Journal de Ciencia e Ingeniería

Journal de Ciencia e Ingeniería, vol. 10, no. 1, pp. 55-61, agosto de 2018
Investigación - Ingeniería de Sistemas
ISSN 2145-2628, e-ISSN 2539-066X

Redes de datos definidas por software - SDN, arquitectura, componentes y funcionamiento

Software Defined Networking - SDN, architecture, components and functioning

Miller Ramírez Giraldo [a] ⋆ Ana María López Echeverry [a], ⋆⋆

[a] *Grupo de Investigación Nyquist, Universidad Tecnológica de Pereira, Pereira, Colombia*

Recibido: 20/06/2017; revisado: 15/11/2017; aceptado: 09/01/2018

M. Ramírez Giraldo, A. M. López Echeverry: Redes de datos definidas por software - SDN, arquitectura, componentes y funcionamiento *Jou.Cie.Ing*. **10** (1): 55-61, 2018. ISSN 2145-2628, e-ISSN 2539-066X.

Resumen

El elevado crecimiento de usuarios conectados a la Internet y la versatilidad de aplicaciones disponibles generan nuevos retos para las redes de datos que soportan la comunicación. Se propone desde el ámbito académico un nuevo paradigma para las redes de datos, las redes definidas por software o por sus siglas en ingles SDN (Software Defined Networking). El objetivo de este artículo es explicar de la manera más clara posible lo que es SDN, su arquitectura, componentes y funcionamiento. Para realizar la explicación del concepto, se propone usar una topología tipo de una red convencional describiendo su comportamiento, para después sobre la misma topología explicar lo que sucedería bajo el nuevo paradigma. Se presentarán las herramientas desarrolladas en la actualidad para llevar a cabo una implementación de SDN, algunos fabricantes que presentan un buen desarrollo y soporte para el nuevo paradigma, así como casos de éxito de implementaciones funcionales de SDN en la vida real.

***Palabras Claves*:** SDN, paradigma, topología convencional, comportamiento.

Abstract

The growing number of users connected to the Internet and the versatility of available applications create new challenges for data networks that support communication. Within the academic field, a new paradigm called Software Defined Networking (SDN) is being proposed for data networks. To understand SDN, suppose a description of a conventional network's function, concepts, and achitecture are used, Then, using the same topology, we will explore what will happend under the new paradigm with these concepts. We will present the tools currently developed to carry out an implementation of SDN, some manufactures that have a well developed support system for the new paradigm, as well as cases of success of functional implementations of SDN in real life.

***Keywords*:** SDN, paradigm, conventional network, function.

1. Introducción

Las redes de datos actuales que soportan la existencia de la Internet se podrían resumir en un conjunto

⋆ miller@utp.edu.co
⋆⋆anamayi@utp.edu.co

de sistemas autónomos que interiormente ejecutan protocolos de enrutamiento de gateway interior â IGP (Enrutamiento estático, RIPv1, RIPv2, OSPF, IS-IS, EIGRP) y que a su vez son interconectados entre sí gracias a protocolos de enrutamiento de gateway exterior EGP (el único existente y en uso BGP). Es sabido que los protocolos IGP se encargan de seleccionar las mejores rutas obedeciendo netamente a criterios técnicos, mientras que en el caso de los protocolos EGP como es el caso particular de BGP, es un protocolo que permite seleccionar las mejores rutas obedeciendo a criterios tanto a nivel técnico como comercial, es decir, permite la manipulación en los procesos de selección de ruta.

A pesar que BGP permite la implementación de políticas de enrutamiento con un alto nivel de complejidad, está enmarcado en un conjunto de reglas y condiciones no muy flexibles, por ejemplo el simple hecho de tomar como criterio de selección de ruta únicamente la dirección IP de destino a la hora de realizar la búsqueda en la tabla de enrutamiento. En ningún momento se puede tomar como criterio la dirección IP de origen, la dirección de capa de enlace de datos de origen o de destino, el puerto UDP o TCP de origen o de destino (es decir datos de las cabeceras de capa 2, 3 y 4). Debido a esta necesidad de poder programar el comportamiento de una red de datos sin estar atado a reglas y condiciones poco flexibles, surge la propuesta de este nuevo paradigma denominado SDN.

