Una Nueva Alternativa a Mininet: Emulación de una Red Definida por Software usando VNX

Lorena Barona, Leonardo Valdivieso
Grupo de Análisis, Seguridad y Sistemas (GASS)
Universidad Complutense de Madrid
Madrid, España
lorebaro@ucm.es
angevald@ucm.es

Danny Guamán.
Facultad de Ingeniería Eléctrica y Electrónica
Escuela Politécnica Nacional
Quito, Ecuador
danny.guaman@epn.edu.ec

Abstract— **Las Redes Definidas por Software (SDN) han introducido un cambio de paradigma en el control y gestión de las comunicaciones, permitiendo el control centralizado de una red desde un dispositivo externo conocido como controlador. Para ello, SDN propone el desacoplamiento del plano de control del plano de datos, estableciendo la comunicación a través de un protocolo, el más conocido OpenFlow. La herramienta más utilizada para experimentación con SDN es el emulador Mininet. No obstante, producto de un proyecto de investigación emprendido por la Universidad Politécnica de Madrid, se ha desarrollado otra herramienta que proporciona soporte para el protocolo OpenFlow, conocida como Virtual Networks over linuX (VNX). En este trabajo se presenta un escenario de red virtual implementado en primera instancia con Mininet y posteriormente con VNX, con el objetivo de contrastar ambas herramientas. Finalmente, se puede constatar que ambas herramientas tienen un desempeño similar, convirtiéndose la integración de VNX y OpenFlow en una nueva opción para experimentación con SDN.**

Keywords- SDN; virtualización; OpenFlow; Mininet; VNX

I. INTRODUCCIÓN

La evolución del Internet y de las tecnologías de la información han marcado no solamente el desarrollo de los medios de comunicación, sino también de aplicativos, dispositivos de red y de la forma de prestación de servicios. Internet pasó de conectar unas pocas computadoras a inicio de los años sesenta, a conectar cada punto del planeta. Sin embargo se continúa utilizando la misma arquitectura de red, que si bien ha funcionado, produce un estancamiento de la misma [1]. El estado de la industria de *networking* para el año 2014 deja como saldo 7000 RFCs, una gran cantidad de protocolos con funciones complejas tal es el caso de los NATs, firewalls, ingeniería de tráfico, servicios diferenciados, que se ven reflejadas en la creación de protocolos específicos, y que dicho sea de paso toman mucho tiempo en ser estandarizados.

Todos estos factores han puesto a trabajar a la comunidad científica en alternativas para mejorar los mecanismos de comunicación que faciliten el despliegue, la experimentación y que aceleren los procesos de estandarización; es así que aparecen las Redes Definidas por Software (*Software Defined Networking*).

Las Redes Definidas por Software tienen sus cimientos en el proyecto *Ethane* [2] realizado en el año 2007. Este proyecto presenta una arquitectura centraliza cuyo control se realiza a través de políticas que permiten mayor granularidad y gestión de la red. El concepto de Redes Definidas por Software propone una arquitectura de red en donde se desacopla el plano de control del plano de datos, permitiendo adaptabilidad, capacidades de automatización y programabilidad de las comunicaciones con un alto grado de eficiencia. Todas estas características permitirán el soporte adecuado para las necesidades actuales [3].

Haciendo una analogía, cuando en la vida real un camión sigue una ruta predefinida hacia un destino, no está exento de problemas de congestión o cierre de la vía, por tanto la solución sería dar un aviso oportuno y proponer una ruta alterna en el momento mismo del incidente. De esta forma se obtiene una mayor eficiencia en cuanto a tiempos de respuesta, retrasos, uso de recursos, etc. SDN se plantea de manera similar, en donde los datos son enviados en forma de *flujos* al controlador, el cual tiene la capacidad de programar el comportamiento de la red a través de software (desvinculación del hardware del software).

Por una parte, los actuales dispositivos de red permiten la reconfiguración de rutas a nivel de hardware, con lo cual las decisiones se toman en cada dispositivo. El control de la red depende del tipo de hardware desplegado, por tanto de la casa fabricante del dispositivo. Por otra parte, SDN proporciona una arquitectura homogénea y modular en donde se controla la conducta de la red a través del *software controlador* que manipula los flujos de la red, sin importar la marca del dispositivo. Para que la comunicación sea óptima debe utilizar un lenguaje común para lo conexión de los dispositivos, tal es el caso del protocolo *OpenFlow* [4], [5].

