Revista de Ingeniería y Región
2014 (11): 57-71

Implementación de *OpenFlow* sobre *NetFPGA*

Implementation of OpenFlow on NetFPGA

Johnnathan Machado [1], Andrés Felipe Ramos [2] y Juan Carlos Cuéllar [3]

Resumen

Las redes tradicionales están diseñadas e implementadas con diferentes dispositivos de red, como *routers*, *switches* y *firewalls*; cada uno tiene unas funciones asociadas y es configurado con diferentes estándares y protocolos para realizar la trasmisión de paquetes. Pero este modelo tradicional no es el único que existe, hoy en día se está empezando a imponer un nuevo modelo de red definida por software [SDN por *Software Define Networks*]. Si bien este modelo usa los mismos dispositivos de red, es diferente pues utiliza las ventajas del software de programación, el cual mediante una interfaz universal controla los dispositivos de red y programa los servicios de envío o enrutamiento de paquetes. Este artículo presenta las ventajas de las SDN respecto de las redes tradicionales y muestra pruebas de desempeño básicas realizadas en un ambiente de laboratorio con una tarjeta NetFPGA implementando OpenFlow, en contraste con pruebas de desempeño realizadas con dispositivos de red comerciales.

Palabras Clave: Redes definidas por Software; OpenFlow; Control de flujo; switch; router.

Abstract

Traditional networks are designed and implemented with network devices such as *routers*, *switches*, *firewalls*, etc. Each network device has associated functions and is configured with different standards and protocols for packet transmission. But this traditional model is not the only one that exists, today is beginning to impose a new model of software-defined network, known as SDN (*Software Define Networks*). While SDN also uses the same network devices, the difference is that these networks are using the advantages of programming software, which means to use a universal control interface; this interface controls the network devices and program delivery services or routing packets. This article presents the advantages of SDN over the traditional networks and displays basic performance tests conducted in a laboratory environment with implementing OpenFlow in a NetFPGA card, in contrast to performance tests conducted with commercial network devices.

Keywords: Software defined networking; OpenFlow; Flow´s control; Switch; Router.

1 Ingeniero Telemático. Universidad ICESI, Cra 48 # 10-09. Cali-Colombia. johnnathan14@hotmail.com
2 Estudiante Ingeniería Telemática. Universidad ICESI, Cra 48 # 10-09. Cali-Colombia. anfera103@gmail.com
3 Ingeniero Electricista. Magister en Ingeniería. Candidato a Doctor en Ingeniería Telemática. Director Programa Ingeniería Telemática. Universidad ICESI, Cra 48 # 10-09. Cali-Colombia. jcuellar@icesi.edu.co

Recibido 3 de marzo de 2014 Aprobado 30 de abril de 2014

1. Introducción

Las redes tradicionales a medida que crecen son notoriamente más difíciles de administrar. Esto se debe en buena parte a que cada dispositivo de red contiene conjuntamente dos funcionalidades, una para los datos y otra para el control, esto conlleva a una serie de inconvenientes. Uno de ellos es que se hace necesario configurar cada dispositivo de red por separado mediante comandos o una interfaz web, ya que a medida que la red va aumentando en tamaño, es necesario adquirir más dispositivos de red convirtiendo la tarea de administración más compleja porque hay que ajustar las direcciones de red, las reglas de los firewall, etc. Estos ajustes toman tiempo y no escalan bien si se agregan numerosos dispositivos de red de manera simultánea (Strom, 2013). Otro inconveniente es el hecho de que una red no es estática, la topología está propensa a cambios, ya sea por la necesidad de expansión de la misma o por las caídas de los diferentes enlaces durante el funcionamiento normal, lo que obliga a agregar tareas a los equipos que pueden aumentar la latencia en su procesamiento.

La división de las funcionalidades de datos y control se pensó como una forma de resolver el problema de administración de las redes actuales, y fue de esta idea que nació la noción de SDN (Software Define Network) o redes definidas por software, donde se busca un control centralizado, mientras que los dispositivos puedan dedicar todos sus recursos al envió y direccionamiento de datos. Lo que se busca con las SDN es ofrecer a los administradores de red la flexibilidad necesaria para configurar, administrar, asegurar y optimizar los recursos de la red a través de programas de automatización definidos por una capa de control.

Bajo el concepto de SDN, se ha creado el estándar OpenFlow. El cual es el primer estándar de comunicación que define un protocolo para la interacción entre el software de control y el hardware de red, como lo describe la arquitectura de Redes definidas por software. OpenFlow permite el acceso directo y la administración de los dispositivos de red tales como *switches* y *routers*, mediante primitivas básicas que pueden ser utilizadas por un software de aplicación externo para programar el reenvío de datos en los dispositivos de red (ONF,2012).

SDN es un concepto que está llamando la atención; en universidades de los Estados Unidos ya inició la implementación del estándar OpenFlow. Existen casos como el de la Universidad de Stanford en California, donde se encuentra en ejecución el protocolo OpenFlow como estándar de comunicación entre sus dispositivos de red, tanto de forma interna, como para la comunicación con otras universidades de los Estados Unidos –como las de Wisconsin e Indiana (Stanford, 2013). Ellos controlan el flujo de la información mediante los controladores de los dispositivos de interconectividad a través de software. También gigantes como Google y Facebook han decidido implementar OpenFlow (Strom, 2013). El Vicepresidente Senior de operaciones de Google anuncio que desde principios de 2012 ya se encontraba funcionando con el estándar OpenFlow (Open-Flow-20120512).

2. Metodología

Redes definidas por software

Los orígenes de las redes definidas por software (SDN-*Software Define Networks*) provienen de la investigación de los profesores Nick McKeown de Stanford, Scott Shenker de Berkeley y un grupo de sus estudiantes. El proyecto, llamado *Ethane*, era una arquitectura de software para corporaciones que permitía definir una política global para la red, funcionaba con *switches* Ethernet normales y con un control centralizado (Casado, 2008). Se tenía entonces un controlador que almacenaba la política global, la cual determinaba el destino de cada paquete de datos. El controlador conocía toda la topología para hacer el enrutamiento de los datos y una administración de permisos de

acceso. El segundo componente eran los *switches* Ethernet, que contienen únicamente una tabla de envío y un canal seguro al controlador.

Uno de los estudiantes que trabajó en "Ethane", Martin Cascado, creó después un pequeño sistema SDN que fue adquirido en 2012 por VMware. Sus esfuerzos dieron como fruto el protocolo OpenFlow (Strom,2013). Más a delante este protocolo obtuvo su propia Fundación, la *Open Networking Foundation* (ONF, 2012), la cual se encarga de la promoción y adopción de SDN a través del desarrollo de estándares abiertos.