El desarrollo de este artículo se compone de las secciones 2 y 3 donde se explica el funcionamiento de una red de datos tipo, compuesta solamente por conmutadores o Switches capa 2 en el modelo de red actual y convencional, y en una etapa posterior se explica el funcionamiento de la misma red tipo bajo el paradigma de las redes definidas por software. En la sección 4 se explica de manera general la arquitectura SDN y sus componentes. La sección 5 presenta las aplicaciones de software y protocolos del estado del arte disponibles en la actualidad para la implementación de una red SDN funcional. Finalmente las secciones 6 y 7 presentan los fabricantes de equipos de comunicaciones que cuentan con desarrollos importantes en SDN y las implementaciones exitosas de SDN en casos reales. El artículo tiene como principal objetivo permitir introducir a las personas con conocimientos en redes de datos convencionales a lo que propone este nuevo paradigma, dejando la puerta abierta para que puedan continuar el aprendizaje a fondo en este nuevo modelo.

2. Funcionamiento de red conmutada convencional

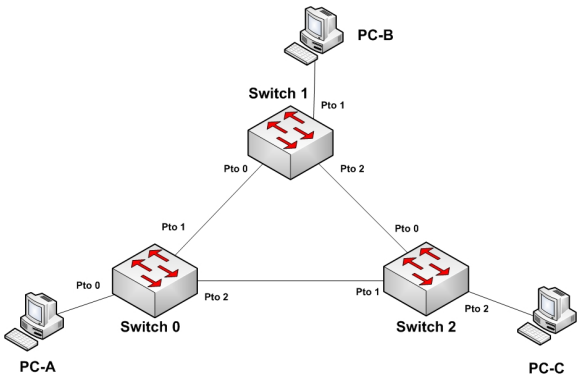

Figura 1. Topología en Paradigma Convencional

En la figura 1, se presenta una red tipo compuesta por Switches capa 2, con una PC conectada a cada Switch. El funcionamiento de esta red tipo en el paradigma convencional presenta varios aspectos a tener en cuenta. El primer aspecto a tener en cuenta es la consideración de diseño de las redes Ethernet en las que no existe un campo TTL en la las cabeceras de capa 2, por lo tanto debe existir un solo camino lógico en toda la topología (es decir un solo árbol Span en todo el grafo físico). Para llevar a cabo esta tarea es bien sabido que las variantes de los protocolos de STP (RSTP, MSTP, PVSTP, etc) son encargados de alcanzar este estado. Para este caso vamos a suponer que RSTP se ejecutó en la topología y que el árbol Span generado es aquel compuesto por las conexiones entre Swtich 0, Swtich 1 y Switch 2, es decir que la conexión directa entre Switch 0 y Switch 2 queda lógicamente deshabilitada.

Para que una comunicación entre PC-A y PC-B se logre, se deben dar los siguientes eventos [3]. Para un paquete dirigido de PC-A a PC-B que ingresa al Switch 0 por el puerto cero, el Switch 0 realiza las siguientes actividades, primero revisa la dirección MAC origen y crea una entrada en la tabla CAM en la que se asocia esa dirección MAC al puerto número 0, esto con la finalidad de que el Switch aprenda que la PC-A se puede alcanzar a través de este puerto. Posteriormente revisa la dirección MAC de destino para determinar por donde debe conmutar el paquete, como no cuenta con una entrada en la tabla CAM que le permita saber dónde se encuentra el PC-B entonces realiza un flooding (enviar el paquete por todos los puertos a excepción del puerto por donde lo recibió), en este caso envía la trama por el puerto número uno (es importante recordar que el puerto número 2 pertenece a un enlace deshabilitado