Existen diferentes herramientas que permiten emular ambientes SDN, tal es el caso de Mininet [6], NS3, VNX/OpenFlow, entre otras. En [7] se presenta el proceso de integración de OpenFlow con la herramienta *Virtual Networks over linuX* (VNX) [8], como una nueva opción para experimentación de esta nueva tecnología. Dentro de este contexto, el presente trabajo presenta VNX/OpenFlow como una nueva alternativa para el despliegue de Redes Definidas por Software. Se realizan pruebas de concepto que permiten constatar el funcionamiento y rendimiento de dicha herramienta, comparada con Mininet.

La memoria se encuentra dividida en cinco secciones, la segunda sección es una introducción de SDN y del protocolo OpenFlow. La tercera sección presenta una reseña del funcionamiento de VNX con soporte OpenFlow. La cuarta sección muestra el escenario propuesto así como las pruebas de concepto realizadas. Finalmente, en la quinta sección se presenta una discusión acerca de SDN.

II. REDES DEFINIDAS POR SOFTWARE

El desarrollo de la tecnología y la masificación de los servicios de Internet han obligado a la comunidad científica a plantearse mecanismos que permitan manejar de forma eficiente las comunicaciones y las necesidades del mercado; para las cuales, la arquitectura de las redes tradicionales no proporciona el rendimiento esperado. Dichas necesidades implican la personalización de los servicios, cambios en los patrones de tráfico, crecimiento de los servicios en *cloud* y requerimientos para *big data*, por mencionar algunos [3]. Como respuesta a estas limitaciones surge el concepto de Redes Definidas por Software. SDN puede ser definida como la tecnología que permite desacoplar el plano de datos del plano de control, en donde la inteligencia de la red reside en un dispositivo externo conocido como controlador. Dicho elemento permite tener un control centralizado de toda la red, proporciona facilidades de programabilidad, automatización y adaptabilidad.

Los primeros despliegues de la tecnología se llevaron a cabo en la Universidad de *Stanford* en el año 2009 y por *Global Environment for Network Innovations* (*GENI*) [9] en el año 2010. GENI juntó a las universidades más importantes de Estados Unidos para realizar investigación, experimentación y despliegue de redes SDN y OpenFlow. Posteriormente, en el año 2011 se crea la organización *Open Networking Foundation* (ONF), la cual tiene como objetivo fundamental la publicación, promoción y adopción de la especificación OpenFlow. Actualmente ONF consta de más de 90 empresas y centros de investigación que ayudan activamente en la adopción de OpenFlow y SDN, dentro de las que se pueden nombrar a empresas como Google, Cisco, HP, entre otras.

SDN es una tecnología muy joven que se encuentra aún en fase de experimentación. Se ha pasado de una estructura totalmente cerrada y limitada a una estructura abierta. El inconveniente principal relacionado con una arquitectura cerrada es la escalabilidad limitada y que la prestación de servicios depende de cada casa fabricante, ya que se requiere esperar la actualización de los sistemas para poder utilizar un nuevo estándar. SDN establece una arquitectura de tres capas, como se muestra en la figura 1.

