell.org/package/nettle-openflow-0.2.0*. Obtenido de http://hackage.haskell.org/package/nettle-openflow-0.2.0

Hata, H. (2013). *A Study of Requirements for SDN Switch Platform.* Tokyo, Japan: NTT Communications.

HP. (2014). *Software-defined Networking*. Obtenido de Revive la Innovación con el Ecosistema abierto SDN: http://h17007.www1.hp.com/es/es/networking/solutions/technology/sdn/

HP. (2014). *Switches HP Networking*. Obtenido de Soluciones de switch para redes Ágiles: www1.hp.com/us/en/networking/products/switches/index.aspx#.U-FTI-N5MXs

HUAWEI. (2003). *Quidway S9300 Terabit Routing Switch, V100R001C02, Configuration Guide - VPN.* Huawei Technologies Co., Ltd.

Huawei. (April de 2013). *Carrier SDN: Next-gen carrier networking*. Obtenido de www.huawei.com/en/about-huawei/publications/communicate/hw-259729.htm

HUAWEI. (18 de Marzo de 2014). *Huawei Launches Industry's First Software-defined Networking App in Google Play and App Store*. Obtenido de pr.huawei.com/en/news/hw-329662-sdn.htm#.U-EfAeN5MXs

HUAWEI, Carrier Network. (s.f.). *Reshaping the Future of Network Arquitecture*. Obtenido de www.huawei.com/en/solutions/softcom/

IEFT. (2006). *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6.* California: RFC 4443.

IETF. Mariana de Rey. (s.f.). Internet Protocol. California: RFC 791, 1981.

Internet Engineering Task Force. (1998). *Internet Protocol, versión 6.* RFC 2460.

IRIS:. (s.f.). *The Recursive SDN Openflow Controller by ETRI*. Obtenido de OpenIRIS: http://openiris.etri.re.kr/

Jesús, G. T. (2002). *Alta Velocidad y Calidad de Seervicio en Redes IP.* Alfa Omega.

Linux Foundation. (2014). *OpenDAYLIGTH*. Obtenido de http://www.opendaylight.org/

M., R. L. (2008). Subsistemas Multimedia IP (IMS) en 3GPP y 3GPP. *IEEE*.

Mejía, F. (2011). *Análisis de Parametros de QoS*. Obtenido de http://saber.ucv.ve

Mejía, F. (2011). *Análisis de Parametros de QoS*. Obtenido de http://saber.ucv.ve/jspui/bitstream/123456789/731/3/Anexo%202.pdf

Mejía, O. (2011). *Migración del Protocolo IPv4 a IPv6.* México: Departamento de Ingenieria Eléctrica, UAM.

Mininet Team . (2014). *An Instant Virtual Network on your Laptop (or other PC)*. Obtenido de http://mininet.org/

Minsterio de Tecnologias de la Información y las Comunicaciones . (2009). *Ley 1341 del 2009.* Bogotá.

Moya, J. M. (2002). *MPLS (ultiprotocol Label Switching).* España: Bit 152.

NetFPGA. (2014). *NetFPGA*. Obtenido de http://netfpga.org/learn_more.html

NetFPGA, I. a. (s.f.). *http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/LearnMore.1.1, 2.3.1*.

Network Computing. (12 de 2 de 2014). *Network Computing*. Obtenido de Software-Defined Networking's 3 Biggest Benefits: www.networkcomputing.com/networking/software-defined-networkings-3-biggest-benefits/a/d-id/1113808

Network Starup Resource Center. (s.f.). *RYU OpenFlow Controller.* Oregon: University of Oregon.

Network Working Group . (1981). *Internet Control Message Protocol.* RFC 972.

Nox. (s.f.). *NOX and POX*. Obtenido de http://www.noxrepo.org/pox/about-pox/

OFN. (2012). *Software Defined Networking the new norms for networks.*

OFN. (2014). *Open Networking Foundation*. Obtenido de https://www.opennetworking.org

OFN. (17 de February de 2014). *OpenFlow-enable SDN and, OFN Solution brief*.

