Emulación de una red definida por software utilizando MiniNet

Washington A. Velásquez Vargas
Escuela Técnica Superior de Ingenieros en Telecomunicaciones (ETSIT - UPM)
Teléfono: +34603746400
Email: wavelasq@gmail.com

Abstract. *This article shows a recent technology that has by name Software Defined Network, it shows a description about of the networks and how the protocol Openflow is used for the functional correct, these networks have been aceppted by the great industries of market as CISCO and Hewlett-Packard because it shows some advantages such for the client and the company. Also, It used the tool Mininet that is an emulator of network to execute a colection of host, switches, routers and links in a core of linux, this article presents a topology of network for so to observe better the behavior of the Software Defined Network.*

Keywords: *Sotware Defined Nework, Openflow, Switch Openflow, Mininet*

1 Introducción

Últimamente el gran crecimiento de datos en la web ha dado mucho que decir, debido que al transportar una gran cantidad de datos a través de la red, ha producido que el control de los paquetes sea más exigente para evitar colisiones que provocarían cuellos de botella en los centros de datos. La suma de estos factores, junto a otros conceptos de igual pujanza como el Big Data y el Cloud Computing, requiere que los proveedores den servicios con una calidad de conexión sin precedentes, más flexible y sobre todo menos dependiente de la mano del hombre. También exigen un único punto de control para toda la infraestructura, que permita desplegar cualquier aplicación de manera centralizada y así atender con agilidad las necesidades actuales del mercado.

Y es que las redes tal como están configuradas al día de hoy, se presentan un poco obsoletas al procesar toda esta información, porque tienen tendencia a la saturación. Para resolver estas cuestiones al tener que manipular una enorme cantidad de datos, se ha planteado una solución que dependa menos de la intervención del hombre y sobre todo de un hardware tan especializado en el procesamiento de paquetes.

2 Redes definidas por software

La idea de una red que pueda ser definida por software que permita separar el plano de control (software) del de datos (máquina que conmuta los paquetes en la red) y con esto poder obtener redes más programables, automatizables y flexibles, ha logrado tener gran acogida, debido a que se ha tenido una respuesta favorable en la industria permitiendo desarrollar soluciones de código abierto, como lo es Openflow que se define de manera única y común para todos los fabricantes a la hora de extender sus equipos y exponerlos a piezas de software que permitan programarlos.

Por esta razón, para lograr una mejor definición de redes definidas por software debemos compararlas con las redes tradicionales. En las redes actuales, la manera como se procesan los paquetes depende de una programación, mientras que en una SDN (Software Defined Network) está condicionada por una interfaz de programación con un software que gobierna su comportamiento. La manera de procesar los paquetes no depende de una programación estática, que se podría encontrar en unos ficheros de configuración en cada uno de los nodos, sino de los mensajes que envía un software a cada elemento de la red de manera dinámica. “Ésa es la gran diferencia: la red ya no es determinista, sino dinámica”.

El objetivo de las SDN no es otro que armar a los administradores con herramientas centralizadas de programación, virtualización y monitorización de la situación en tiempo real para que las redes se puedan adaptar de forma eficiente a las necesidades concretas de cada momento, lo que promete acelerar la transformación de los datacenters hacia una nueva generación más escalable, automatizar recursos, simplificar tareas y, en definitiva, marcar un antes y un después en el terreno del networking y las comunicaciones. Con SDN el procesamiento de paquetes ya no deriva de una serie de ficheros de configuración estáticos en cada uno de sus miles de nodos, sino de una organización puramente dinámica a través de una capa de software que virtualiza la red y la independiza de la infraestructura física subyacente.

Algo interesante que dijo Ignasi Errando, director técnico de Cisco en España, es que las SDN sustituyen el nivel de control del hardware de red por

una capa de software abstraída mediante técnicas de virtualización, tratando de hacer la red más programable. Esto se concreta en distintas aproximaciones, como son: proporcionar un acceso al hardware basado en programación mediante protocolos como OpenFlow; arquitecturas construidas sobre un nivel de control basado en software; y crear redes virtuales por encima del hardware que dirijan el tráfico a través de redes físicas". [1]

La (figura 1) representa la arquitectura lógica de las redes definidas por software, en donde la inteligencia de la red es lógicamente los controladores de las SDN que mantienen una visión global de la red. Con SDN, empresas y operadores pueden tomar el control sobre toda la red desde un único punto lógico, lo que simplifica en gran medida el diseño de la red.