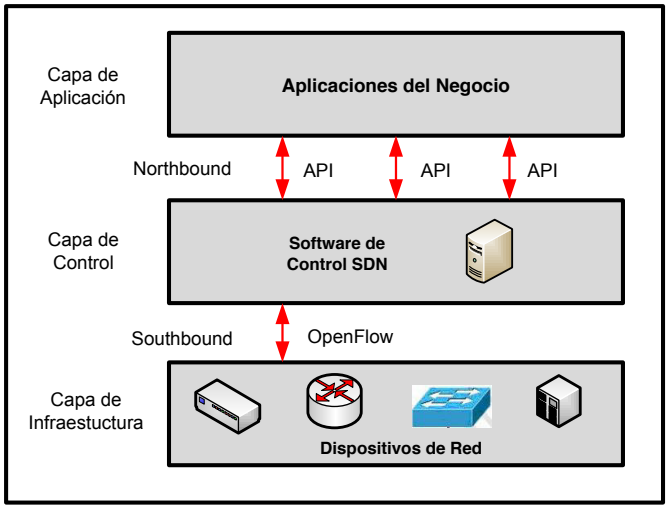

Figure 1. Arquitectura SDN

- La capa de aplicación.- tiene relación directa con las aplicaciones o servicios de alto nivel requeridos por los usuarios (QoS, firewall, etc). Las instrucciones de las aplicaciones son abstraídas y enviadas al controlador mediante el uso de lo que se conoce como interfaz de *northbound*.
- La capa de control.- se encarga de *definir* el comportamiento de los flujos de la red. El controlador permite la transmisión de las instrucciones hacia el switch a través de lo que se conoce como interfaces de *southbound*, la más conocida *OpenFlow*. Esta capa se encuentra ligada con el plano de control.
- La capa de infraestructura.- formada por los dispositivos físicos de comunicación, tal es el caso de switches y routers. Los dispositivos de red administran las tablas de flujo en función de las indicaciones del controlador. Esta capa se encuentra relacionada con el plano de datos.

A continuación se detalla las principales características y el modo de funcionamiento *de OpenFlow*.

A. *OpenFlow*

OpenFlow es el primer protocolo para redes definidas por software, el cual facilita la programabilidad de la red mediante la configuración, gestión y el control de los flujos desde un controlador.

OpenFlow introduce el concepto de flujos (*flow*), que se define como una secuencia de paquetes que atraviesan una red y que comparten campos de su cabecera de datos. En la figura 2 se muestra los componentes y el funcionamiento de un switch OpenFlow [5].

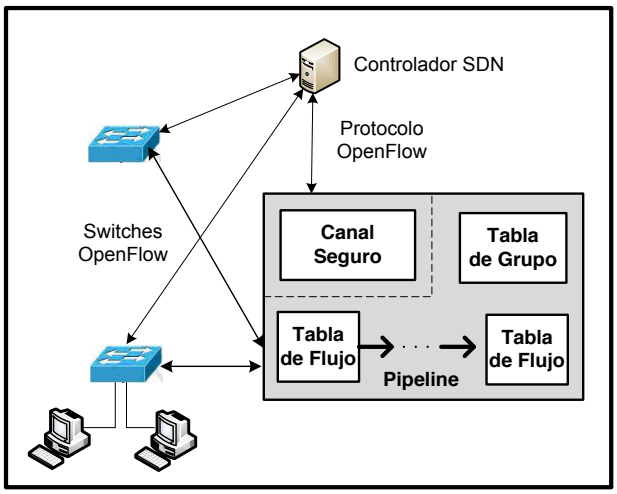

Figure 2. Componentes Switch OpenFlow [10]

- Un controlador que se comunica con los switches y mantiene una visión global de la red. El controlador genera las políticas de tratamiento de la información [11], [12].
- Un canal de comunicación seguro, como *Secure Sockets Layer* (SSL), el cual se encarga de conectar el software de control y el switch.
- Un conjunto de tablas y una tabla de grupos.

OpenFlow se basa en la estructura de un switch Ethernet tradicional, identificando las funciones comunes entre las tablas de diferentes fabricantes y explotando las mismas.

Un switch OpenFlow consta de tres tablas: *flow table*, *group table y meter table*. Un switch *OpenFlow* puede tener una o varias tablas de flujo (*flow tables*) que son organizadas en un *pipeline*, El *pipeline* describe el proceso de decisión y tratamiento de un paquete que ingresa a un switch *OpenFlow*, como se puede visualizar en la Figura 3.

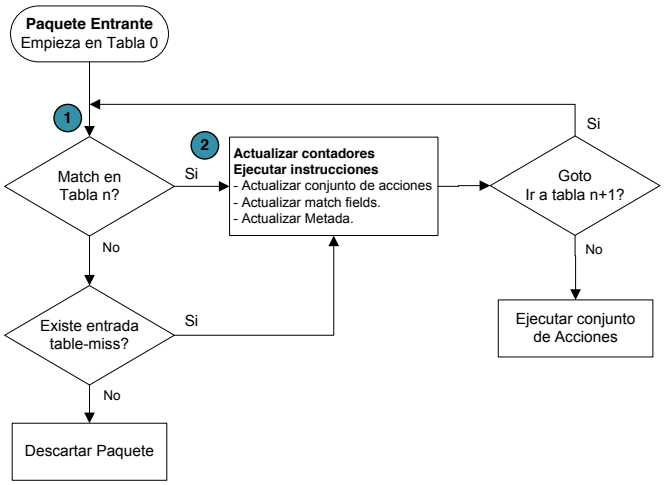

Figure 3. Diagrama de Flujo del Pipeline

Las tablas de flujo son numeradas secuencialmente y el proceso empieza siempre desde la tabla 0. Cuando un paquete ingresa se evalúa si existe coincidencia con alguna de las entradas de la tabla (Paso 1). Existen dos posibilidades:

