TELECOMUNICACIONES Y ELECTRONICA

Escenarios SDN en Mininet

SDN scenarios in Mininet

Autores: Ing. Yudisleidy Ruiz Rios, MSc. Ing. Arelys E. Ramos Fleites, Ing. Daniel Darián Iglesias de la Torre, DrC. Ing. Félix Álvarez Paliza

Universidad Central de las Villas. Villa Clara. Cuba, yrrios@uclv.edu.cu
Universidad Central de las Villas. Villa Clara. Cuba, arelys@uclv.edu.cu
Universidad Central de las Villas. Villa Clara. Cuba, danielit@uclv.cu
Universidad Central de las Villas. Villa Clara. Cuba, fapaliza@uclv.edu.cu

Resumen

El crecimiento de los dispositivos móviles, las redes sociales, la computación en la nube y muchos otros servicios, han dado como resultado el aumento exponencial del tráfico que circula por la red y los centros de datos. Surge así el concepto de las Redes Definidas por Software y el protocolo OpenFlow, que está revolucionando el campo de las tecnologías de la información mediante la separación del plano de control del plano de datos. El presente trabajo se centra en el desarrollo de escenarios SDN mediante el uso de la herramienta de emulación Mininet. Para ello fueron analizados los fundamentos teóricos de este tipo de redes, del protocolo OpenFlow, los controladores disponibles en la industria y los principales simuladores y/o emuladores existentes. Se hace una propuesta de escenarios típicos de SDN aplicando temas sobre la creación de topologías, tráfico cliente-servidor, mediciones de ancho de banda y análisis de paquetes.

Abstract: *The growth of mobile devices, social networks, cloud computing and many other services have resulted in an exponential increase in traffic circulating through the network and data centers. Thus arose the concept of Software Defined Networks and the Open Flow protocol, which is revolutionizing the field of information technologies by separating the plane of control of the data plane. The present work focuses on the development of SDN scenarios through the use of the Mininet emulation tool. The*

theoretical foundations of this type of networks, the OpenFlow protocol, the controllers available in the industry and the main existing simulators and / or emulators were analyzed. A proposal of typical SDN scenarios is made applying topics on the creation of topologies, client-server traffic, bandwidth measurements and package analysis.

Palabras Clave: SDN; OpenFlow; Mininet; Controlador; Tráfico

Keywords: *SDN; OpenFlow; Mininet; Controller; Traffic*

1. Introducción

Hace años se viene generando un crecimiento fuerte y sostenido de las redes de telecomunicaciones, resultado de una demanda cada vez mayor por parte de los usuarios. Este crecimiento, desordenado por basarse en principios y modelos no escalables, demostró que las infraestructuras y arquitecturas tradicionales de redes presentan serias limitaciones de complejidad, adaptación, flexibilidad, escalabilidad, políticas inconsistentes: por lo que se ha hecho necesario realizar un cambio de enfoque sobre las arquitecturas de redes, para lograr un control y administración centralizados poniendo la red al servicio de los operadores y programadores. Junto a ello, se busca acompañar el creciente despliegue de servicios de tipo "cloud computing" y la virtualización de las funciones de red. Se considera entonces a SDN como un nuevo paradigma de red que brinda la habilidad de programar directamente operaciones de red, usando lenguajes y sistemas operativos estándar separados de los dispositivos de red , separa el plano de datos o infraestructura del plano de control que se encarga de convertir las solicitudes de las aplicaciones SDN en órdenes hacia los dispositivos físicos de la red en cuanto a cómo los paquetes deberán ser reenviados a través de la red, así como de la programación de los nodos para implementar la decisión tomada [1], [2]. De las anteriores consideraciones teóricas se hace necesario el entrenamiento con escenarios típicos SDN para el futuro despliegue de esta nueva arquitectura.

Objetivos:

Este trabajo se centra en el uso de la herramienta Mininet para la emulación de redes SDN [9], teniendo como problema científico:

¿Cómo elaborar en Mininet topologías y escenarios típicos SDN que contribuyan a adquirir habilidades en esta nueva arquitectura de red?

El objetivo general a alcanzar:

- Desarrollar escenarios típicos de redes definidas por software.