La idea de las SDN no es nueva, aunque las empresas han invertido en virtualización y tecnología de Cloud Computing aún no han logrado la habilidad de proveer una conexión de red de forma inmediata. La causa de esto tiene que ver con la poca flexibilidad de una infraestructura donde sus dispositivos cuentan con una funcionalidad de datos y otra de control, y se requieren tener en cuenta parámetros de configuración de red como la direcciones IP o las reglas del Firewall, esto sin contar con la variedad de aplicaciones y protocolos ejecutándose de manera simultánea. Como resultado aparecen problemas en la administración y complejidad en la infraestructura de las redes a medida que estas van creciendo en usuarios y aplicaciones.

La intención con las SDN fue separar los diferentes dispositivos de red, como un enrutador, su parte de datos y su parte de control, para colocar una interfaz de programación entre ambas partes y poder lograr una "abstracción" de ambas funcionalidades. Con esta abstracción se busca dividir el problema de infraestructura en piezas más sencillas de tratar (Shenker, 2011). Y este enfoque está claramente plasmado en la definición propuesta por la ONF, 2012: "*En la arquitectura de la SDN, los planos de control y de datos están*

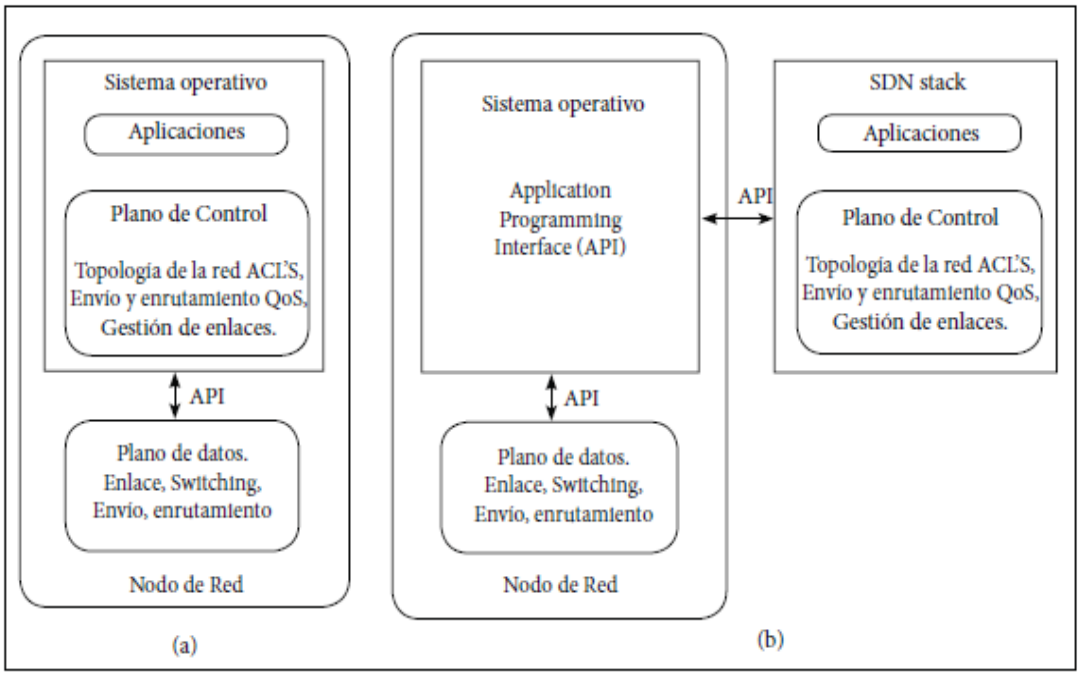

Figura 1. Comparación entre el enfoque tradicional de red (a) y el enfoque basado en SDN (b). Adaptado de Sezer, 2013.

desacoplados, la inteligencia y estado de la red están lógicamente centralizados, y la infraestructura subyacente de red es abstraída desde las aplicaciones". SDN se enfoca en cuatro características principales: separación del plano de control del plano de datos, un control y vista centralizada de la red, interfaces abiertas entre dispositivos en el plano de control y el plano de datos y programación de la red por aplicaciones externas (Sezer, 2013).

La Figura 1 refleja la diferencia entre el enfoque tradicional de las redes de datos y el enfoque de SDN; en su parte (a) se aprecia el enfoque tradicional, en el cual el plano de control es responsable de la configuración del dispositivo de red y de la programación de los caminos para los flujos de datos; en su parte (b), se presenta el enfoque basado en SDN, en el cual el control es trasladado fuera del dispositivo de red en un control centralizado.

La primera versión de SDN (SDNv1) estaba conformada por dos partes:

- Abstracción de envío: Que permitía tener un modelo flexible de reenvío en la funcionalidad de control de los dispositivos de red, hardware "abstracto", que permite obtener soluciones no atadas a vendedores específicos y flexibilidad en su arquitectura.
- Sistema Operativo de Red o NOS (Network Operating System): Que corre en los controladores en la red, crea una vista de la red, donde el control opera sobre ésta y por tanto no es un control distribuido y hace uso de la Abstracción de envío.

En la Figura 2 se puede apreciar la estructura de la versión de SDN. El resultado fue una red más fácil de programar, verificar, y administrar.

Más adelante se dio una transición hacia SDNv2 de estructura similar a la versión previa y la transición surge de hacer una completa separación de funcionalidades. Que dio como resultado tres interfaces:

1. Interfaz de envío (Modelo abstracto de envío), oculta las capas superiores del hardware de envío.
2. Interfaz de distribución (Vista Global de la Red), oculta las capas superiores del estado de difusión.
3. Interfaz de especificación (Vista abstracta de la red), oculta el control de programa de los detalles de la red física.