- Si existe coincidencia.- se actualizarán los contadores y se ejecutarán las instrucciones (Paso 2). Luego se evaluará si se debe ir a la siguiente tabla (n+1). En el caso positivo se regresará al paso 1. En el caso negativo se ejecutarán el conjunto de acciones, terminando así el proceso.
- De no existir coincidencia se evaluará si existe la entrada *table-miss*. En el caso positivo se irá al paso 2. En el caso negativo se descartará el paquete.

La entrada *table-miss* contiene las instrucciones de tratamiento para paquetes que no tienen coincidencia con alguna entrada de la tabla de flujo. Las opciones pueden ser, enviar hacia otra tabla, al controlador, etc. El estándar recomienda que toda las tablas de flujo tengan una entrada *table-miss*. Para poder experimentar con OpenFlow se ha desarrollo el emulador Mininet.

B. Mininet

Mininet es un emulador que permite la rápida creación de escenarios de red virtuales en una sola computadora. Una de las principales características de Mininet es que sus switches soportan el protocolo OpenFlow.

Mininet se basa en las funcionalidades que presta la virtualización de un sistema operativo *Linux*, permitiendo la creación de cientos de nodos. Mininet se ha convertido en la herramienta preferida para emular Redes Definidas por Software debido a su flexibilidad, sus facilidades de despliegue, administración e intercambio de escenarios, interactividad, escalabilidad y ancho de banda para pruebas en el orden de los 2 Gbps, etc.

Para la creación de una nueva topología en Mininet en primera instancia se debe descargar la máquina virtual de la página oficial del proyecto [13]. Posteriormente se podrá interactuar con la misma a través de la interfaz de línea de comandos.

III. VNX/OPENFLOW

Como parte de los proyectos de investigación realizados por la Universidad Politécnica de Madrid, se ha desarrollado la herramienta de código abierto VNX que permite la creación de escenarios de prueba virtuales (*testbeds*) de forma automática. Dentro de las principales características de VNX se puede mencionar:

- Introduce el uso de *libvirt* para la integración de nuevas plataformas de virtualización que a su vez permiten otros sistemas operativos como *Windows*, *FreeBSD*.
- Integra el uso de *Dynamips* y *Olive*, el primero es el emulador de router Cisco y el segundo de Juniper.
- Autoconfiguración y capacidades para ejecución de comandos a través de la creación de imágenes de máquinas virtuales, similares a los archivos de

virtualización *Open Virtualization Format* (OVF). Además se incluye un demonio de autoconfiguración y ejecución de comandos (ACED) que se ejecuta dentro de las propias máquinas virtuales. El demonio ACED realiza la lectura del fichero *XML* y configura las máquinas con esos parámetros.

- Gestión individual de máquinas virtuales.
- Versión para sistemas distribuidos, tiene la capacidad de trabajar sobre un clúster de servidores.

VNX funciona sobre el sistema operativo Linux y presenta una arquitectura modular controlada por un API basada en plugins, como se puede visualizar en la Figura. 4.

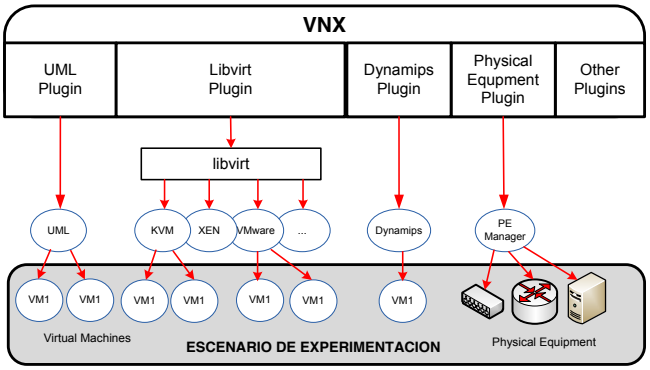

Figure 4. Arquitectura VNX [8]