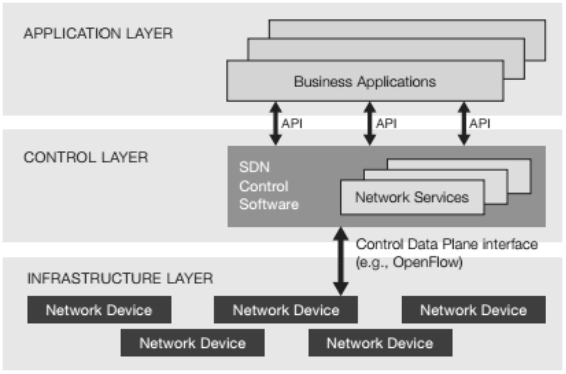

Fig. 1 - Arquitectura de redes definidas por software [3]

Para descifrar y comprender las SDN debemos examinar el interior del software de redes. Una vez que lo comprendamos, podremos definir los principios para solucionar los problemas. [2]

Estos son los seis principios de las SDN y los correspondientes beneficios que ofrecen a los clientes:

- Separan de forma eficiente el software de red en cuatro niveles (planos): administración, servicios, control y reenvío. Esto ofrece el apoyo estructural necesario para optimizar cada plano dentro de la red.

- Centralizan los aspectos necesarios de los planos de administración, servicios y control para simplificar el diseño de red y para reducir los costes operativos.

- Utilizan la nube para una implementación flexible y con una escala adaptable, que permite una estructura de precios basada en el uso con el fin de reducir el tiempo de servicio y establecer una correlación entre costes/valor.

- Crean una plataforma para las aplicaciones de red, los servicios y la integración en los sistemas de administración, lo que permite el desarrollo de nuevas soluciones de negocio.

- Permiten una estandarización de los protocolos, lo que hace posible disponer de asistencia inter-operativa y heterogénea entre proveedores, y que da lugar a la posibilidad de elección y de reducción del coste.

- Aplican con una amplia perspectiva los principios de las SDN a todos los servicios de red, incluidos los de seguridad. Esto abarca desde el centro de datos y el campus empresarial hasta las redes móviles e inalámbricas utilizadas por los proveedores de servicio.

Dentro de cada dispositivo de red y de seguridad (conmutadores, enrutadores y firewall), es posible separar el software en cuatro niveles o planos. A medida que nos movemos hacia las SDN, es necesario diferenciarlos y separarlos con claridad. Se trata de algo esencial para el desarrollo de la siguiente generación de redes altamente escalables, por tal motivo; se presentan cuatro capas que representarían el plano de red de una SDN:

1. ***Forwarding*.** Se encarga del envío de paquetes en la red, la cual está optimizado para mover los datos lo más rápido posible, esta capa puede implementarse en el software, aunque suele desarrollarse sobre circuitos integrados específicos para aplicaciones (ASIC), que están diseñados para este propósito.

2. ***Control*.** Si pensamos en el plano de forwarding como en los músculos de la red, el control sería el cerebro. El plano de control analiza la topología de red y toma decisiones sobre el destino del flujo de la red.

3. ***Servicios*.** Algunas redes requieren un mayor procesamiento, para eso se tiene la capa de servicios, pero no todas los equipos cuentan con esta capa, como por ejemplo los conmutadores. Sin embargo, en muchos enrutadores y firewalls, el plano de servicios se encarga de la toma de decisiones y de llevar a cabo las operaciones complejas con los datos de red

4. ***Administración*.** proporciona las instrucciones básicas sobre cómo debe interactuar el dispositivo de red con el resto de esta.

Por tanto, las SDN representan un gran cambio en la evolución de la red, como el que supuso la sustitución por Ethernet e IP del modelo SNA (System Network Architecture) de IBM, pero con la diferencia de que la industria lo está asumiendo mucho más rápido. Debido a que al mantener la red y SDN libre permite que los desarrolladores puedan operar con total libertad en código abierto y así generar un sin número de soluciones, presentando como una de las principales a Openflow.

3 Openflow

Openflow es un protocolo de comunicaciones diseñado para dirigir el manejo y enrutamiento del tráfico en una red conmutada, hasta la fecha los switches decidían hacia donde llevar el tráfico de manera individual y en base al firmware de cada uno de ellos, eran dependiente del fabricante y dispositivo, con OpenFlow esa decisión dejara de ser tomada de manera individual por cada switch y pasará a estar controlada por una fuerte de inteligencia externa.