Lo que se definió en su comienzo con SDN fue que no se necesitaba de la creación de nuevos mecanismos ya que para envío se podía usar MPLS,

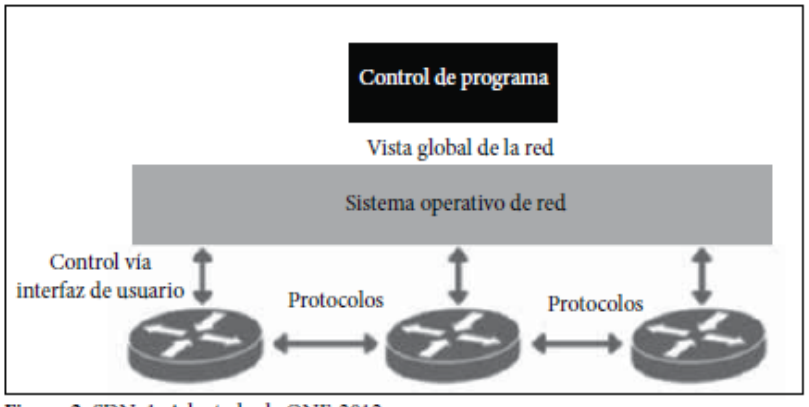

Figura 2. SDNv1. Adaptado de ONF, 2012.

para distribución de estados se podían utilizar protocolos como OSPF y para el control, BGP.

En SDN la idea era agregarle a las redes los beneficios de la programación modular que al igual que con la programación y la abstracción permitiera más flexibilidad y construir funcionalidades más complejas que las infraestructuras actuales con su poca flexibilidad no permitían.

También con SDN se busca poder definir comportamientos que sólo deberían ser especificados en la red abstracta y que no se siguieran diseñando protocolos de control distribuido.

La ONF ha venido haciendo desarrollos en las SDN. Ellos definen a las SDN como *traer programabilidad de software a las redes* (Cariden, 2013). En la SDN se refiere a un control de software lógicamente centralizado.

Existen diferentes clases de SDN(Cariden, 2013):

- FlowServices SDN, referencia una gran cantidad de aplicaciones de seguridad y visibilidad que interactúan con flujos de datos.
- Virtualization SDN, provee conectividad de red virtual para eficiencia y agilidad.
- Infraestructure SDN, provee visibilidad y control de recursos de red y aplicaciones de software.

La clase de interés para este análisis es la Flow Service SDN con interfaz OpenFlow. Esta clase de SDN básicamente permite a un software especializado interactuar con flujos individuales o arreglos de flujos. Por ejemplo en OpenFlow, se definen reglas de flujo, si un paquete cumple con la regla, se realiza una acción. En las SDN el control, el NOS, está separado del plano de desarrollo.

SDN es una arquitectura de red donde el control es desacoplado de la trasmisión y se vuelve directamente programable (ONF, 2012). Este desacoplamiento crea una capa de abstracción entre la infraestructura y las aplicaciones y servicios de red, por lo que la red puede ser tratada como una entidad virtualizada y se vuelve independiente de cada uno de los dispositivos de comunicación que la conformen y de sus fabricantes, que es una ganancia para las empresas. En la Figura 3 se puede apreciar la arquitectura lógica de la SDN (ONF, 2012) y también se puede observar su similitud con SDN v1.

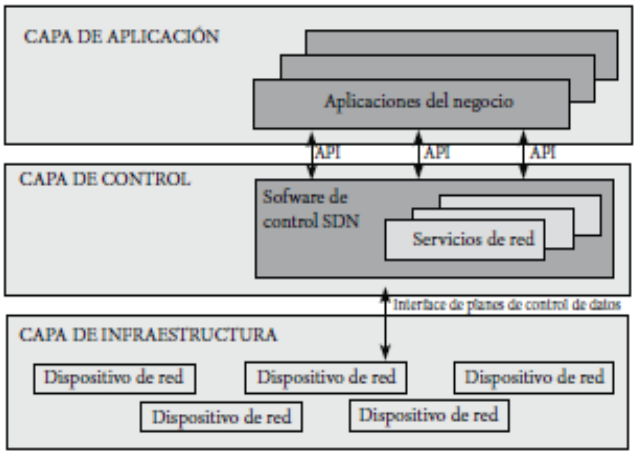

Figura 3. Arquitectura SDN. Adaptado de (ONF, 2012)

En las SDN el control ya no está sujeto a dispositivos individuales de red sino a dispositivos informáticos. La inteligencia de red se centraliza en controladores SDN, los cuales mantienen una visión global de la red y permiten un control centralizado. Por esto, la red es como una entidad lógica y será vista como un único *switch* lógico para las aplicaciones y demás políticas. Al existir esta centralización se tendrá un control total de la red desde un sólo punto lógico lo que simplifica enormemente su diseño y operación, también simplifica los dispositivos que la conforman en sí mismos, pues estos ya no necesitan entender ni procesar estándares de protocolos, sólo necesitan aceptar instrucciones de los controladores SDN (ONF, 2012).

Una característica importante de las SDN es su configuración, pues los operadores de la red pueden configurarla con facilidad ya que su abstracción dispone de un controlador centralizado y no tiene la necesidad de programar varias líneas de código en dispositivos dispersos. Además se puede alterar/configurar el comportamiento de la red en tiempo real e implementar nuevos servicios o nuevas aplicaciones rápidamente, en horas o días y no en meses como sucede con otras arquitecturas. El controlador provee también programas especiales tipo SDN que permiten a los operadores más flexibilidad a la hora de configurar, administrar, proteger y optimizar los recursos de manera dinámica. Estos programas pueden ser escritos por los mismos operadores, por lo que no dependen de vendedores (ONF, 20). Adicionalmente de la abstracción del control de la red, la arquitectura SDN admite un conjunto de API que hacen posible la implementación de servicios de red comunes como enrutamiento, multidifusión, seguridad, control de acceso, gestión de ancho de banda, ingeniería de tráfico, calidad del servicio, uso de energía y demás políticas de gestión de redes.

3. Resultados

OpenFlow- *Vista general*

OpenFlow es un nuevo estándar para control avanzado de redes; de código abierto, fue desarrollado por investigadores de las Universidades de Stanford y California con el fin de estandarizar la comunicación entre *switches* y un controlador basado en software en una arquitectura SDN (McKeown, 2008). OpenFlow se agrega como una función para los *Switches* comerciales permitiendo a sus usuarios llevar a cabo experimentos, sin necesidad de estar ligados directamente al proveedor de los dispositivos de red. OpenFlow está siendo implementado por los principales fabricantes de dispositivos de red del mercado.

El inicio de esta tecnología se dio debido a la imposibilidad de manejar las asignaciones de rutas en Internet y cambiar la infraestructura del modelo OSI de Internet, con la investigación de esta tecnología lo que se pretende es básicamente abrir Internet a los investigadores, hacer conexiones de red por software para tener control sobre el tráfico, lo que permitiría encaminar mejor los paquetes, teniendo mejoras en el desempeño de servicios como video, telefonía móvil, ancho de banda, entre otros.