La principal ventaja de VNX es que permite la creación de múltiples máquinas virtuales desde un solo sistema de archivos (*filesystem*), para lo cual utiliza la técnica conocida como *copy on write* (*cow*). Esta técnica permite que dos o más máquinas virtuales puedan compartir un mismo *filesystem*, el cual se encuentra en modo lectura (*read-only*). Cada vez que una máquina necesita cambiar los parámetros del sistema utiliza un *filesystem* privado donde las diferencias (con el *filesystem* original) son guardadas. En consecuencia permite la reducción de la capacidad de almacenamiento. Otras características de VNX son su estructura modular, permite el control sobre cada máquina virtual, transparencia y su escalabilidad, limitada únicamente por las capacidades de la máquina *host*. El proyecto VNX ha pasado a una etapa de maduración en donde es posible el despliegue de grandes y complejas redes, incluyendo el soporte del protocolo OpenFlow. Para lograr la integración de VNX y OpenFlow se han realizado las siguientes modificaciones:

- Se integró la herramienta *Open vSwitch* (OVS) [14] con la máquina host nativa. OVS permita la generación de switches virtuales con soporte para OpenFlow. VNX utiliza por defecto la utilidad *bridge* del sistema operativo Linux.
- Se modificó el *root filesystem* ubuntu-12.04-gui-v024 el cual se puede descargar desde la página oficial del proyecto VNX. Se agregó tres controladores, NOX [11], POX [12] y Beacon [15] para poder programar el tráfico de la red. Adicionalmente se instaló herramientas de control de tráfico y monitoreo para que sea posible diferenciar el protocolo *OpenFlow*. Estas herramientas fueron el disector para *Wireshark*, el analizador *tcpdump* y la herramienta *iperf* para evaluar el desempeño de las comunicaciones.

Cabe recalcar que para la instalación de VNX es indispensable el uso de un sistema operativo Linux en la máquina *host* nativa. Una vez realizados estos cambios se genera el archivo *.xml* y se despliega los escenarios virtuales. En la figura número 5 se muestra parte del código para la generación del escenario.

```
<vnx xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
  "/usr/share/xml/vnx/vnx-2.00.xsd">
  <global>
    <version>2.0</version>
    <scenario_name>vnx_openflow</scenario_name>
    <automac/>
    <vm_mgmt type="none" />
    <vm_defaults>
        <console id="0" display="no"/>
        <console id="1" display="yes"/>
    </vm_defaults>
  </global>

  <net name="Net0" mode="virtual_bridge" />

  <vm name="C1" type="libvirt" subtype="kvm" os="linux">
    <filesystem type="cow">/usr/share/vnx/filesystems/
    rootfs_ubuntu-gui</filesystem>
    <mem>512M</mem>
    <console id="0" display="yes"/>
    <console id="1" display="no"/>
    <if id="1" net="Net0">
      <ipv4>10.0.0.5/24</ipv4>
    </if>
  </vm>


  <vm name="h1" type="libvirt" subtype="kvm" os="linux">
    <filesystem type="cow">
    /usr/share/vnx/filesystems/rootfs_ubuntu-gui</filesystem>
    <mem>512M</mem>
    <console id="0" display="yes"/>
    <console id="1" display="no"/>
    <if id="1" net="Net0">
      <ipv4>10.0.0.1/24</ipv4>
    </if>
  </vm>

  <vm name="h2" type="libvirt" subtype="kvm" os="linux">
    <filesystem type="cow">
    /usr/share/vnx/filesystems/rootfs_ubuntu</filesystem>
   <mem>128M</mem>
   <if id="2" net="Net1">
      <ipv4>10.0.0.2/24</ipv4>
    </if>
  </vm>

  <host>
    <hostif net="Net0">
       <ipv4>10.0.0.6/24</ipv4>
    </hostif>
  </host>
</vnx>
```

Figure 5. Archivo XML del escenario virtual en VNX

De forma predeterminada, el OVS funciona como un switch Ethernet convencional, por tanto es necesario conectar el OVS con el controlador. La sintaxis del comando usado es la siguiente:

 # ovs-vsctl set-controller name_bridge tcp:IP:port

En donde:

name_bridge representa el nombre del bridge configurado.

tcp:IP:port específica la IP del controlador y el puerto hacia el cual se conectará el bridge. El puerto por defecto es el 6633.