OFN. (s.f.). *Open Networking Foundation*. Obtenido de https://www.opennetworking.org

ONF. (2013). *SDN Architecture Overview.*

Open Mul. (13 de 05 de 2012). *SDN Openflow controller*. Obtenido de http://sourceforge.net/projects/mul/

Open Networking Foundation. (s.f.). *https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers*.

Open vSwitch. (s.f.). *Production Quality, Multilayer Open Virtual Switch*. Obtenido de http://vswitch.org/

OpenFlow. (s.f.). Obtenido de http://archive.openflow.org

OpenFlow. (2011). *Pantou : OpenFlow 1.0 for OpenWRT*. Obtenido de http://archive.openflow.org/wk/index.php/OpenFlow_1.0_for_OpenWRT

OpenFlowHub. (2010). *Indigo - Open Source OpenFlow Switches - First Generation*. Obtenido de http://www.openflowhub.org/display/Indigo/Indigo+-+Open+Source+OpenFlow+Switches+-+First+Generation

Orzach, Y. (December 2013). *Network Analysis Usnig Wireshark Cookbook.* Packt Publishing.

PÉREZ HERRERA, E. (2003). *Tecnologías y Redes de Transmisión de Datos.* México: Limusa, SA.

QLogic. (s.f.). *Introduction to Ethernet Latency*. Obtenido de www.qlogic.com/Resources/Documents/TechnologyBriefs/Adapters/Tech_Brief_Introduction_to_Ethernet_Latency.pdf

Redes Cisco.NET. (s.f.). *CCNA, CCNP, Wireless, y networking en general*. Obtenido de Tipos de Áreas en OSPF.

Redes Cisco.NET. (s.f.). *CCNA, CCNP, Wireless, y networking en general*. Obtenido de Tipos de Áreas en OSPF: http://www.redescisco.net/v2/art/tipos-de-areas-en-ospf/

RENATA . (s.f.). Recuperado el 20 de Marzo de 2014, de RENATA: www.renata.edu.co

REY, M. d. (1981). *IETF. Internet Protocol.* California: RFC 791.

RFC 2544. (s.f.). *Benchmarking Methodology for Network Interconnect Devices*. Obtenido de www.ietf.org/rfc/rfc2544.txt

Ruy SDN Framework Community. (2014). *Component _Based Software Defined Networking Framework*. Obtenido de http://osrg.github.io/ryu/

SDN Hub. (2014). *OpenFlow version 1.3 Tutorial*. Obtenido de http://sdnhub.org/tutorials/openflow-1-3/

SITIME. (s.f.). *Clock Jitter Definitions and Measurement Methods*. Obtenido de www.sitime.com/support2/documents/AN10007-Jitter-and-measurement.pdf

Sombredero, J. C., & Silva, E. F. (2014). *Analisis Comparativo de Prestaciones entre SDN y Redes IP Convencionales.* Bogotá D.C.

Specification, OpenFlow Switch. (Sept. 2012). *https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf*.

Tapasco, M. O. (2008). *MPLS, El Presente de las Redes IP.* Pereira: Universidad Tecnologíca de Pereira.

Unión Internacional de Telecomunicaciones (UIT). (2005). *Manual sobre Redes Basadas en el Protocolo Internet (IP) y asuntos conexos.*

Universidad de Alcalá. (2012). *Arquitectura y Tecnología de Computadores.* España: Departamento de Informática.

Universidad Nacional del Rosario. (2006). *Sistemas Distribuidos. El Protocolo IPv6.* Depatamento de Electrónica, Faculta de Ciencias Exactas .

VELÁZQUEZ E, G. (2009). *Análisis Comparativo de Herramientas de Evaluación de Desempeño en Redes de Computadores.* Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación.

WADDINGTON, D. G. (2012). *Realizing the Transition to IPv6.* Bell Research Laboratories.

ZHANG, Y. -L. (2004). *IPv6 Conformance Testing: Theory and Practice.* Beijing, China: Institute of Computing Technology, Chinese Academy of Sciences.