La puesta en marcha de este proyecto se tornó algo complicado ya que acceder al código fuente del software que controla el hardware de los *switches* comerciales era casi imposible debido a los licenciamientos y privacidad de cada uno de los fabricantes. Se empezó con el desarrollo en *switches* de laboratorio creados por estudiantes y profesores de las universidades de Stanford y California, con el auge del proyecto y lo valioso de sus resultados, grandes compañías como Cisco, Juniper, HP, entre otras, se empezaron a interesar por el proyecto y en la actualidad HP tiene en su lista de productos *switch* que soportan el protocolo OpenFlow. La política principal es que es un estándar de código abierto.

Actualmente se ha probado en el campus de la Universidad de Stanford con un total éxito, se han unido otras nueve universidades a esta investigación tales como la Universidad de California, Universidad de Washington, Universidad de Wisconsin entre otras (Lara, 2013), y se está

probando para crear una red de universidades regidas por esta tecnología.

En cuanto a las versiones de OpenFlow se puede indicar hay disponibles diferentes versiones. La primera versión fue la 0.2.0 lanzada en marzo del 2008, las versiones 0.8.0 y 0.8.1 llegaron en mayo de 2008. La versión 0.8.2 se lanzó en octubre de 2008 adicionando mensajes de *Echo Request* and *Echo Replay*. En diciembre de 2008 se lanzó la versión 0.8.9 que incluía máscaras IP e información estadística adicional. La versión 0.9.0 fue lanzada en julio de 2009 y la versión 1.0 que es la más usada fue lanzada en diciembre de 2009 (Lara, 2013). Las versiones 1.0.0, 1.1.0, 1.2 y 1.3.0 están disponibles. Y en el documento (OpenFlow Switch Specification, 2012) se encuentra de manera detallada los cambios incluidos a cada versión.

Funcionamiento de dispositivos de red

Como se puede apreciar en la Figura 4, los dispositivos de red en la actualidad cuentan con tres planos de operación muy importantes: el plano de control, que son los protocolos de enrutamiento como OSPF, BGP, entre otros, que corren sobre el dispositivo para controlar el re-envío de paquetes. El plano de administración del dispositivo, para su respectiva configuración que se realiza a través de protocolos de administración como SNMP, SSH o XML. Por último se encuentra el plan de reenvío (*Forwarding*) de paquetes el cual será indicado por el plan de control (Protocolo de enrutamiento) que se ejecute sobre el dispositivo.

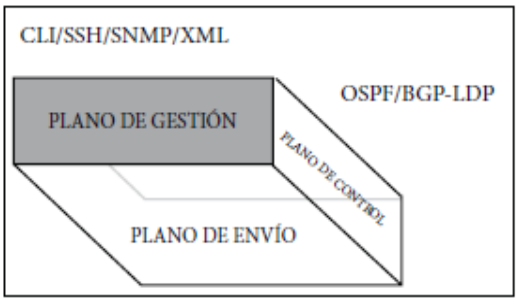

Figura 4. Operación de dispositivos de red actualmente. Adaptado de Ferro, 2012.

Pero, ¿cómo los dispositivos comienzan a intercambiar paquetes? Pues bien, los dispositivos de red, como los *switches* y *routers,* cuentan con tablas de información básicas de los dispositivos, cuya función es encontrar la interfaz adecuada de entrada para que el paquete pueda ser transmitido, conocidas como *Forwarding tables o Forwarding Information Base* (FIB) (Trotter,2001).

Con la información básica en los dispositivos las FIB, aparecen las *routing information base* (RIB), también conocidas como tablas de enrutamiento son las que contienen las listas con las diferentes rutas a los diferentes nodos en una red informática. Para tener una idea más clara de estos dos conceptos de FIB y RIB, los FIB se han optimizado para la búsqueda rápida de direcciones de destino, son usadas principalmente en redes pequeñas y en donde se puede tener un conocimiento uno a uno de todos sus nodos, los RIB se utiliza principalmente para tener conocimiento de todo Internet y saber cómo llegar desde cualquier punto hasta un destino (Berkowitz, 2005).

Con la información de las FIB y RIB en cada uno de los dispositivos de intercomunicación, los protocolos de enrutamiento empiezan a jugar su papel en el conocimiento de las redes, los protocolos empiezan a mapear caminos y a llenar las tablas de enrutamiento de manera dinámica para encontrar los mejores caminos y los más cortos utilizando diferentes algoritmos dependiendo del protocolo.

Con base en esto, el controlador mediante software, se encarga de la administración del hardware de los dispositivos de red, permitiendo constituir una estructura en la cual los FIB sean construidos desde los RIB, y las tablas de información de reenvío se construyan solo con las mejores rutas obtenidas en las tablas de enrutamiento.

Con esta nueva parte en la infraestructura de las redes, el controlador, la arquitectura de los dispositivos de red será mucho más sencilla y se

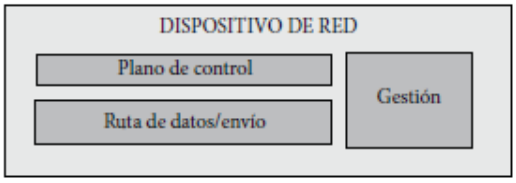

Figura 5. Arquitectura típica de los dispositivos actuales. Adaptado de McKeown, 2008.

podrán construir dispositivos a más bajo costo. En la Figura 5 se puede observar la arquitectura actual de un dispositivo de red, los cuales cuentan con software complejo en el plano de control para la implementación de los protocolos de red y con su administración embebida en el mismo dispositivo.

Con un dispositivo OpenFlow, el sistema operativo es mínimo, solo un *firmware* y un administrador del dispositivo serán incluidos en la nueva

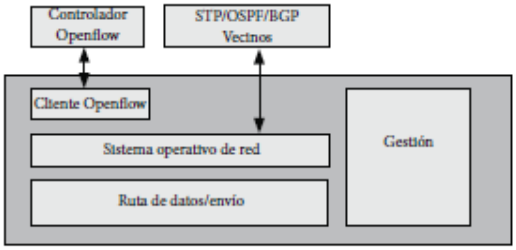

Figura 6. Arquitectura de dispositivo OpenFlow. Adaptado de Ferro, 2012.

arquitectura del dispositivo, como se puede observar en la Figura 6.