Luego de realizado todo este procedimiento el escenario se encontrará listo para la fase de experimentación y pruebas.

IV. ESCENARIOS DE PRUEBA

Para la fase de pruebas se realiza el despliegue de un escenario tanto en VNX y Mininet. La topología utilizada es similar a la presentada en el *OpenFlow Tutorial* desarrollada por la Universidad de Stanford [16], como se puede visualizar en la Figura. 6.

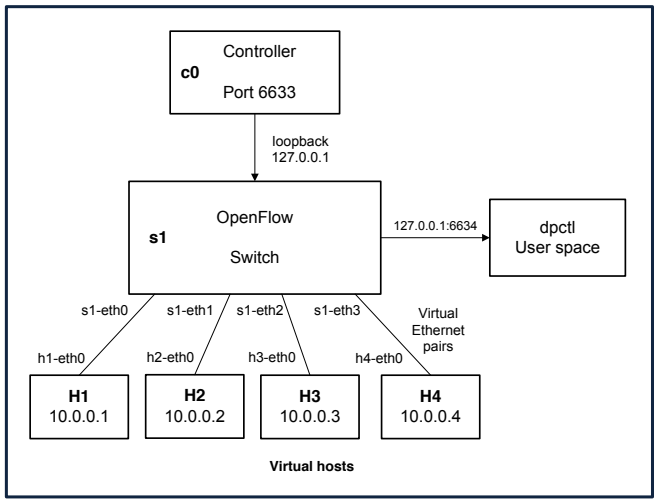

Figure 6. Escenario de Prueba con Mininet

La topología presenta un esquema que conecta 4 hosts virtuales (h1, h2, h3, h4) a una misma subred, la 10.0.0.0/24, a través de un switch (s1). La red será controlada por el Controlador *c0*.

De igual forma para el escenario con VNX se diseño la misma topología como se puede observar en la Figura. 7. Se muestra al controlador (C1) que tiene la dirección 10.0.0.5/24 y la máquina host la IP 10.0.0.6/24. La conexión con la máquina host permite realizar la conexión entre el controlador y el OVS instalado en la misma. Sin esta conexión es imposible la experimentación de OpenFlow con VNX. El controlador C1 y H1 brindan una interfaz gráfica basada en el *root filesystem* ubuntu-12.04-gui-v024 modificado y los hosts H2, H3 y H4 son consolas textuales de Ubuntu 12.04.

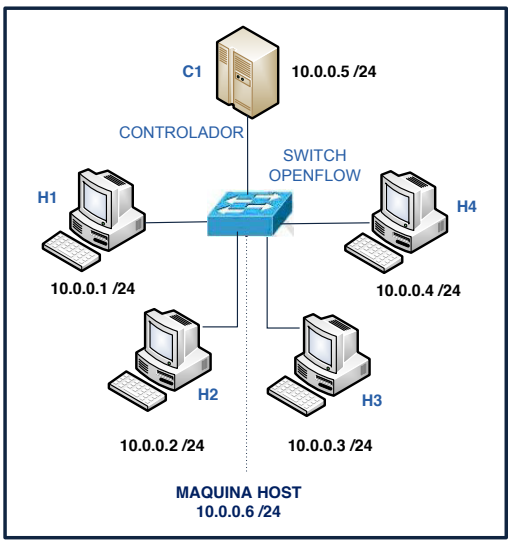

Figure 7. Escenario de Prueba con VNX

Ahora, se realizan algunas pruebas a la par con ambas herramientas para comprobar el desempeño de las mismas. Tanto Mininet como VNX tienen incorporado el software controlador POX, en consecuencia se ha activado el componente forwarding.l2_learning (Figura. 8) el cual permite la emulación de un switch.

```
root@C1:/home/vnx/pox# ./pox.py log.level forwarding.l2_learning
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.1.0 (betta) going up...
DEBUG:core:Running on CPython (2.7.3/Aug 1 2012 05:16:07)
DEBUG:core:Platform is Linux-3.2 generic-i686-with-Ubuntu-12.04
INFO:core:POX 0.1.0 (betta) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[5e-38-fb-61-af-4c 1] connected
```

Figure 8. Activación del Controlador POX

Se realiza una prueba de conectividad desde el host h2 (10.0.0.2/24) hacia el host h1 (10.0.0.1/24) y se activa la herramienta *Wireshark* para verificar el tráfico OpenFlow en los dos emuladores, como se observa en la Figura. 9 (VNX) y Figura. 10 (Mininet).