Este protocolo ha sido diseñado para que se apoye en tres componentes:

1. Las tablas de flujos instaladas en cada uno de los switches que indicarán a cada dispositivo que hacer con el tráfico.
2. El controlador, que será la "inteligencia" que dialogue con todos los switches y les transmita a estos la información que necesiten.
3. Dispositivos (switches) con soporte para OpenFlow.

Los switches consultaran sus tablas de flujo antes de hacer el forwarding del tráfico y lo realizarán conforme a ellas. Si se da la circunstancia de no saber qué hacer con el tráfico, preguntaran al controlador y este les dirá que hacer, lo que la convertirá en una red gestionada de manera centralizada.

3.1 Switch Openflow

Los switch openflow se los pueden dividir en tres partes, tal cual como se lo observa en la (figura 2), las tablas de flujo, la seguridad del canal y por último el protocolo openflow [4].

Las tablas de flujo indican a los switch como deben proceder los paquetes a través de la red. La seguridad permite conectar a los switch con dispositivos remotos y el protocolo openflow que es un estándar de comunicación abierta entre los controladores y el switch.

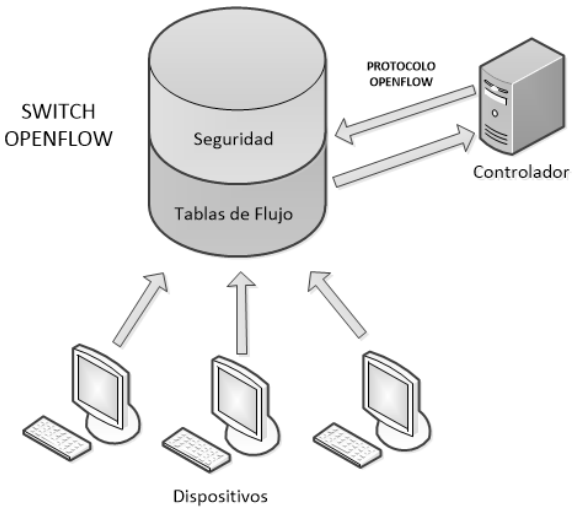

Fig. 2 - Switch openflow

Los switch openflow pueden tener una o varias tablas de flujo, el cual realizan operaciones de búsqueda en las tablas cuando le llega un paquete para que sean enviados a su destino, una entrada de flujo en las tablas tiene los siguientes campos:

- Un paquete cabecera que define el flujo
- Contadores para hacer coincidir los paquetes, estos contadores son actualizados por tablas, flujo, puerto y cola.
- Acciones que se deben realizar para hacer coincidir los paquetes, cada entrada de flujo es asociada a cero o más acciones que son procesadas en un orden especifico, como lo indica la lista que es insertada en cada una de las entradas de flujo.

El conjunto de acciones son apoyadas por los switch openflow debido a su extensibilidad, pero el requisito mínimo para todos los conmutadores es que la ruta de acceso proporcione esta flexibilidad y así poder tener un alto rendimiento con bajo costo.

Una vez que el switch recibe un paquete, se realizan las funciones mostradas en la (figura 3), inicialmente el switch recibe el paquete, luego revisa las tablas de flujo y si la dirección se encuentra en las tablas envía el paquete, caso contrario se lo envía al controlador y este se hace responsable de indicarle al switch que debe hacer con el mismo.

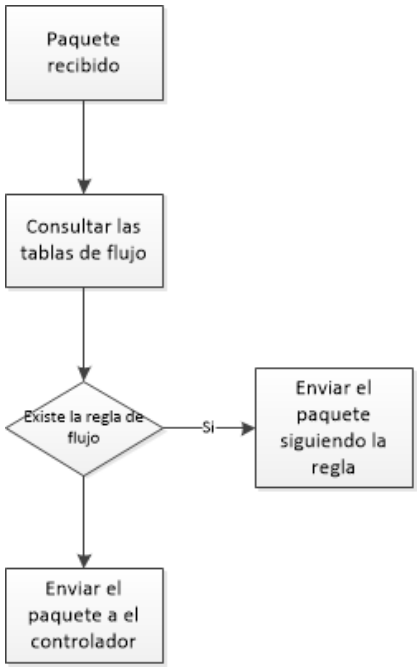

Fig. 3- Diagrama de flujo del Switch Openflow

3.2 Protocolo Openflow

El protocolo openflow permite la comunicación entre el switch y el controlador. Los switch deben ser capaz de establecer comunicación con los controladores, usando una dirección IP y puerto especificado por el usuario. Si el switch conoce la dirección del controlador éste inicia una conexión TCP estándar con el controlador.