lógicamente, por lo tanto no es un puerto válido para inundar el paquete). Cuando la trama ingresa por el puerto número 0 del Swtich 1 el proceso se repite nuevamente, se aprende que la PC-A se puede alcanzar a través del puerto cero y se inunda la trama a los puertos diferentes a aquel por donde se recibió. Por lo tanto se envía la trama por los puertos uno y dos del Switch 1. Hasta este punto el único aprendizaje alcanzado en la red es el de los Switches 0 y 1 quienes saben a través de qué puertos alcanzar a la PC-A.

La trama se envía a PC-B quien acepta la trama ya que la dirección MAC de destino es su propia dirección MAC, por lo que prepara la respuesta (dependiendo de la aplicación usada â FTP, Web, Chat, Video, etc). Por otro lado la trama que se transmitió por el puerto 2 del switch uno sigue su camino hacia el Switch 2 quien repite nuevamente la operación, es decir, ingresa en su tabla CAM una entrada para la dirección MAC de la PC-A asociada al puerto número 0 y realiza un flooding hacia el puerto número 2. En lo que respecta al Switch 2 las actividades asociadas a esta comunicación finalizan en este punto, por lo tanto el Switch 2 aprende que para llegar al PC-A lo puede hacer enviando las tramas mediante el puerto número cero. Se retoma la comunicación al PC-B quien prepara la respuesta para la PC-A. Envía entonces un paquete con dirección MAC origen de PC-B y dirección MAC de destino de PC-A. Al ingresar esta trama de respuesta al Switch 1 mediante el puerto, este genera una nueva entrada en la tabla CAM en la que relaciona la dirección de la PC-B asociada al puerto número uno, es decir que en este punto el Switch uno sabe exactamente por donde enviar los paquetes destinados a los PCs A y B. Como el Switch 1 ya había generado previamente una entrada para el PC-A, este sabe que únicamente debe enviar la trama a través del puerto cero (consultando previamente su tabla CAM). La trama ingresa por el puerto número uno al Switch 0, quien ingresa un nuevo registro en su tabla CAM, asociando a la dirección MAC de la PC-B por el puerto número uno. Como el Switch 0 ya contaba con una entrada en su tabla CAM que le indicaba por qué puerto alcanzar a la PC-A, este envía la trama por el puerto número cero logrando finalmente la comunicación entre las dos computadoras.

Para la comunicación posterior entre la PC-A y la PC-B los paquetes seguirán la misma trayectoria PC-A â Switch 0 â Switch 1 â PC-B, ya que hasta ese momento ambos Switches saben cómo alcanzar a estas PCs.

Para la comunicación entre el PC-A y PC-C o PC-B y PC-C el comportamiento y dinámica de aprendizaje será muy similar cambiando únicamente los registros de las tablas CAM de los Switches. El estado de convergencia final para esta red tipo es que cada Switch de la red conozca por qué puertos enviar la información para cada una de las computadoras conectadas.

En este punto es necesario tener en cuenta varios aspectos sobre el paradigma actual y convencional de las redes de datos. Primero, el único criterio para el envío de las tramas usado por los Switches es la dirección MAC de destino, es decir, no se tiene en cuenta la dirección MAC de origen (usada solamente para el aprendizaje más no para el envío). Segundo, las direcciones IP de origen y destino, números de puerto UDP o TCP de origen y destino, es decir cabeceras de capa 3 y 4 y dirección de origen de capa 2 no son tomadas en cuenta para realizar la conmutación de la trama a un puerto o puertos específicos. Sucede de esta manera, acorde al diseño del modelo de referencia OSI.