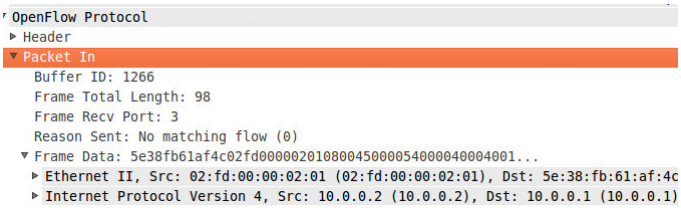

Figure 9. Tráfico OpenFlow en VNX

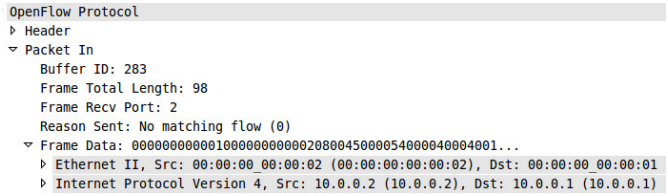

Figure 10. Tráfico OpenFlow en Mininet

En ambos escenarios se puede constatar el mismo comportamiento del tráfico OpenFlow. Se observa un paquete de entrada "*Packet In*", desde la dirección IP de origen 10.0.0.2 hacia la de destino 10.0.0.1. Como respuesta a este requerimiento se generará un paquete "*Packet Out*", el cual indica el tratamiento del paquete, que depende netamente de la programación del controlador.

Otra de las pruebas que se realizan en el OpenFlow Tutorial es la medición del ancho de banda a través de la herramienta *iperf*. De igual forma se realiza un test para ambos escenarios entre el host h1(servidor) y h2(cliente), como se puede observar en la Figura. 11 y Figura. 12.

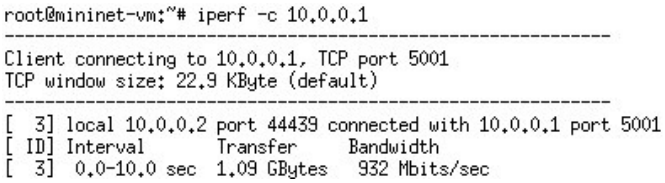

Figure 11. Ancho de Banda en Mininet

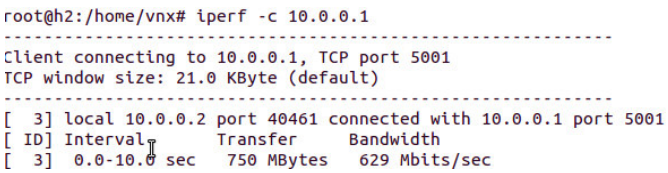

Figure 12. Ancho de Banda en VNX

El ancho de banda para VNX y Mininet es de 932 Mbits/sec y 629 Mbits/sec, respectivamente.

Se realizó pruebas de conectividad y medición de ancho de banda entre todos los hosts de la topología y en ambas herramientas el funcionamiento es idéntico. Dentro de las similitudes encontradas se tiene:

- Permiten la conectividad de un controlador.
- Permiten la captura de tráfico con Wireshark y realizar pruebas de desempeño con Iperf.
- Permiten fácil administración y escalabilidad.
- Interfaz amigable para interacción con los elementos de los escenarios.

Por otra parte existen algunas diferencias entre las dos herramientas, las cuales son:

- VNX no permite la conexión entre switches o en cascada, a diferencia de Mininet que posibilita el fácil despliegue de este tipo de topologías.
- Si el OVS de la máquina host de VNX no funciona, será imposible experimentar con OpenFlow. Mininet no depende de ninguna herramienta o módulo externo, basta con tener la imagen de la máquina virtual.
- Mininet permite el despliegue de escenarios en pocos segundos, lo que no sucede con VNX, que necesita un tiempo de despliegue considerablemente mayor (alrededor de un minuto). Para destruir los escenarios ambas herramientas son casi instantáneas.
- Mininet proporciona una gran cantidad de topologías preestablecidas y que cambian fácilmente, dependiendo del tipo de red que se necesite y del número de host. Para desplegar un nuevo escenario en VNX es indispensable la configuración del archivo *.xml* y realizar el respectivo proceso de despliegue.
- VNX presenta por defecto una interfaz propia para cada elemento de la topología, lo que no sucede con Mininet, que utiliza su línea de comandos para controlar todo el escenario.
- La principal ventaja de VNX sobre Mininet es que permite el uso de diversos sistemas operativos tal es el caso de Windows.