Cuando una conexión es establecida por primera vez, cada dispositivo conectado debe enviar mensajes de HELLO con la versión más alta soportada por el protocolo para él envió, aquellos que reciben los mensajes deben tener una versión igual o inferior para poder realizar una conexión, caso contrario se envía un mensaje de ERROR y se termina la conexión [4].

Para una pequeña red de oficina o una red doméstica, lógicamente no tiene sentido incurrir en este nivel de complejidad en el diseño, pero en una red grande, que de soporte a servicios de virtualización (donde un servidor virtual puede cambiar de manera automática de host) o Cloud, SDN y OpenFlow facilitan enormemente la gestión de estas infraestructuras y así mismo, ofrecen grandes beneficios para los clientes, como [3]:

Control centralizado de entornos en múltiples proveedores. El software de control puede controlar cualquier dispositivo de red habilitado para OpenFlow de cualquier proveedor, incluyendo switches, routers y switches virtuales. En lugar de tener que gestionar grupos de dispositivos de vendedores individuales.

Reducción de la complejidad a través de la automatización. Ofertas SDN basados en OpenFlow como un marco de automatización y de gestión de redes flexibles, lo que hace posible el desarrollo de herramientas que automatizan muchas tareas de administración que se realizan manualmente. Estas herramientas de automatización reducen gastos generales de operatividad y disminuyen la inestabilidad de la red introducida por error del operador.

El aumento de la fiabilidad y la seguridad de la red, SDN hace posible que TI pueda definir sentencias de configuración y políticas de alto nivel, que luego serán traducidas a la infraestructura a través de OpenFlow. Debido a que elimina la necesidad de configurar de forma individual la red cada vez que un nodo, un servicio o aplicación se añade o se mueve, lo que reduce la probabilidad de fallas en la red debido a la configuración o inconsistencias de políticas.

Mejoramiento en las experiencias del usuario, Al centralizar el control de la red y que la información esté disponible para las aplicaciones de alto nivel, la infraestructura de una SDN puede tener mejor adaptación a las necesidades del usuario.. Por ejemplo, un proveedor podría introducir un servicio de vídeo que ofrece a los suscriptores premium la más alta resolución posible de una manera automatizada y transparente. Hoy en día, los usuarios deben seleccionar explícitamente un ajuste de resolución, por tanto, cuando se tengan dispositivos con soporte para OpenFlow y un controlador que los dirija, podremos tomar de manera centralizada decisiones sobre QoS, rutas para reducir la latencia, velocidad, etc.

3.3 Controladores

Hasta ahora conocemos los beneficios de openflow, pero, ¿En qué se basa la eficiencia de este protocolo?, la respuesta está en el uso de controladores que se encargan de centralizar y transmitir la información a la red y manteniendo un paso distribuido a través de los switch y routers. Estos controladores en la red van ubicados dependiendo de cómo estén los switch conectados, se podrían identificar dos formas de configuración, una de ellas es que debería haber un controlador centralizado para que administre y configure todos los dispositivos de la red y la otra que los controladores se repartan de manera distribuida, es decir, un controlador para cada conjunto de switch.

El objetivo de un controlador es redirigir ICMP o paquetes de Ethernet enviados entre los dispositivos

finales sin importar el tipo de controlador que se tenga, pero si es importante saber cómo la red debe tomar estas decisiones, los posibles modos de operación de un controlador son los siguientes:

Centralizado. Solo se necesita un controlador para manejar los switch, es decir; un solo controlador maneja todos los dispositivos de la red.

Por flujo. Cada entrada es configurada por el controlador y así mismo, las tablas contienen las nuevas líneas del flujo.

Reactivos. El controlador está diseñado inicialmente para no hacer nada hasta que reciba un mensaje, sean estos: Solicitudes de ping, mensajes RST (Ping Request Message) o introducir una regla en las tablas de flujo para así ser enviados al destino. A este método se lo conoce como reactivo, porque una vez que se procese el mensaje, el controlador usará una regla específica para enviar el paquete al destino.

Proactivos. El controlador anticipa el mensaje de respuesta, el mensaje Ping Responder, y finalmente antes de recibir el mensaje Ping Reply, el controlador inserta la regla en las tablas de flujo del switch. Entonces, cuando el switch recibe el mensaje Ping Reply, el switch ya tiene la regla para envía el paquete.

En la (figura 4) se presenta un esquema general de como los controladores deben ir conectados en la red.