Con vistas a alcanzar este objetivo general se deben completar los siguientes objetivos específicos:

- Realizar un estudio de caracterización técnica y operacional de las redes definidas por software y el protocolo OpenFlow.
- Identificar y argumentar el software de simulación adecuado para alcanzar el cumplimiento del objetivo general planteado.
- Realizar un análisis de los controladores SDN.
- Confeccionar escenarios típicos SDN donde se reflejen todas las consideraciones teóricas de este tipo de redes.
- Implementar los escenarios SDN en Mininet

2. Metodología

2.1 Arquitectura SDN

En la recomendación UIT-Y.3302 se define la arquitectura básica de una Red Definida por Software como se observa en la Figura 1, la cual consta de la capa de aplicación SDN (SDN-AL, por sus siglas en inglés), capa de control SDN (SDN-CL, por sus siglas en inglés) y la capa de recursos SDN (SDN-RL, por sus siglas en inglés), así como las interfaces entre ellas y las funciones de gestión multicapa (MMF, por sus siglas en inglés) [3], [4].

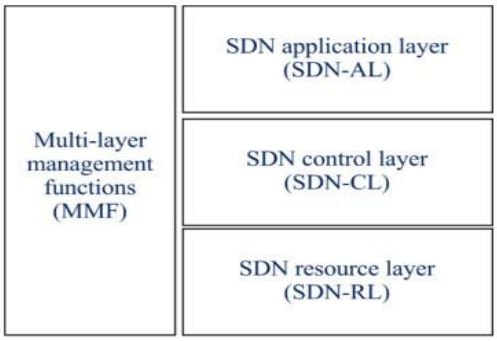

Figura 1. Arquitectura SDN según la UIT [4]

Según la recomendación UIT-Y.3300 las capas de la arquitectura SDN se definen como:

Capa de recursos SDN

El SDN-RL es donde los elementos de la red física o virtual realizan el transporte y/o procesamiento de paquetes de datos según las decisiones de SDN-CL. La información de aprovisionamiento de políticas (incluyendo información de configuración) que resultan como decisiones tomadas por la SDN-CL, así como la información acerca de los recursos de red se intercambian a través da la interface de control de recursos (RCI, por sus siglas en inglés). El SDN-RL también interactúa con MMF utilizando el punto de referencia de la capa de recursos MMF (MMFR) [5]. La información intercambiada a través de la interfaz de control de recursos (RCI, por sus siglas en inglés) incluye información de control proporcionada por la SDN-CL a la SDNRL (por ejemplo, para configurar un recurso de red o proporcionar políticas), así como la información que se refiere a las notificaciones (no solicitadas) enviadas por el SDN-RL siempre que en un recurso de red se detecta un cambio. El punto de referencia de RCI proporciona acceso de alto nivel a los recursos de la red, independientemente de su respectiva tecnología [4].

Capa de control SDN

SDN-CL proporciona medios programables para controlar el comportamiento de los recursos SDN-RL (como transporte y procesamiento de datos), siguiendo solicitudes recibidas de SDN-AL y de acuerdo con las políticas MMF. SDN-CL opera con recursos proporcionados por la SDN-RL y expone una vista abstracta de la red controlada a la SDNAL. El SDN-CL interactúa con el SDN-RL utilizando el punto de referencia de las interfaces de control de recursos (RCI), y con MMF usando el punto de referencia de la capa de control MMF (MMFC). También interactúa con el SDN-AL a través de la interface de control de aplicaciones (ACI, por sus siglas en inglés) [5].

Capa de aplicación SDN

SDN-AL habilita un comportamiento con conocimiento del servicio de la red subyacente de una manera programática. Esta capa dispone de una abstracción superior a las demás capas que permite crear aplicaciones para automatizar tareas de configuración, provisión y despliegue de los servicios en la red. La SDN-AL interactúa con la SDN-CL a través del punto de referencia de la interfaz de control de aplicaciones (ACI), e interactúa con el MMF a través del punto de referencia de capa de aplicación de MMF (MMFA) [5], [4].

2.2 OpenFlow

Se estandariza a OpenFlow como el protocolo que establece la comunicación entre el plano de control y el plano de datos. Permite el acceso directo y manipulación del plano

de reenvíos de los dispositivos de red, tanto físicos como virtuales. OpenFlow hace posible que se pueda mover el control de la red fuera de los dispositivos para centralizarlo lógicamente en el software de control y programar el plano de reenvío. El protocolo OpenFlow se implementa entre la infraestructura de los dispositivos de red y el software de control de SDN. Utiliza el concepto de flujos para identificar el tráfico de red basado en reglas de coincidencia predefinidas que pueden ser programadas de forma estática o dinámica por el controlador SDN lo cual permite definir cómo el tráfico debe fluir a través de los dispositivos de red [2], [6], [7].