V. DISCUSIÓN

El presente trabajo presenta el despliegue de escenarios para pruebas del protocolo OpenFlow contrastando dos herramientas de emulación de SDN: Mininet y VNX. Ambas herramientas tienen un funcionamiento similar, sin embargo se puede explotar una de las fortalezas de VNX, la posibilidad de utilizar múltiples sistemas operativos, tal es el caso de Windows.

El escenario presentado ha ayudado a probar el funcionamiento del protocolo *OpenFlow*. Gracias a las funcionales de VNX se podrá desplegar grandes redes en función del *root filesystem* modificado y la correcta utilización de OVS.

Las pruebas de concepto realizadas se basaron en la captura del tráfico OpenFlow y en mediciones de ancho de banda, para lo cual se utilizó las herramientas *Wireshark* e *iperf*, respectivamente. Ambas herramientas permiten la experimentación con OpenFlow y arrojan resultados similares trabajando con los mismos recursos del computador. Los resultados obtenidos indican que VNX/OpenFlow es una herramienta que puede ser utilizada para el despliegue de Redes Definidas por Software.

Sin lugar a duda SDN es un tecnología que va ganado espacio por tanto es de vital importancia proveer a la

comunidad de herramientas que permitan el fácil despliegue y experimentación de la misma, tal es el caso de VNX.

RECONOCIMIENTOS

Ángel Leonardo Valdivieso Caraguay y Lorena Isabel Barona López son auspiciados por la Secretaría Nacional de Educación Superior, Ciencia y Tecnología e Innovación SENESCYT (Quito, Ecuador) bajo el Programa de Becas Convocatoria Abierta 2012 y 2013, respectivamente.

REFERENCIAS

[1] B. Heller, R. Sherwood, D. Erickson, H. Shimonishi, S. Seetharaman, and M. McCauley, "*OpenFlowTutorial Open Networking Summit*" http://www.stanford.edu/_brandonh/ONS/OpenFlowTutorial_ONS_Heller.pdf, April 2012.

[2] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking Control of the Enterprise," *in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, (New York, NY, USA), pp. 1–12, ACM, August 2007.

[3] "*The New Norm for Networks*," Open Networking Foundation ONF, April 2012.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM *Computer Communication Review*, vol. 38, pp. 69–74, March 2008.

[5] O. S. Consortium et al., "*OpenFlow Switch Specification v.1.1.3*," pp. 1–128, August 2012.

[6] B. Lantz, B. Heller and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 19.

[7] L. Barona, "Propuesta de Escenarios Virtuales con la Herramienta VNX para Pruebas del Protocolo OpenFlow." 2013.

[8] D. Fernández, A. Cordero, J. Somavilla, J. Rodriguez, A. Corchero, L. Tarrafeta, and F. Galán, "Distributed Virtual Scenarios over multi- Host Linux Environments," in *5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management*, pp. 1–8, IEEE, October 2011.

[9] C. Elliott, "GENI: Opening Up New Classes of Experiments in Global Networking," IEEE Internet Computing, vol. 14, pp. 5–10, January 2010.

[10] W. Stallings, "Software Defined Networks and OpenFlow," The Internet Protocol Journal, vol. 15, pp. 2–14, April 2013.

[11] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 105–110, July 2008.

[12] "POX." [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki.

[13] "An Instant Virtual Network on your Laptop. Mininet" [Online]. Available: http://mininet.github.com.

[14] "Open vSwitch Manual." http://openvswitch.org/ovs-vswitchd.conf.db.5.pdf..

[15] O D. Erickson, "The Beacon Openflow Controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, (New York, NY, USA), pp. 13–18, ACM, August 2013.

[16] "OpenFlow Tutorial." [Online]. Available: http://www.openflow.org/wk/index.php/ OpenFlow Tutorial.

[17] Á. Valdivieso, A. Benito, L. Barona, L. García. Software-Defined Networking: Evolution and Opportunities in the Development IoT Applications. *To be appear in International Journal of Distributed Sensor Networks*, 2014.