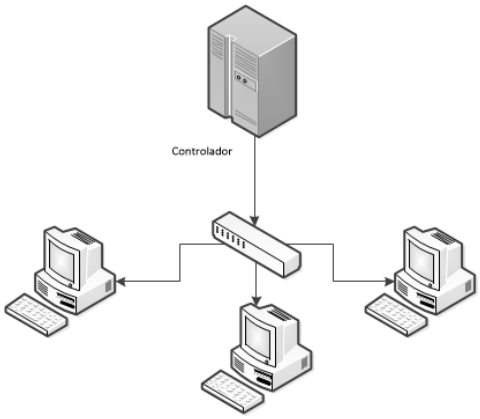

Fig. 4 - Esquema básico de un controlador en la red

Un receptor de paquetes cumple las funciones mostradas en la (figura 5), Si el switch no sabe qué acción realizar con un paquete en específico, lo pasa al controlador. El controlador recibirá el paquete y revisara si este es un paquete Ethernet, hay dos casos posibles:

- Si el paquete no es uno de Ethernet, el controlador lo descarta.
- Si el paquete es Ethernet, el controlador analizara las cabeceras del paquete para obtener la dirección de destino (dirección MAC y dirección IP). Después, el controlador inserta las reglas necesarias en los switch para que pueda ser enrutado al destino.

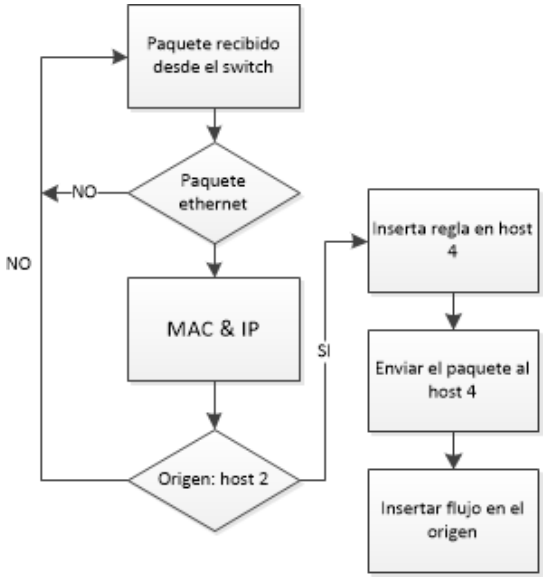

Fig. 5 - Diagrama de flujo de las funciones de un Controlador

4 MiniNet

MiniNet es un emulador de red que ejecuta una colección de dispositivos finales, switches, routers y enlaces en un solo core de Linux. Se utiliza la virtualización ligera para hacer que un solo sistema parezca una red completa. Los programas que se ejecutan pueden enviar paquetes a través de lo que parece ser una interfaz de Ethernet real, con una velocidad de enlace y con retardo. Los paquetes se procesan por lo que parece un verdadero interruptor de Ethernet, un enrutador o middlebox, con una determinada cantidad de colas. Cuando dos programas, como un iperf cliente y el servidor, se comunican a través MiniNet, el rendimiento medido debe coincidir con el de dos máquinas nativas más lentas.

En resumen, los dispositivos virtuales de MiniNet, conmutadores, enlaces y los controladores se crean utilizando software en lugar de hardware, en su mayor pate el comportamiento es similar a los elementos de hardware. La eficiencia de mininet permite que con un solo comando se ejecute la red [6], tal como se muestra en la (figura 6):

Fig. 6 - Comando para habilitar MiniNet

Es fácil interactuar con MiniNet mediante el api que ofrece el software, debido a que el uso está destinado

para el desarrollo, la enseñanza y la investigación. En este artículo nos ayuda a simular una red definida por software y experimentar con Openflow. Actualmente Mininet se está desarrollando de manera activa y se distribuye bajo licencia BSD de código abierto permisivo. La manera más fácil de empezar a utilizar MiniNet es descargar la máquina virtual que ofrece la página oficial, pero si desea instalarlo por su propia cuenta debe clonar el código fuente de la dirección que le proporciona el manual de instalación y seguir los pasos del mismo.

4.1 Ventajas de Mininet

Mininet presenta un sin número de ventajas al querer simular una red, la cual se presentan a continuación:

- Es **rápido** – la puesta en marcha de una red simple tarda sólo unos segundos. Esto significa que el bucle de gestión edit-debug puede ser muy rápido.

- Puede **crear topologías personalizadas**: un solo switch, grandes topologías tipo Internet, la red troncal de Stanford, un centro de datos, o cualquier otra cosa de redes.