2.3 Controladores

Los controladores son otro de los componentes que forman las SDN, estos ofrecen una interfaz de programación para los conmutadores Openflow que les permita gestionar el estado de la red y es quien dicta el comportamiento general de la red a partir de los requerimientos de las aplicaciones. Ejemplos de algunos controladores de código abierto existentes son: Beacon, Floodlight, NOX, POX, Ryu, Trema y OpenDayLight (ODL). Una vez que se analizan los resultados que se publican en [8], [9], [10], [11], [12], [13], [14], [15], [16] se decide utilizar en este trabajo el controlador OpenDayLight [17].

2.4 Simulación de redes SDN

Las redes SDN están en constante desarrollo y su simulación para el estudio o para su implementación es de vital importancia [18]. Softwares utilizados en la simulación y/o emulación de redes SDN: Mininet, Cisco Packet Tracert, GNS3, EsitNet.

Mininet [19], [20] es un emulador para el despliegue de redes sobre los limitados recursos de un ordenador sencillo simple o máquina virtual. Éste utiliza el kernel de Linux para emular elementos de las SDN como el controlador, los switch OpenFlow y los hosts. Usa virtualización muy ligera para realizar un sistema que simula una red completa, ejecutando el mismo kernel, sistema y código de usuario.

2.5 Herramientas necesarias

MiniNAM

MiniNAM es una herramienta de apoyo al Mininet basada en GUI escrita en Python Tkinter Proporciona animación en tiempo real de cualquier red creada por el emulador Mininet. Permite la modificación dinámica de preferencias y filtros de paquetes: un

usuario puede ver flujos selectivos con opciones para codificar paquetes de códigos basados en el nodo de origen / destino y/o el tipo de paquete [20].

IPerf, cURL y Wget

Iperf: Programa cliente-servidor muy sencillo que permite medir la velocidad máxima que alcanzan 2 ordenadores conectados en red local. Esto es útil si se quiere ver la velocidad máxima del switch o router y si cambiando ciertos parámetros como el MTU de la red, se consige más velocidad [21].

cURL: El comando Curl se usa para transferir archivos desde un servidor, es compatible con varios protocolos como HTTP, HTTPS, FTP, FTPS, IMAP, IMAPS, DICT, ARCHIVO, GOPHER, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET y TFTP, etc. Curl también es compatible con muchas funciones como soporte de proxy, autenticación de usuario, carga FTP, publicación HTTP, conexiones SSL, cookies, pausa y reanudación de transferencia de archivos [22], [23].

Wget: es una herramienta informática creada por el Proyecto GNU. Su uso principal es para recuperar contenido y archivos de varios servidores web. El nombre es una combinación de World Wide Web y la palabra get. Admite descargas a través de FTP, SFTP, HTTP y HTTPS [22].

Wireshark: analizador de protocolo de red más importante y ampliamente utilizado en el mundo. Le permite ver lo que está sucediendo en su red a un nivel microscópico y es el estándar de facto en muchas empresas comerciales y sin fines de lucro, agencias gubernamentales e instituciones educativas.

Hipervisores: Hipervisor o Monitor de Máquina Virtual (VMM, por sus siglas en inglés) es una tecnología compuesta por una capa de software que permite utilizar, al mismo tiempo, diferentes sistemas operativos o máquinas virtuales en una misma computadora central. Es la parte principal de una máquina virtual que se encarga de manejar los recursos del sistema principal exportándolos a la máquina virtual [7].

3. Resultados y discusión

A partir del uso de herramientas de simulación y emulación de redes SDN Mininet y el analizador de protocolos de red Wireshark se desarrollan 2 escenarios referidos a la nueva arquitectura de red SDN.

- El primer escenario incluye temas relacionados el uso del Mininet, el MiniNAM, el controlador OpenDayLight y las topologías básicas SDN prediseñadas que ofrece dicho emulador y el análisis del flujo utilizando Wireshark.
- El segundo escenario incluye la emulación de un servidor HTTP en redes SDN y mediciones de ancho de banda con la utilización del controlador OpenDayLight.

Escenario 1: Introducción al uso de Mininet y el controlador ODL. Creación de las topologías SDN básicas.

En este proyecto se realiza una introducción al uso de Mininet y a la implementación de un controlador externo para el desarrollo de los escenarios. Se simulan diferentes topologías predefinidas en Mininet y su visualización en la interface visual del controlador ODL. Se muestran los paquetes ICMP con el uso del MiniNAM y OpenFlow con la utilización del Wireshark.