Toda la complejidad del software se escala a un nivel superior, al controlador (*Open Flow Controller* en la Figura 6). Los dispositivos OpenFlow también podrán soportar los ya conocidos protocolos de enrutamiento como OSPF, BGP, entre otros. El Network OS o sistema operativo de red será el encargado de la operación de todos los dispositivos operacionales tales como el arranque, la administración de memoria, controlador del protocolo OpenFlow y agente SNMP. Esta nueva arquitectura que es menos compleja, consumirá menos recursos y su costo será menor.

El controlador será el software que brinde conectividad a todos los dispositivos en la red, construirá una topología de la red en memoria, correrá el algoritmo para mapear la red y actualizará las tablas de reenvío con un API, usando el protocolo OpenFlow, que será el que sostenga la comunicación entre el dispositivo y el controlador.

El controlador está conformado por la interfaz de usuario (UI), la cual será el acceso y la administración para los usuarios del controlador. El controlador tendrá un mapa de toda la red. Contará con el protocolo OpenFlow con el cual se comunicará con cada uno de los dispositivos de la red. El controlador podría correr un algoritmo de camino más corto para hacer lo mismo que OSPF y podría implementar BGP para la interoperabilidad con otras redes, como un *router* server.

Funcionamiento

En un *router* o *switch* clásico, la transmisión de paquetes y el enrutamiento se producen en el mismo dispositivo. Lo que se pretende con este nuevo estándar es separar estas dos funciones. La porción del camino continuará residiendo en el *switch*, mientras que las decisiones de alto nivel como el enrutamiento se trasladarán a un controlador independiente.

El estándar de OpenFlow se constituye de tres elementos básicos:

Hardware: Un *switch* que soporte el estándar OpenFlow.

Software (Controlador): Un software que será el controlador de los dispositivos de red, donde el administrador de red define la manera de funcionamiento del dispositivo de red, ya sea como *switch* o como *router*.

Protocolo: El protocolo OpenFlow que se encargará de la comunicación entre el hardware y el software.

El *Switch* de OpenFlow se comunicará con el controlador a través del protocolo OpenFlow, que definirá los mensajes –tales como recepción, envío y modificación de paquetes– y ayudará a obtener estadísticas.

OpenFlow trabaja con base en una tabla de flujos o *flowtable*, la cual contiene las entradas sobre el flujo, la llegada y salida de paquetes en el *Switch*. En la Tabla 1 se puede observar la estructura de una entrada de flujo en una *flowtable*.

Tabla 1. Componentes de una entrada de flujo en una *flowtable*. Tomado de (Openflow, 2011)

MATCH FIELDS	*COUNTERS*	*INSTRUCTIONS*

Cada entrada de la *flowtable* contiene:

Match field: Contiene información tal como puerto de entrada, cabecera de los paquetes, especificaciones de la tabla anterior, entre otros.

Counter: Lleva la cuenta de la actualización de los paquetes

Instrucción: Contiene la acción que se va a realizar, puede un set para cambiar algo, o un borrar la entrada entre otras instrucciones.

El funcionamiento básico de OpenFlow consiste que cuando a un *Switch* llega una entrada que no conoce, se le envía el paquete al controlador mediante el protocolo OpenFlow. El controlador entonces especifica los campos y las acciones que deben ser aplicadas a los paquetes (Palacin, 2009).

Se pueden aplicar diferentes acciones a los paquetes:

- Apply-action acción: Aplica la acción especificada inmediatamente.
- Clear-action acción: Borra todas las acciones establecidas inmediatamente.
- Write-action acción: Combina la acción actual con las ya existentes.

El protocolo OpenFlow es el que describe la entrega de datos entre el controlador y el dispositivo. Conceptualmente, en términos generales, igual a SNMP, utiliza una conexión SSL que le proporciona una comunicación segura entre el controlador y el dispositivo.

Controlador NOX

NOX es la plataforma para crear aplicaciones de control de red y se destina a proporcionar una plataforma de programación para controlar uno o más *switches* OpenFlow. Es una plataforma abierta para el desarrollo de las funciones de gestión de redes empresariales y del hogar. NOX se ejecuta en hardware y proporciona un entorno de software en la parte superior de los programas que pueden controlar grandes redes a velocidad Gigabit.

NOX permite lo siguiente:

- NOX proporciona funcionalidades sofisticadas de red (gestión, visibilidad, control) en *switch* de muy bajo costo.
- Los desarrolladores pueden añadir su propio software de control para el manejo de la red.
- NOX ofrece un modelo de programación central para toda la red. Un programa puede controlar las decisiones de envío en todos los *switches* de la red.

NOX fue el primer controlador de OpenFlow donado a la comunidad científica en 2008 por la compañía Nicira. En una red OpenFlow, NOX controla la red enviándole mensajes al *Switch* de OpenFlow a través del protocolo OpenFlow, que define los mensajes, tales como recepción de paquetes, envío de paquetes, modificación de paquetes y ayudará a obtener estadísticas de la red (Gude, 2008).

NetFPGA

NetFPGA es una plataforma hardware de bajo costo, diseñada por la Universidad de Stanford como una herramienta para el diseño de hardware de redes (*switches* y enrutadores) que permite a los investigadores la construcción de prototipos de trabajo de alta velocidad y sistemas de redes con hardware de alto desempeño.

Las FPGA (*Field Programmable Gate Array*) son dispositivos lógicos de propósito general que se pueden programar por el usuario, compuesto de bloques lógicos comunicados por conexiones programables. Puede realizar funciones tan sencillas como las hechas por una puerta lógica o hasta funciones complejas como las realizadas por un sistema basado en microprocesador.

Las FPGA son utilizadas en aplicaciones hechas a la medida para un uso particular y los requerimientos de un usuario. Tienen la ventaja de ser reprogramables, lo cual añade una enorme flexibilidad al flujo de diseño, sus costes de desarrollo y adquisición son más bajos en comparación con otros dispositivos con características similares y el tiempo de desarrollo también es menor (Watson, 2006).

Entre las aplicaciones más frecuentes que tienen uso las FPGA se encuentran, el procesamiento digital de señales, radio definido por software, sistemas aeroespaciales, sistemas de imágenes para medicina, reconocimiento de voz, bioinformática, entre otras (etFPGA-OneGig, 2012).