Otro aspecto que se debe clarificar son los planos de datos y de control en el modelo convencional [2]:

Plano de control – Plano encargado de determinar las políticas de conmutación y/o enrutamiento en un dispositivo de comunicación. Es decir quien determina qué criterios se toman en cuenta para transmitir un paquete de datos y también los criterios para la selección de las mejores rutas. Plano de datos – Plano que sólo es encargado de la transmisión y recepción de paquetes

En modelo convencional el plano de datos y de control están completamente distribuidos en los equipos de comunicaciones, es decir cada Switch de la red tiene su propio plano de datos y de control. Por lo tanto cada Switch de manera independiente determina la manera en que una trama de datos será enviada hacia los nodos vecinos. Otro punto a tener en cuenta es que si se desea generar una implementación en el plano de control, se debe hacer en cada Switch de la red, uno a uno. A continuación se presenta el mismo escenario de red tipo pero bajo el nuevo paradigma de redes definidas por software SDN. Se realizará igualmente una descripción del comportamiento paso a paso en este nuevo modelo.

3. SDN en el mismo escenario propuesto de red conmutada convencional

Para este escenario se presenta la misma red tipo del caso anterior pero esta vez bajo el paradigma SDN. El primer aspecto diferente es un dispositivo llamado Controller (representado por un servidor o estación de trabajo con las características de hardware necesarias para correr el software controlador). A continuación se describe una comunicación entre el PC-A y PC-B como en el caso de la red convencional.

El PC-A envía una trama de datos con destino al

PC-B, en primer lugar envía la trama hacia el Switch 0 a través del puerto número cero, hasta este punto todo es igual respecto a las redes convencionales. En el momento que la trama ingresa al switch es donde el paradigma SDN aparece [1] [7]. Este observa las cabeceras del paquete (puede analizar las cabeceras de capa 2, 3 y 4, mientras que un switch convencional solo puede revisar cabeceras de capa 2). Si el Switch no posee una entrada en sus tablas de flujo (en redes convencionales denominadas tablas CAM), envía el paquete o las cabeceras del paquete al dispositivo controlador con la finalidad que se le indique detalladamente como debe conmutar ese paquete, para el caso particular el Switch 0 envía el paquete al controlador y recibe como respuesta una entrada para su tabla de flujo que le indica que el paquete con destino a la PC-B debe ser enviado al puerto número uno, en este momento el switch ingresa esa entrada de flujo, es decir realiza un proceso de aprendizaje, pero esta vez no por su propia cuenta (o su propio plano de control) sino a través de las indicaciones entregadas por el controlador quien actúa como un único plano de control para toda la red. Ocurre algo similar cuando la trama ingresa por el puerto número cero del Switch 1, éste consulta al controlador para que le indique como debe conmutar el paquete y una vez recibe la respuesta ingresa la entrada en su tabla de flujo y conmuta el paquete a través del puerto número uno. El paquete finalmente llega a la PC-B quien dependiendo de la aplicación que ejecute (capa de aplicación del modelo OSI) prepara la respuesta para la PC-A. El PC-B envía entonces la trama al PC-A llegando al Switch uno a través del puerto número uno.

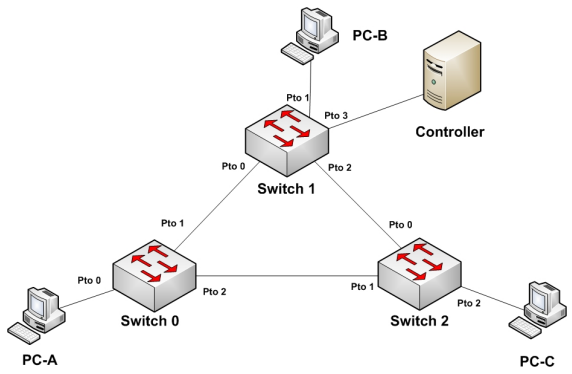

Figura 2. Topología en el Nuevo Paradigma SDN

El Switch 1 consulta nuevamente al controlador quien le indica que debe conmutar la trama a través del puerto número cero, quien conmuta la trama y almacena esa entrada en su tabla de flujo. Al llegar la trama al Switch 0, este realiza nuevamente el proceso de consulta al controlador y finalmente realiza la entrega de la trama por su puerto número cero hacia el PC-A, así se logra la comunicación bidireccional entre ambos PCs. Después de este proceso, cualquier otra trama que requiera ser intercambiada entre las PCs A y B no requerirá de una nueva consulta al controlador para saber cómo conmutarla.