Ejecutar topología sin controlador remoto:

Para ejecutar el MiniNAM se abre una ventana de terminal de Ubuntu (Ctrl+Alt+t) y se inicia una topología single de dos usuarios mediante la CLI: $ sudo Python MiniNAM.py --topo single,2. Esta línea crea una topología en Mininet la cual está compuesta por un Switch OpenFlow conectado a dos hosts, además de un controlador de referencia OpenFlow e inicia MiniNAM

Iniciar el controlador remoto:

Para iniciar el controlador se abre una nueva ventana de terminal y se ejecuta desde la ubicación donde se encuentre que en este caso sería mediante la CLI: sudo ./Opendaylight/bin/karaf. Esta línea ejecuta la versión Boron del controlador OpenDayLight.

Iniciar topología single y analizar paquetes con Wireshark

Para iniciar la topología single con la utilización de un controlador ODL usando la CLI: $ sudo mn--topo single,2 controller=remote,ip=127.0.0.1,port=6633 –switch ovs,protocols=OpenFlow13

Iniciar topología linear, verificar direcciones IP y conexiones de los equipos

Para iniciar la topología linear con la utilización de un controlador ODL se ejecuta la CLI:

$ sudo mn--topo linear,4 -- controller=remote,ip=127.0.0.1,port=6633 –switch ovs,protocols=OpenFlow13

Queda creada una red compuesta por 1 controlador ODL, 4 switch conectados entre ellos y un usuario conectado a cada switch.

Iniciar topología tree

Para ejecutar la topología *tree* conformada por 16 *hosts*, 15 conmutadores y un controlador ODL conectados en forma de ramificación de cuatro niveles se utiliza la CLI:

$ sudo mn --topo tree,4 --controller=remote,ip=127.0.0.1,port=6633 –switch ovs,protocols=OpenFlow13

Todas las topologías presentadas anteriormente no se han podido visualizar puesto que el emulador Mininet no cuenta con una interface para mostrar la red, pero como han sido implementadas con un controlador ODL mediante la interface visual que este ofrece si es posible verlas. En la Figura 2 se muestra la topología *tree* visualizada en el DLUX ODL.

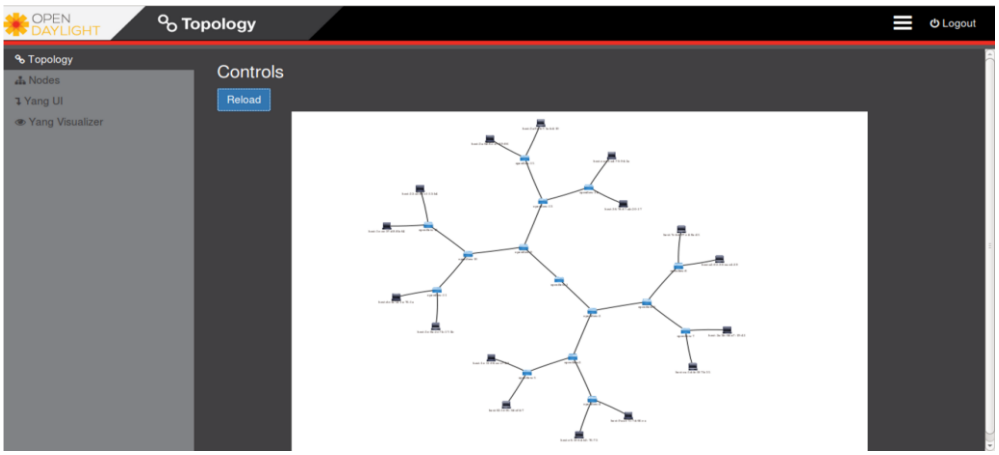

Figura 2. Topología usando el DLUX. ODL.

En la Figura 3 se puede comprobar otra característica teórica de SDN y OpenFlow, y es que cuando al dispositivo llega un paquete que no coincide con ninguna entrada en la tabla de flujo, lo envía al controlador para que le indique qué hacer con él. Una vez que lo sabe, ya no necesita esta comunicación, y realiza el reenvío por su cuenta. Si se realiza de nuevo un ping entre los dos hosts, se puede apreciar la diferencia de tiempo entre la primera vez y sucesivos intentos. En Figura 3 se muestran los resultados el correcto funcionamiento del Mininet mediante la ejecución de la CLI *h1 ping h2* en la topología

single de dos hosts creada sin controlador remoto y en la Figura 4 se visualiza en el MiniNAM.