- **Ejecutar programas reales**: cualquier cosa que se ejecute en Linux está disponible para que funcione, desde los servidores de Internet hasta las herramientas de TCP para el monitoreo de paquetes.

- **Personalizar el reenvío de paquetes**: los switch de MiniNet son programables usando el protocolo OpenFlow, lo que significa que los diseños de red se pueden transferir fácilmente a los switches OpenFlow.

- Puede ejecutar MiniNet en su computadora portátil, en un servidor, en una máquina virtual, en una máquina Linux nativo (MiniNet se incluye con Ubuntu 12.10), o en la nube (por ejemplo, Amazon EC2)

- Puede **compartir y replicar los resultados**: cualquiera con una computadora puede ejecutar el código una vez que se han instalado los paquetes necesarios para su funcionamiento.

- Se puede **crear y ejecutar** experimentos MiniNet escribiendo simples (o complejos de ser necesario) scripts de Python.

4.2 Limitaciones

- Correr en un solo sistema es conveniente, pero también impone límites de recursos: si su servidor tiene 3 GHz de CPU y puede cambiar alrededor de 3 Gbps de tráfico simulado, tendrán que ser equilibradas y compartidas entre los hosts virtuales y switches.

- MiniNet utiliza un solo núcleo de Linux para todos los hosts virtuales, lo que significa que no se puede ejecutar el software que depende de BSD, Windows, u otros núcleos del sistema operativo. (Aunque se puede conectar máquinas virtuales a MiniNet.)

- MiniNet no escribirá su controlador OpenFlow para usted, si usted necesita enrutamiento personalizado o el comportamiento de conmutación, usted tendrá que encontrar o desarrollar un controlador con las características que usted requiere.

- Actualmente MiniNet no hace NAT fuera de la caja. Esto significa que sus máquinas virtuales se pueden aislar de la red LAN de forma predeterminada, lo que suele ser bueno, pero significa que sus anfitriones no pueden hablar directamente a Internet a menos que proporcione un medio para que lo hagan. (el ejemplo nat.py muestra cómo configurar hosts MiniNet para la conectividad externa, y los ejemplos hwintf.py muestran cómo agregar una interfaz física para MiniNet.).

- Todos los hosts MiniNet comparten el sistema de archivos y el espacio PID, lo que significa que es posible que tenga que tener cuidado si se está ejecutando demonios que requieren de configuración en /etc, y hay que tener cuidado de que no se mata a los procesos erróneos por error. (Tenga en cuenta el ejemplo bind.py que demuestra cómo tener directorios privados por el huésped.)

- A diferencia de un simulador, MiniNet no tiene una fuerte noción de tiempo virtual, lo que significa que las medidas de temporización se basarán en tiempo real, y que los resultados serán en tiempo real (por ejemplo, redes a 100 Gbps) no pueden ser fácilmente emuladas.

Lo más importante que se tiene que tener en cuenta para los experimentos de red limitada es que probablemente tendrá que utilizar los enlaces más lentos, por ejemplo 10 o 100 Mb/s en lugar de 10 Gb/s, debido al hecho de que los paquetes son transmitidos por un conjunto de conmutadores de software, los recursos de la CPU que comparten la memoria, y por lo general tienen un menor rendimiento que el hardware de conmutación dedicado. Para los experimentos limitados de CPU, también tendrá que asegurarse de que se limite cuidadosamente el ancho de banda de la CPU de sus

dispositivos MiniNet. Si usted se preocupa principalmente de la corrección funcional, puede ejecutar MiniNet sin límites específicos del ancho de banda - Esta es la manera rápida y fácil de ejecutar MiniNet, ya que también proporciona el más alto rendimiento a expensas de la precisión de sincronización de carga.

Antes de presentar la topología simulada en MiniNet se presentara los comandos básicos para el correcto uso de la herramienta. El siguiente comando muestra la ayuda que describe las opciones de inicio de MiniNet: *sudo mn-h.*

4.3 Interactuar con los Host y Switch

Puede interactuar con mininet, iniciando una topología mínima utilizando el comando: *sudo mn*, la red predeterminada que se cargará consta de un controlador que administra un switch conectado a dos host. A continuación se presentan los comandos básicos:

Help: Muestra los comandos de MiniNet.

Nodes: Muestra los nodos de la red

Net: Muestra los enlaces que se tiene en la red.