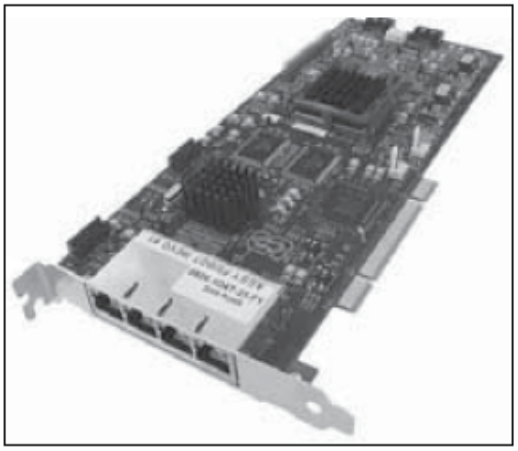

Figura 7. Tarjeta NetFPGA. Tomado de NetFPGA, 2012

La tarjeta de desarrollo de NetFPGA que se utilizó para las pruebas se puede apreciar en la Figura 6 y tiene las siguientes especificaciones: Conector PCI: Bus estándar de 32 bits, 33Mhz, 4 Interfaces GigaBitEthernet (1Gbps), un chip FPGA Virtex-2, SRAM de 4,5 MBytes y DRAM de 64 MBytes (Figura 7).

Ya que la tarjeta NetFPGA cuenta con memoria SRAM y DRAM que son memorias volátiles, al desconectar su alimentación eléctrica perderán la información por lo que la tarjeta quedará desconfigurada y será necesario su reconfiguración (NetFPGA,2012).

La tarjeta tiene tres opciones de uso para implementar un *switch* o un enrutador: una de ellas es descargando las librerías de un *switch* o enrutador básico e instalarlas directamente en la tarjeta, otra descargando las librerías básicas y agregar extensiones (módulos hardware) creados por el usuario y creando un nuevo *router* o *switch* con librerías desarrolladas por el usuario o por el desarrollador (Naous et al., 2008).

Pruebas de laboratorio

Para la realización del montaje de pruebas se utilizó el material la Universidad de Stanford. La cual ya ha realizado con OpenFlow es sus laboratorios diferentes experimentos y montajes. Principalmente se siguió una de sus guías para el montaje de una red OpenFlow con múltiples PC/ NetFPGAs con un controlador NOX en un laboratorio (OpenFlow, 2012); la cual contenía toda la información necesaria para realizar un montaje de manera eficiente y sencilla, indicando requerimientos de software y hardware así mismo como detallando una serie de pasos a seguir. Esta guía se adaptó al Laboratorio de Redes de la Universidad Icesi, donde se realizaron los montajes

y pruebas (Tanenbaum, 2003) que se describen a continuación:

Para el montaje de pruebas, básicamente se realizaron los siguientes procesos:

- Instalación de la tarjeta NetFPGA: Para este montaje se instaló el LiveDVD de Fedora(2012), que trae los paquetes básicos para el funcionamiento de la tarjeta NetFPGA y por medio de comandos se puede comprobar que la instalación esté correcta (NETFPGA, 2011).

- Instalación del módulo OpenFlow sobre el Hardware NetFPGA: Luego de tener todo el hardware configurado y listo para su uso, se instala el módulo OpenFlow para realizar pruebas de capa 2 y capa 3 sobre el protocolo OpenFlow.

- Instalación del software del Controlador NOX para el control de OpenFlow (Noxrepo,2012). Para la instalación del software del controlador se usó el sistema operativo Ubuntu 12.04.

- Comunicación entre el *Switch*OpenFlow y el controlador: Una vez se tienen todos los componentes de la solución listos, tanto el *switch*OpenFlow como el controlador se puede iniciar la comunicación entre estos.

OpenFlow funcionando como switch de capa 2

Después de haber sincronizado correctamente el controlador con el *Switch* se validó el funcionamiento de *Switch* OpenFlow de capa 2. Se configuró la topología que se puede observar en la Figura 8 en el laboratorio de la Universidad Icesi.

Para validar que el *switch* funcione correctamente se utiliza el comando de administración y el monitoreo del *switch* OpenFlow:#dpctl. Con este comando se puede observar el estado actual de los puertos, tabla de flujos, agregar y eliminar flujos, ver la tabla de direcciones MAC, entre otros (Openvswitch, 2012)

Comparación de rendimiento entre Switch OpenFlow y un Switch Cisco 2950

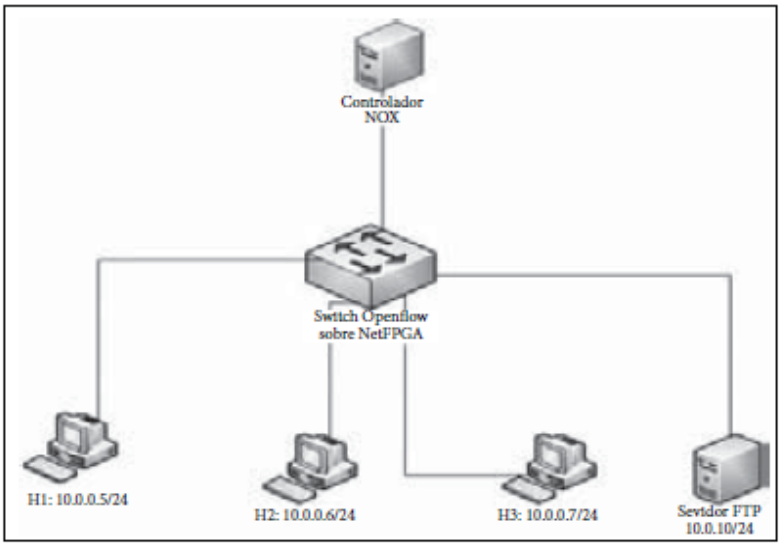

Figura 8. Topología diseñada en el laboratorio para implementar OpenFlow como Switch capa 2.

Para realizar una comparación de rendimiento se siguieron los siguientes pasos: Se realizó un montaje de red igual al de la Figura 7, intercambiando el *Switch* OpenFlow sobre NetFPGA por un *switch* Cisco capa 2, con 2 puertos GB Ethernet, de referencia Catalyst 2950 (Cisco2950, 2012). Para las pruebas se utilizaron paquetes ICMP de solicitud y respuesta mediante el comando PING. Se enviaron paquetes de diferentes tamaños entre el equipo H1 y el equipo H2. El tamaño de los paquetes se varió desde 64 bytes hasta 1500 bytes.

En la Figura 9 se pueden apreciar los resultados obtenidos, mostrando que cuando los paquetes de ping son pequeños el tiempo de envió es similar. Pero a medida que el tamaño de los paquetes aumenta, el *switch* Cisco se demora más en el envió. De esta figura se puede concluir que OpenFlow tiene mejor desempeño que Cisco, ya que el tiempo de envío se mantuvo constante, esto, aclarando que el *switch* Cisco no tenía más equipos conectados, estaba solo dedicado a la transmisión del ping.