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.473 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.196 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.189 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.168 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.130 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.203 ms
```

Figura 3. Ping entre el host 1 y el host 2.

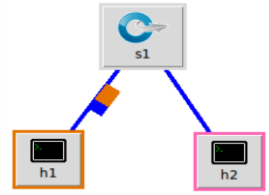

Figura 4. Flujo ICMP en MiniNAM.

En la Figura 5, mediante el uso del Wireshark ejecutado en un nuevo terminal de Ubuntu se muestra el tráfico de los paquetes ICMP entre los hosts de la red single y los paquetes OpenFlow entre el conmutador y el controlador al ejecutar la CLI *pingall.*

```
$ sudo wireshark
```

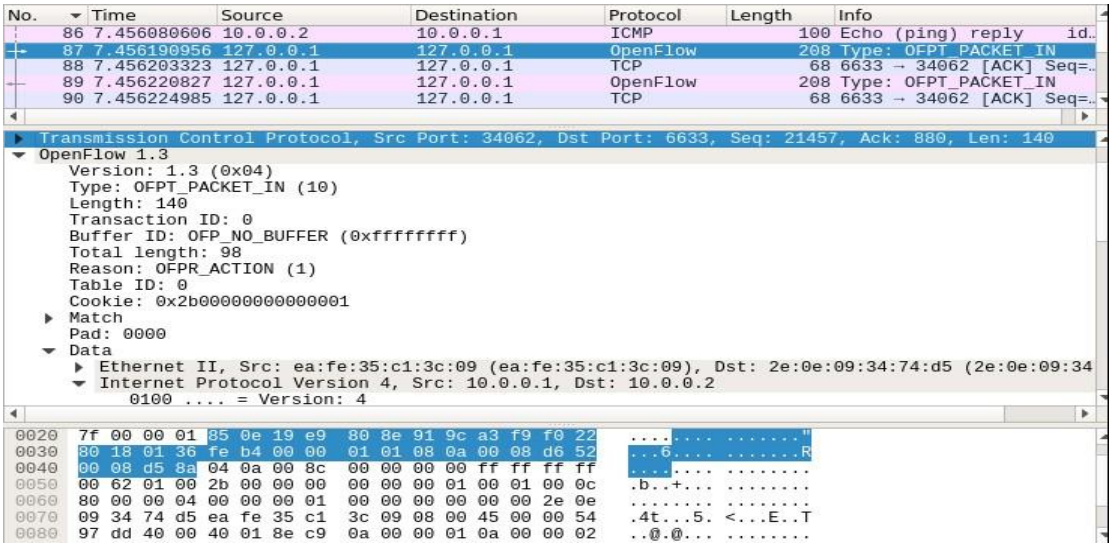

Figura 5. Captura de tráfico ICMP y OpenFlow generado por la utilización de *pingall*. Fuente: Autor.

En Figura 5 se observa como al realizar un ping entre dos usuarios de la red el conmutador por el puerto 34062 le envía un paquete OpenFlow al controlador que lo recibe por el puerto 6633 con la información de los eventos ocurridos especificando el protocolo, en este caso el ICMP y la fuente y el destino de los datos.

En la Figura 6 se observa la información correspondiente a los enlaces de la red *linear* mediante la utilización de a CLI *net.*

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
```

Figura 6. Información de los enlaces de la red de 4 *host*. Fuente: Autor.

Escenario 2. Emulación de un servidor HTTP y mediciones de ancho de banda.

Mediante la utilización de la herramienta de emulación de redes SDN Mininet y el controlador ODL en este escenario se realizan mediciones de ancho de banda entre hosts con la utilización de la herramienta iperf de Ubuntu estableciendo conexiones TCP y se realiza el acceso a un servidor HTTP mediante la herramienta wget y cURL de Ubuntu.

Iniciar el controlador ODL

Creación del escenario

$ sudo mn --topo single,7 --controller=remote,ip=127.0.0.1,port=6633 --switch ovs,protocols=OpenFlow13

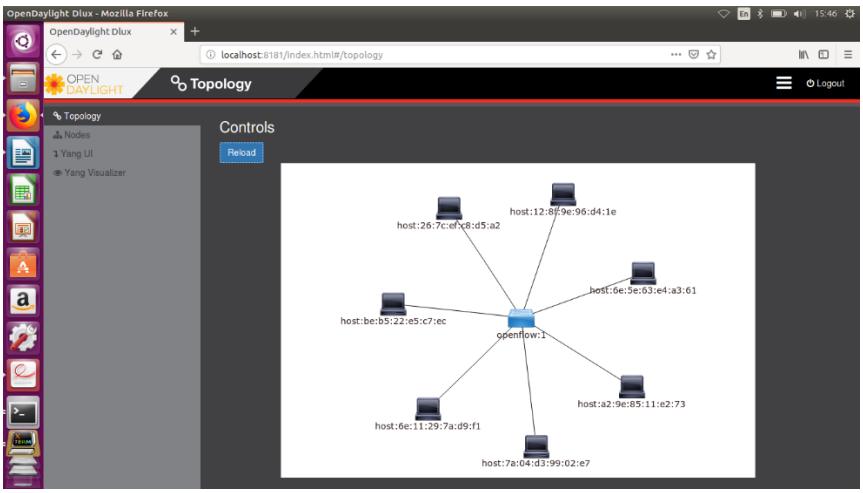

Figura 7. Topología single visualizada en el DLUX ODL. Fuente: Autor.