Ifconfig-a: Muestra la configuración IP del dispositivo que se desea consultar, puede ser un host o switch. El uso es de la siguiente manera: *h1 ifconfig-a*, donde *h1* representa el host 1 en la red.

Ps-a: Muestra en pantalla los procesos que se están ejecutando en la máquina, la forma de usarlo es: *h1 ps-a,* donde h1 es el host 1 en la red. Se debe recordar que los procesos a mostrar serán los mismos tanto en los host como en los switch, debido a que se está virtualizando la red más no se tiene máquinas individuales.

Ping -c: Permite hacer ping entre los dispositivos de la red, la manera de uso es la siguiente: *h1 ping-c 1 h2*. El host 1 está haciendo ping al host 2, pero solo le está enviando un mensaje, la cantidad de mensajes puede ser configurada indicando el número de mensajes después del comando *ping -c.*

4.4 Topología de la simulación

En esta sección, se muestra el diseño de la red simulada, tal como se observa en la (figura 7), en el gráfico se puede observar claramente una estructura con un controlador centralizado, encargado de reenviar los paquetes a los switch y estos a su vez transmitirlos a los dispositivos finales.

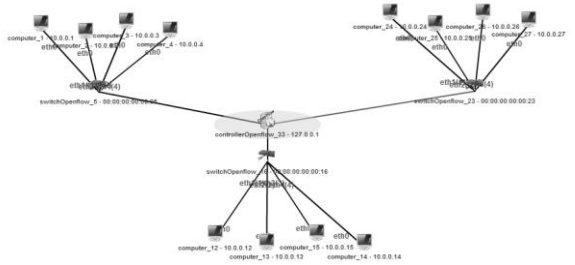

Fig. 7 - Topología de la red simulada

El código esta implementado en Python, siguiendo el API que ofrece mininet para su desarrollo, el código se o puede observar en mejor detalle en el apéndice I.

El código es muy fácil de entender, se tienen 12 host, 3 switch y 1 controlador. Los host están repartidos en cuatro por cada switch, switch 1 (s5) tiene los host (h1, h2, h3, h4), switch 2 (s16) tiene los host (h14, h15, h13, h12), switch 3 (s23) tiene los host (h24, h25, h26, h27) y controlador administra los tres switch. Primeramente importamos las librerías de mininet para usar el api, luego definimos una topología, es decir; instanciamos un objeto "Mininet". Creamos los nodos, todos los host con su dirección ip, los switch y el controlador, luego agregamos las relaciones entre ellos y finalmente ejecutamos el controlador.

Al ejecutar el archivo *mininetTopologia.sh* se crean en la maquina las interfaces virtuales y al mismo tiempo compila el código de la simulación, en la (figura 8) se muestra la ejecución del comando net:

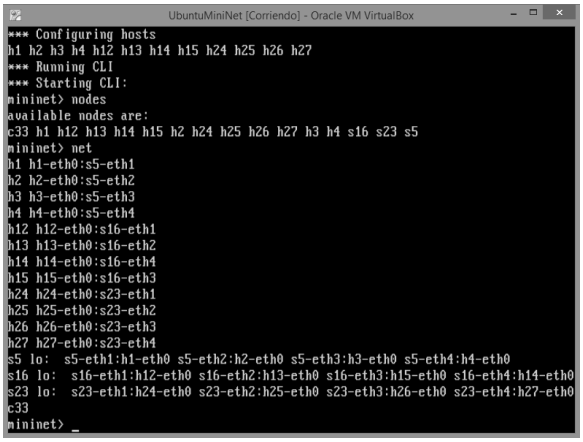

Fig. 8 - Muestra de la red simulada

De esta manera podemos interactuar con la red, utilizando los comandos anteriormente descritos, podremos observar el comportamiento de la red.

5 Conclusiones

Tendencias como la movilidad del usuario, la virtualización de servidores, IT- as-a -Service, y la necesidad para responder a las cambiantes condiciones de negocios, significan demandas sobre las redes. Las redes definidas por Software

proporcionan una nueva arquitectura, red dinámica que transforma redes tradicionales en ricas plataformas que ofrecen grandes prestaciones de servicios.

Al desacoplar el control de la red y los planos de datos permite que OpenFlow, que está basado en la arquitectura SDN abstraiga la infraestructura subyacente de las aplicaciones que lo utilizan, lo que permite a la red llegar a ser tan programable y manejable a escala de la infraestructura informática.