Todas las especificaciones y políticas de conmutación o enrutamiento son definidas por el administrador o programador de la red en el dispositivo controlador. Este modelo ofrece la posibilidad de establecer abiertamente los tiempos para cada implementación (políticas de enrutamiento y conmutación por minutos, horas, días, etc).

Este paradigma realmente permite programar el comportamiento de la red de datos a decisión del administrador o programador, ya que permite que un paquete se conmute o enrute hacia un determinado puerto dependiendo de muchas opciones que incluyen dirección MAC de origen, dirección MAC de destino, dirección IPv4 de origen, dirección IPv4 de destino, dirección IPv6 de origen y de destino, puertos UDP o TCP de origen y de destino, números de VLAN, prioridad de la VLAN, IP ToS, TTL, etiquetas MPLS, etc; mientras que en redes convencionales los criterios solamente se ajustan a la dirección MAC de destino, dirección IP de destino y valores en cabeceras IP para QoS (también las opciones de enrutamiento especial presentes en capa 3) [19].

En SDN el plano de datos está en cada uno de los Switches de la red (al igual que el paradigma convencional), pero el plano de control solamente está presente en el controlador, es decir que está centralizado, lo que permite establecer políticas de conmutación, enrutamiento y actualización en un único dispositivo, evitando la necesidad de realizarlo por cada equipo de comunicación de la red.

A continuación se presenta la arquitectura de SDN, sus componentes y funcionamiento general.

4. Arquitectura y componentes de SDN

La arquitectura de SDN está compuesta por los siguientes elementos:

- Protocolo (Software)
- Equipo de cómputo Controlador (Hardware y Software)
- Sistema operativo de red (NOS â Net Operative System) (Software del controlador)

- Lenguajes de programación de alto nivel para el establecimiento de las políticas de comunicación
- Equipos convencionales de Switching y Routing con soporte para el protocolo del paradigma SDN

Para la implementación de una red bajo el paradigma SDN se requiere tener cada uno de los componentes relacionados. En la topología tipo propuesta para ilustrar ambos paradigmas si un administrador o programador de la red desea implementar el modelo SDN deberá realizar las siguientes tareas:

- Conectar los equipos de comunicaciones como lo muestra la topología, incluyendo el equipo de cómputo o servidor que actuará como controlador.
- Garantizar que los equipos de comunicaciones soportan el protocolo para la comunicación con el controlador (normalmente OpenFlow).
- Instalar el sistema operativo en el controlador (GNU/Linux, Mac o Windows según sea el caso).
- Instalar el software controlador (ONOS, OpenDaylight, Beacon, Floodlight, etc).
- Definir las políticas de conmutación o enrutamiento a través de lenguajes de programaciń de alto nivel (C, C++, Java, Python o definición de flujos a través de JSON o XML) o aplicaciones ya desarrolladas.
- Verificar la adecuada inserción de los registros de las tablas de flujo en los equipos de comunicaciones de la red.

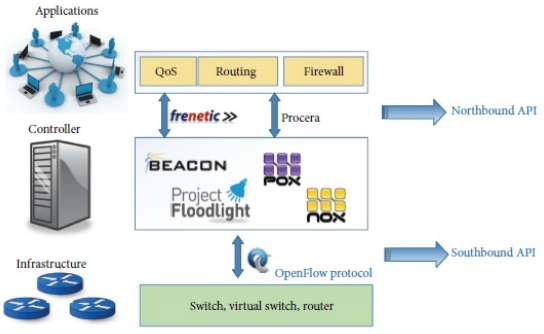

Figura 3. Arquitectura Paradigma SDN [7]

Una vez implementada la red las tablas de los Switches SDN tendrán un aspecto similar al presentado en la figura 4.