Análisis de ancho de banda

Para configurar los *hosts* para la realización de la transferencia y el análisis del ancho de banda se ejecuta dentro de Mininet la CLI:

```
mininet> xterm h1 h2
```

Esta línea abrirá dos nuevas ventanas de terminal para el trabajo con el *host* 1 y el *host* 2 como se muestra en la Figura 8.

En la ventana de terminal "Node: h2" se utiliza la herramienta de Ubuntu iperf mediante la CLI:

```
root@yudi:~# iperf -s
```

Y en la ventana de terminal "Node: h1", para acceder al *host* 2 se ejecuta la CLI:

```
root@yudi:~# iperf -c 10.0.0.2
```

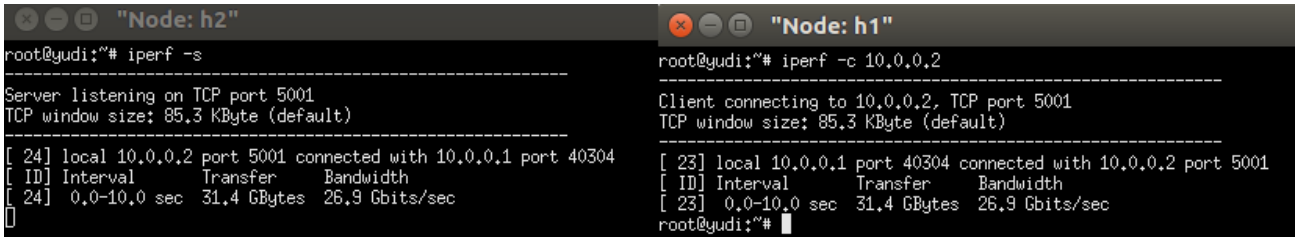

Figura 8. Ventanas xterm del host1 y host 2. Fuente: Autor.

Con la utilización de la herramienta iperf se realiza una transferencia TCP entre el *host* 1 y el *host* 2 y se mide el ancho de banda de transmisión que Mininet estableció.

Configuración de los hosts

Para ello se abren las ventanas de terminal correspondiente a cada host mediante la CLI:

```
mininet> xterm h1 h2 h3 h4 h5 h6 h7
```

El host 2 en la ventana de terminal "Node: h2" se configura como un servidor HTTP accesible por el puerto 80 como se muestra en la Figura 9, mediante la CLI:

```
root@yudi:~# python -m SimpleHTTPServer 80
```

Figura 9. Host 2 como servidor HTTP. Fuente: Autor.

En la Figura 10 se observa como desde el host 1 se accede al servidor creado en el host 2 utilizando la herramienta wget mediante la CLI:

```
root@yudi:~# wget http://10.0.0.2
```

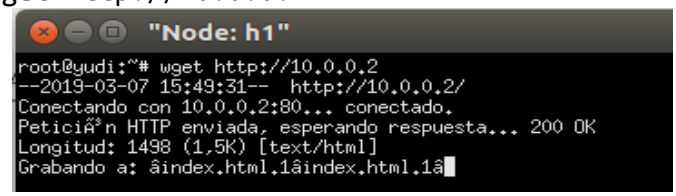

Figura 10. Host 1 accediendo a servidor HTTP

En la Figura 11 se muestra como queda registrado con fecha y hora el acceso al servidor para la descarga de la página web.

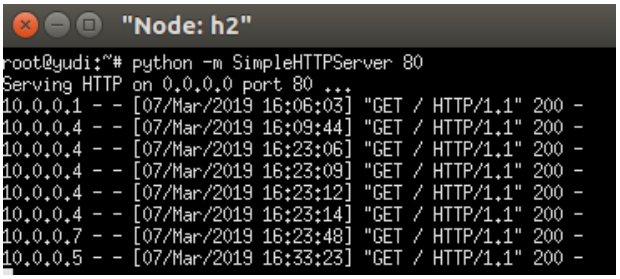

Figura 11. Servidor HTTP después realizada la descarga de la página. Fuente: Autor.