Luego de la prueba de envío de paquetes ICMP, se realizó una prueba más rigurosa. Se instaló un Servidor FTP mediante el Software FileZilla FTP

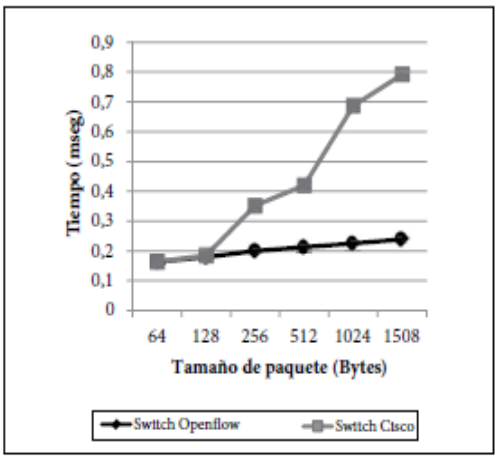

Figura 9. Envío de paquetes ICMP de diferentes tamaños en switch OpenFlow y switch Cisco

Server (FTP Server, 2012) en un equipo del laboratorio, tal como lo muestra la topología de la Figura 7. Se configuró un usuario para tener acceso desde el equipo H3 al servidor (FTP, 2012). Para el equipo H3 y el FileServer se transfirió una imagen de disco ISO de 1GB. El tiempo de esta transferencia en el *switch* Cisco fue de 18 segundos, mientras que utilizando el *switch* OpenFlow fue de 14 segundos. La diferencia fue de 4 segundos,

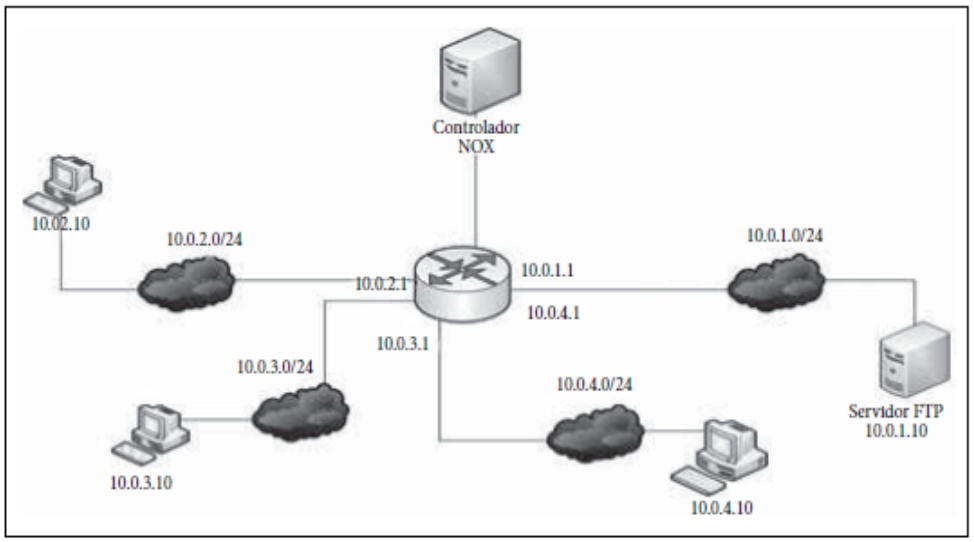

Figura 10. Topología diseñada en el laboratorio para implementar OpenFlow como *Switch* capa 3.

esto se debe a que los algoritmos utilizados en el *Switching* de OpenFlow son mucho más eficientes que los utilizados en Cisco.

OpenFlow funcionando como switch de capa 3

Para configurar OpenFlow como un *Switch* de capa 3 se debe reprogramar la tarjeta y volver a configurar como el módulo *Switch*, pero en esta ocasión se le dirá al *Switch* que también sea un *Router*. En la Figura 10 se muestra la topología utilizada para realizar las pruebas de OpenFlow en capa 3.

Comparación de rendimiento entre *Switch*OpenFlow y un *Router* Cisco 3825

Para realizar una comparación y validar la implementación del protocolo OpenFlow, se realizó un montaje de red mostrado en la Figura 10, para un caso la prueba se realizó sobre la tarjeta Net FPGA configurada como *switch* de capa 3 y para la otra con un *router* Cisco 3825, con 2 puertos GB Ethernet (Cisco3825, 2012). Se transfirió un archivo de 1GB en ambos montajes y utilizando el *Switch* OpenFlow capa 3 la transferencia del archivo tardó 18 segundos mientras que en *Router* Cisco la transferencia del archivo tardó 20 segundos. Se puede apreciar que al igual que con los Montajes con *Switches* capa 2, en capa 3 el *Switch*OpenFlow tiene un menor rendimiento que el *Switch* y *router* Cisco.

4. Conclusiones

Con las pruebas realizadas en el laboratorio de la Universidad Icesi, se puede concluir que OpenFlow se puede convertir en una gran alternativa para el *Switching* en Ethernet, ya que tiene las mismas funciones que un *switch* de capa 2, agregándole un gran desempeño a la transmisión de paquetes. Además de su desempeño como *switch* de capa 2, posee la capacidad de configurarse como *Switch* de capa 3, añadiéndole un excelente desempeño en términos de *throughput* y procesamiento, en comparación con las tecnologías y dispositivos actuales.

Openflow, aparte de su gran desempeño, añade una gran flexibilidad, permite a sus usuarios crear caminos para sus paquetes, al permitir crear reglas que le indiquen a los paquetes por qué puerto deben entrar y por qué puerto deben salir. Permite la fácil administración de una red de comunicaciones; mediante su controlador (software de control NOX), se puede realizar una administración centralizada, no solo de uno, sino de varios dispositivos de red que implementen la tecnología Openflow.

Aparte del desempeño, su flexibilidad y la administración que brinda la tecnología Openflow, tiene un valor agregado y es la seguridad que se puede tener en una red al implementar Openflow. Con reglas, similares a las que se configuran en un firewall, se pueden crear en los *Switch* OpenFlow para evitar el paso de paquetes maliciosos o que provengan de sitios que se han identificado como peligrosos, lo que añade un modulo de seguridad a los dispositivos.

OpenFlow no solo se convierte en una solución para las grandes empresas, sino que también se convierte en una gran posibilidad para que las medianas y pequeñas empresas puedan invertir en tecnología mucho más fácil de administrar y a unos costos mucho más asequibles, ya que con la implementación de un *switch* OpenFlow podrán tener no solo el *switching* de la red sino también funciones de enrutamiento IP.