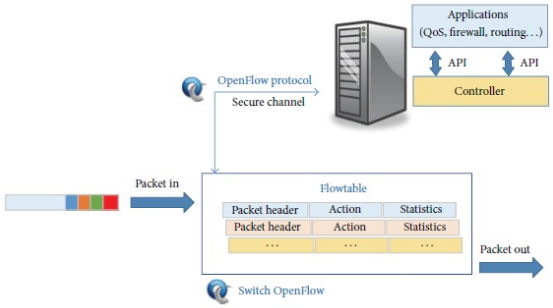

Figura 4. Arquitectura Paradigma SDN [7]

La estructura de los registros de la tabla de flujo se compone de tres elementos [7]:

- PacketHeader o cabeceras del paquete – donde se establecen los valores específicos de las cabeceras del paquete que generarán un match con el registro (datos de las cabeceras de capa 2, 3 y 4 de la trama de datos entrante).
- Action o acciones – las acciones a ejecutar una vez se genere el match con las cabeceras del paquete, por ejemplo conmutar el paquete al puerto número tres.
- Statistics o estadísticas – Contadores con las estadísticas de los match para cada registro de flujo.

Otra característica a resaltar que pueden ofrecer las redes definidas por software es la opción de aplicar las políticas de comunicación con base en el estado de los recurso de la red (procesador y memoria de los equipos de comunicaciones y ancho de banda y latencias de los enlaces), es decir, si la red presenta mucha saturación se podrían aplicar unas políticas, pero se podrían aplicar otras políticas diferentes si la red se encuentra con poca carga (sin importar si son unos pocos equipos o enlaces o la totalidad de los recursos). Esto permite programar el comportamiento de la red dependiendo no sólo de las políticas establecidas sino de su estado.

A continuación se detallan las herramientas disponibles para cada uno de los componentes de la arquitectura SDN.

5. Herramientas disponibles para la implementación de SDN

Las herramientas disponibles para llevar a cabo una implementación real de SDN son las siguientes:

Protocolo – OpenFlow el principal protocolo para la implementación de redes SDN, guiado por la ONF (Open Networking Foundation). Existen otras

iniciativas que aún no son ampliamente adoptadas como:

- NETCONF/YANG
- ForCES
- SoftRouter
- LISP
- PCEP
- OpFlex

Software del equipo controlador – Sistemas operativos de red – NOS (Net Operative System) [23]:

- ONOS, OpenDaylight y Beacon – Tres controladores basados en Java, modulares y multiplataforma.
- NOX – Plataforma OpenSource escrita en C++ o Python
- Trema – Full-Stack Framework de Openflow para lenguajes Ruby y ANSI C
- Maestro – Escrito en JAVA con soporte para Switches OpenFlow
- MUL – Escrito en lenguaje C con una arquitectura centralizada.

Lenguajes para definición de políticas:

- Frenetic – Escrito en Python posee dos módulos, uno para monitoreo de los recursos de la red y otro para el establecimiento de las políticas.
- Procera – Políticas de flujo y enrutamiento basadas en hora del día, cantidad de datos transmitidos, privilegios o grupos de usuarios y tipo de tráfico transmitido. Se puede usar FRP (Funtional Reactive Programming) y Haskell.
- Directamente a través de languajes de programación como C, C++, Java, Python, o de especificación directa de flujos como JSON y XML.

Con estas soluciones de software desarrolladas en la actualidad, un equipo de cómputo para el controlador y dispositivos de comunicación con soporte para el protocolo OpenFlow es posible realizar una implementación real de una red SDN. A continuación se detallan algunos de los principales fabricantes con desarrollos de software importantes para el soporte adecuado del protocolo OpenFlow en sus equipos de comunicaciones.