Acceso al servidor HTTP

Desde los restantes *hosts* de la red (h3, h4, h5, h6, h7) se accede al servidor HTTP ubicado en la dirección 10.0.0.2 como se muestra en la Figura 12, a través de la ventana de terminal correspondiente a cada uno usando la CLI:

```
root@yudi:~# curl 10.0.0.2
```

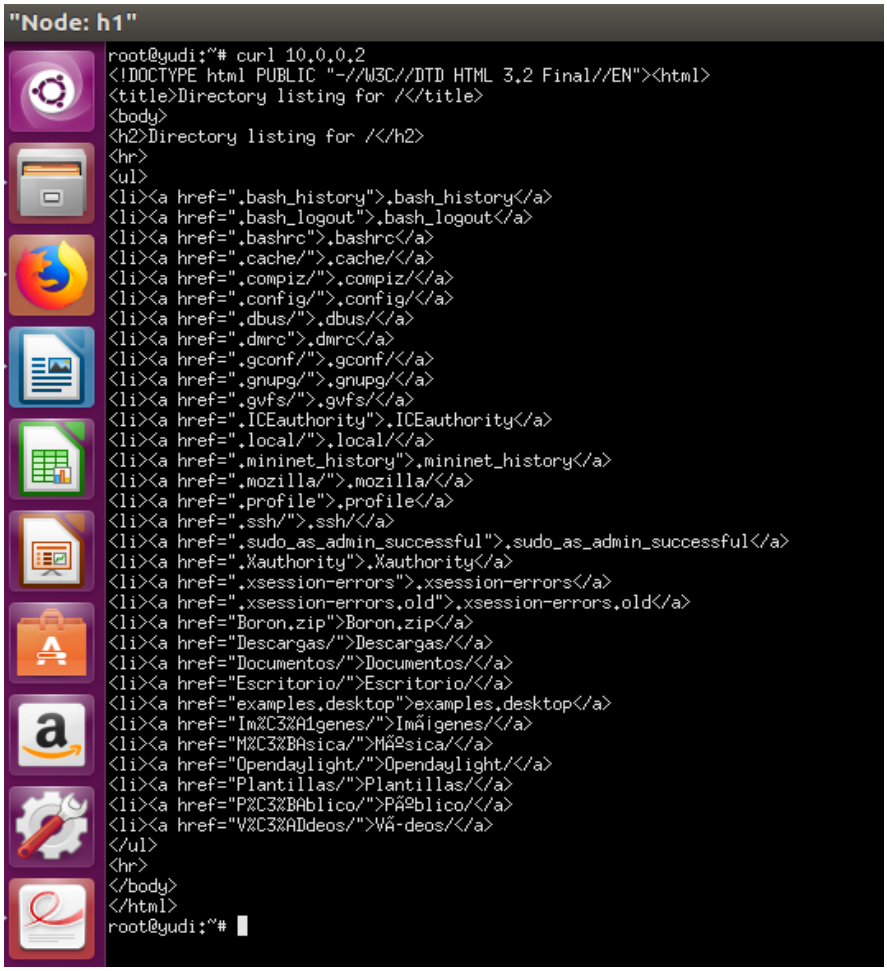

Figura 12. Acceso a la página web creada en los servidores HTTP.

4. Conclusiones

Con la realización de este trabajo se obtuvieron los siguientes resultados:

Las redes SDN ofrecen la posibilidad de hacer una gestión mucho más especializada del tráfico de la red, de esta forma cuando un switch OpenFlow recibe un paquete que no sabe encaminar, se lo transmite al controlador para que decida qué hacer con este tráfico.

La comunicación entre el controlador y el switch ocurre usando el protocolo OpenFlow, donde se puede intercambiar un conjunto de mensajes definidos entre estas entidades a través de un canal seguro. El puerto TCP predeterminado del controlador suele ser el 6633. OpenFlow define tres tipos de mensajes con sus respectivos subtipos para mantener la comunicación entre las partes: Controlador a conmutador, asíncronos y simétricos.

Mininet es una herramienta ideal para usuarios primerizos que no tienen acceso a entornos más avanzados. Permite crear redes conmutadas simuladas para entender el funcionamiento básico de los controladores y los flujos.

Los resultados obtenidos para los proyectos desarrollados fueron:

Escenario1: Se comprobó la conectividad total entre los *hosts* de las diferentes topologías predefinidas creadas y que las topologías creadas fueran las deseadas por el usuario. Además, se realizó un análisis con el uso del Wireshark del tráfico de paquetes ICMP y OpenFlow al realizar ping entre los *hosts* de la red.