Las SDN suponen un gran cambio en los sectores de redes y de seguridad. Su impacto va a llegar más allá del centro de datos y será mucho más amplio de lo que muchos prevén en la actualidad. Las SDN darán lugar a nuevos ganadores y perdedores. Podremos ver nuevas empresas que emergerán con éxito y a otras que no podrán superar esta transición. En cualquier caso, y al igual que con cualquier otra tendencia del sector, las ventajas para el cliente son reales, como también el cambio tecnológico es inevitable.

Apéndice I

Código en phyton que se implementó para realizar la simulación de la topología mostrada en la fig. 7.

```
#!/usr/bin/python
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch, OVSLegacyKernelSwitch, UserSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink


def topology():
   "Create a network."
net = Mininet( controller=RemoteController, link=TCLink, switch=OVSKernelSwitch )

print "*** Creating nodes"
h1 = net.addHost( 'h1', mac='00:00:00:00:00:01', ip='10.0.0.1/8' )
h2 = net.addHost( 'h2', mac='00:00:00:00:00:02', ip='10.0.0.2/8' )
h3 = net.addHost( 'h3', mac='00:00:00:00:00:03', ip='10.0.0.3/8' )
h4 = net.addHost( 'h4', mac='00:00:00:00:00:04', ip='10.0.0.4/8' )
s5 = net.addSwitch( 's5' )
h12 = net.addHost( 'h12', mac='00:00:00:00:00:12', ip='10.0.0.12/8' )
h13 = net.addHost( 'h13', mac='00:00:00:00:00:13', ip='10.0.0.13/8' )
h14 = net.addHost( 'h14', mac='00:00:00:00:00:14', ip='10.0.0.14/8' )
h15 = net.addHost( 'h15', mac='00:00:00:00:00:15', ip='10.0.0.15/8' )
s16 = net.addSwitch( 's16' )
s23 = net.addSwitch( 's23' )
h24 = net.addHost( 'h24', mac='00:00:00:00:00:24', ip='10.0.0.24/8' )
h25 = net.addHost( 'h25', mac='00:00:00:00:00:25', ip='10.0.0.25/8' )
h26 = net.addHost( 'h26', mac='00:00:00:00:00:26', ip='10.0.0.26/8' )
h27 = net.addHost( 'h27', mac='00:00:00:00:00:27', ip='10.0.0.27/8' )
c33 = net.addController( 'c33', controller=RemoteController, ip='127.0.0.1', port=6633 )

print "*** Creating links"
net.addLink(h27, s23, 0, 4)
net.addLink(h26, s23, 0, 3)
net.addLink(h25, s23, 0, 2)
net.addLink(h24, s23, 0, 1)
net.addLink(h14, s16, 0, 4)
net.addLink(h15, s16, 0, 3)
net.addLink(h13, s16, 0, 2)
net.addLink(h12, s16, 0, 1)
net.addLink(h4, s5, 0, 4)
net.addLink(h3, s5, 0, 3)
net.addLink(h2, s5, 0, 2)
net.addLink(h1, s5, 0, 1)
print "*** Starting network"
net.build()
s23.start( [c33] )
s16.start( [c33] )
s5.start( [c33] )
c33.start()
print "*** Running CLI"
CLI( net )
print "*** Stopping network"
net.stop()
if __name__ == '__main__':
   setLogLevel( 'info' )
topology()
```

Referencias

[1] Data Center Dinamics - Focus , Edición Nº 12 página 26 – 27, Revista Internacional de diseño y gestión de centro de datos, Enero 2013 http://content.yudu.com/Library/A20af7/DataCentreDynamicsIs/resources/27.htm

[2] Juniper Networks, Inc.1194 North Mathilda Avenue Sunnyvale, CA 94089 EE. UU. http://www.juniper.net/es/es/local/pdf/whitepapers/sdn-whitepaper.pdf

[3] Open Networking Foundation / 2275 E. Bayshore Road, Suite 103 / Palo Alto, CA 94303 / www.opennetworking.org https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[4] The Wi-Fi Alliance: www.wirelessethernet.com http://www.valleytalk.org/wp-content/uploads/2013/02/Evaluation_Of_OF_Controllers.pdf

[5] The OpenFlow Switch Specication. Available at: http://OpenFlowSwitch.org

[6] MiniNet Open Source http://mininet.org/

[7] Lantz, B., Heller, B., & McKeown, N. (2010, October). A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (p. 19). ACM.

[8] Drutskoy, D., Keller, E., & Rexford, J. (2013). Scalable network virtualization in software-defined networks. IEEE Internet Computing, 17(2), 20-27. doi:10.1109/MIC.2012.144