5. Referencias Bibliográficas

1. Berkowitz H., RFC 4098: Terminology for Berchmarking BGP Device Conveergence in the control plane, 2005.
2. Cariden, "Infrastructure SDN with Cariden Technologies," 15 Agosto 2013. [Online]. Available: http://cdn.sdncentral.com/wp-content/uploads/2012/08/Infrastructure_SDN.pdf. [Accessed 27 Marzo 2013].
3. Casado M., M. J. Freedman, J. Pettit, J. Luo, N. McKeown y S. Shenker, 2008 «Ethane Taking Control of the

4. Cisco 2950, «Switch Cisco 2950,» [En línea]. Available: http://www.cisco.com/en/US/products/hw/switches/ps628/index.html. [Último acceso: 19 Noviembre 2012].

5. Cisco3825, «Router Cisco 3825,» [En línea]. Available: http://www.cisco.com/en/US/products/ps5857/index.html. [Último acceso: 28 Noviembre 2012].

6. Defined Networks,» IEEE Communication Magazine, pp. 36-43, July 2013.

7. Enterprise,» [En línea]. Available: http://www.cs.utexas.edu/users/yzhang/teaching/cs386m-f8/Readings/fp298-casado.pdf. [Último acceso: 3 Mayo 2013].

8. Fedora «Fedora core with netfpga live cd,» [En línea]. Available: http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/LiveDVDInstall. 4.1.1 . [Último acceso: 2 Julio 2012].

9. Ferro G.2012. [En línea]. Available: demo.ipspace.net/get/Software%20Defined%20Networking.pdf. [Último acceso: 25 Septiembre 2012].

10. FTP .«Creación y Gestión de Usuarios de FileZilla FTP Server,» [En línea]. Available: http://www.adslayuda.com/filezilla_ftp_server-usuarios.html. [Último acceso: 17 Noviembre 2012].

11. FTP Server «Manual FileZilla FTP Server,» [En línea]. Available: http://www.adslayuda.com/filezilla_ftp_server.html. [Último acceso: 19 Noviembre 2012].

12. Gude N., B. Pfaff, T. Koponen, M. Casado, S. Shenker, J. Pettit y N. Mckeown. 2008[En línea]. Available: http://www.cs.stonybrook.edu/~vyas/teaching/CSE_592/Fall12/papers/nox.pdf. [Último acceso: 1 Marzo 2012].

13. Lara A. K. a. B. R. A., «Network Innovation using OpenFlow: A Survey.,» IEEE Communications Surveys and Tutorials, Aug 2013.

14. McKeown, T. A. H. G. P. L. P. J. S. S. a. J. T. N. «OpenFlow: enabling innovation in campus networks.,» SIGCOMM Cumpt. Commun. Rev, vol. 38, nº 2, pp. 69-74, 2008.

15. Naous J, D. Erickson, A. Covington, G. Appenzeller y N. Mckeown. 2008 [En línea]. Available: http://yuba.stanford.edu/~jnaous/papers/ancs-openflow-08.pdf. [Último acceso: 20 Abril 2012].

16. NetFPGA . 2012 «NetFPGA,» [En línea]. Available: http://www.netfpga.org/. [Último acceso: 24 Junio 2012].

17. NETFPGA «Verificación de Hardware y Software,» 13 Octubre 2011. [En línea]. Available: http://wiki.netfpga.org/foswiki/bin/view/NetFPGA/OneGig/VerifyHardwareAndSoftware#Run_Selftest. [Último acceso: 17 Noviembre 2012].

18. NetFPGA-OneGig.2012 [En línea]. Available: http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/. [Último acceso: 27 Octubre 2012].

19. Noxrepo [En línea]. Available: https://github.com/noxrepo/nox-classic/wiki/Using-NOX . [Último acceso: 2 Julio 2012].

20. ONF, 2012 «Software-Defined Networking:The New Norm for Networks.,» white paper https://www.opennetworking.org.

21. Open Networking Foundation (ONF) 13 Abril 2012. [En línea]. Available: https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf. [Último acceso: 1 Septiembre 2012].

22. Openflow .28 Febrero 2011. [En línea]. Available: http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf. [Último acceso: 2012 Enero 30].

23. OpenFlow «OpenFlow deployment portal,» [En línea]. Available: http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Deployment/HOWTO/LabSetup. [Último acceso: 2 Julio 2012].

24. OpenFlow Switch Specification. 2012 «OpenFlow Switch Specification, version 1.3.0 (Wire Protocol 0x04),» [On Line] Avaible https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf.

25. Open-Flow-20120512 «Google funciona con Open Flow,» 12 Mayo 2012. [En línea]. Available: http://www.elancasti.com.ar/tecnologia/Google-funciona-con-Open-Flow-20120512-0012.html. [Último acceso: 3 Junio 2013].

26. Openvswitch [En línea]. Available: http://openvswitch.org/cgi-bin/ovsman.cgi?page=utilities%2Fovs-dpctl.8 . [Último acceso: 20 Noviembre 2012].

27. Palacin M. Mateo, OpenFlow Switching: Data Plane Performance, Tesis de Maestría. Torino: Politécnico de Torino. Facultad de Ingeniería. Departamento de Ingeniería de Telecomunicaciones, 2009.

28. Sezer, S. S.-H. P. B. F. D. J. F. N.-. V. M. M. S. «Are We ready for SDN? Implementation Challenges for Software-

29. Shenker S., «Slideshare,» 5 Enero 2011. [En línea]. Available: http://www.slideshare.net/martin_casado/sdn-abstractions. [Último acceso: 1 Mayo 2013].

30. Stanford, «Dashboard,» [En línea]. Available: https://openflow.stanford.edu/dashboard.action. [Último acceso: 3 Junio 2013].

31. Strom D., «arstechnica,» 19 Febrero 2013. [En línea]. Available: http://arstechnica.com/information-technology/2013/02/100gbps-and-beyond-what-lies-ahead-in-the-world-of-networking/2/. [Último acceso: 1 Mayo 2013].

32. Tanenbaum A., Redes de computadoras, México: Pearson Educación, 2003.

33. Trotter G., RFC 3222: Terminology for Forwarding Information Base (FIB) based Router Performance, 2001.

34. Watson G.N. M. a. M. C. 2006 [En línea]. Available: http://groups.csail.mit.edu/cag/warfp2006/submissions/watson-stanford.pdf. [Último acceso: 27 Octubre 2012].