Escenario 2: Se desarrollaron mediciones de ancho de banda y se emuló la creación de un servidor HTTP utilizando las herramientas iperf, wget y cURL, comprobándose el correcto funcionamiento de la arquitectura de la red SDN propuesta mediante el acceso de los hosts a dicho servidor.

5. Referencias bibliográficas

[1] Y. Jarraya, T. Madi, y M. Debbabi, «A Survey and a Layered Taxonomy of Software-Defined Networking», *IEEE Communications Surveys Tutorials*, vol. 16, n.º 4, pp. 1955-1980, Fourthquarter 2014.
[2] E. H. O. Koufopavlou, «RFC 7426 ¨Software-Defined Networking (SDN): Layers and Architecture Terminology¨». ene-2015.
[3] ITU-T, «"Y.3300 : Framework of software-defined networking"». 06-jun-2014.
[4] ITU-T, «"Y.3302 : Functional architecture of software-defined networking"». 15-mar2017.
[5] ITU-T, «"Y.3301 : Functional requirements of software-defined networking"». 16-dic-2016.
[6] A. L. B. Ramamurthy, «''Network innovation using OpenFlow: A survey''», *IEEE Commun. Surv. Tut.*, vol. 16, n.º 1, pp. 493–512, First Quart 2014.
[7] Guy Pujolle, *Software Networks Virtualization, SDN, 5G and Security*, vol. 1. UK USA: ISTE Ltd and John Wiley & Sons, Inc., 2015.
[8] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, y E. C. Popovici, «A comparison between several Software Defined Networking controllers», en *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, Nis, Serbia, 2015, pp. 223-226.
[9] M. Brooks y B. Yang, «A Man-in-the-Middle attack against OpenDayLight SDN controller», en *Proceedings of the 4th Annual ACM Conference on Research in Information Technology - RIIT '15*, Chicago, Illinois, USA, 2015, pp. 45-49.
[10] A. Mayoral, R. Vilalta, R. Munoz, R. Casellas, R. Martinez, y J. Vilchez, «Integrated IT and network orchestration using OpenStack, OpenDaylight and active stateful PCE for intra and inter data center connectivity», en *2014 The European Conference on Optical Communication (ECOC)*, Cannes, France, 2014, pp. 1-3.
[11] T. Kim, S.-G. Choi, J. Myung, y C.-G. Lim, «Load balancing on distributed datastore in opendaylight SDN controller cluster», en *2017 IEEE Conference on Network Softwarization (NetSoft)*, Bologna, Italy, 2017, pp. 1-3.
[12] D. Suh, S. Jang, S. Han, S. Pack, T. Kim, y J. Kwak, «On performance of OpenDaylight clustering», en *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, South Korea, 2016, pp. 407-410.
[13] Y. Zhao, L. Iannone, y M. Riguidel, «On the performance of SDN controllers: A reality check», en *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, San Francisco, CA, 2015, pp. 79-85.
[14] P. Berde *et al.*, «ONOS: towards an open, distributed SDN OS», en *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN '14*, Chicago, Illinois, USA, 2014, pp. 1-6.
[15] Z. K. Khattak, M. Awais, y A. Iqbal, «Performance evaluation of OpenDaylight SDN controller», en *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Hsinchu, Taiwan, 2014, pp. 671-676.
[16] R. K. Arbettu, R. Khondoker, K. Bayarou, y F. Weber, «Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers», en *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, Montreal, QC, Canada, 2016, pp. 37-44.

[17] «OpenDaylight A Linux Foundation Collaborative Project», *OpenDaylight*, 2018. [En línea]. Disponible en: http://www.opendaylight.org.

[18] D. Bolatti, «Estudio de herramientas de simulación en redes definidas por software», presentado en XIX Workshop de Investigadores en Ciencias de la Computación, Buenos Aires, 2017.

[19] «"Mininet." [Online]». .

[20] B. V. y J. J. Padilla, «Mininet: una herramienta versátil para emulación y prototipado de Redes Definidas por Software», *Entre Ciencia e Ingeniería*, vol. 9, n.º 17, pp. 62-70, 2015.

[21] «Ubuntu Manpage: iperf - perform network throughput tests». .

[22] «CURL vs. WGET: Sus diferencias, uso y cuál deberías usar | Maslinux». 04-ene-2018.

[23] J. López, «cURL: control de páginas y transferencias», *Todo linux: la revista mensual para entusiastas de GNU/LINUX*, n.º 76, pp. 36-39, 2007.