6. Fabricantes con desarrollos en SDN

El principal protocolo para soporte de redes SDN, OpenFlow, desde su desarrollo ha presentado las siguientes versiones [19]:

- Versión 1.0 liberada en el mes de diciembre del año 2009
- Versión 1.1 liberada en el mes de febrero del año 2011
- Versión 1.2 liberada en el mes de diciembre del año 2011
- Versión 1.3 liberada en el mes de junio del año 2012
- Versión 1.4.0 liberada en el mes de agosto del año 2013
- Versión 1.5.0 liberada en el mes de diciembre del año 2014 última versión.

Los fabricantes de equipos de comunicaciones que soportan completamente hasta el release 1.3 son:

- NEC
- HP Networking
- Cisco Systems
- Alcatel-Lucent
- Huawei
- Spirent
- ICIA
- DCN
- xNet
- Greenet
- ZTE
- Pica8
- H3C
- EstiNet

7. Casos de éxito de implementaciones de redes SDN

En la implementación de este nuevo paradigma se presentan algunos casos de éxito llevados a la realidad, como son los siguientes [6]:

- Empresa de Transporte en Metro - East Japan RailwayCompay, presente en el área este de Japón hasta el área metropolitana de Tokyo.
- Hospital Universitario - Nagoya City University Hospital, en Nagoya Japón.
- Datacenter Interno de NEC.
- Empresa Nippon Express.
- Empresa Genesis Hosting Solutions.
- Hospital Universitario - Kanazawa University Hospital, en Kanazawa Japón.

Son algunos de los casos de éxito de implementaciones reales de redes de datos definidas por software mediante el uso de tecnologías y equipos propietarios de la empresa NEC.

8. Conclusiones

Las redes definidas por software se presentan como el nuevo paradigma que definirá la futura arquitectura de las redes de datos incluyendo la Internet. Las etapas de migración y convivencia con el modelo convencional,

la viabilidad comercial, sostenibilidad, rentabilidad y confiabilidad del funcionamiento y rendimiento aún se encuentran en evaluación. Independiente de estas barreras las posibilidades y flexibilidad que brinda este modelo permitirá que las redes de datos incluyendo la Internet puedan soportar las demandas de nuevas tecnologías, aplicaciones y usuarios (exponencial) que se presentarán en el mediano y largo plazo. El desarrollo e implementación de este nuevo paradigma presenta también unos nuevos desafíos, como son la capacitación y actualización de los ingenieros de redes de datos a los nuevos conceptos y tecnologías (arquitectura de SDN, protocolo OpenFlow, algoritmos, lenguajes de programación y creación de módulos de software).

Algunos de los casos de éxito relacionados en este artículo permitirán comprender y comprobar (mediante vigilancia tecnológica) que el paradigma de las redes definidas por software es un modelo eficiente, versátil y confiable que se presenta como solución a las problemáticas en el modelo de red convencional.

Finalmente se debe comprender que es un paradigma abierto que permitirá generar desarrollos que no provengan únicamente de los proveedores de tecnología típicos, sino desde personas independientes, pequeñas y medianas empresas hasta semilleros o grupos de investigación al interior de las universidades, con el conocimiento y herramientas de programación como principal recurso para lograrlo.

Referencias

[1] Open Networking Foundation, *Sitio Web Oficial - https://www.opennetworking.com*.

[2] L. Yang, R. Dantu, T. Anderson, and R. Gopal, *"Forwarding and Control Element Separation (ForCES) framework"*, RFC 3746, 2004.

[3] Cisco Systems, *ÇCNP Switch 642-813 Official Certification Guide"*, 2010.

[4] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, *"The soft-router architecture"*, in Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking, November 2004.

[5] Organización EuChina Fire, *Sitio Web Oficial - http://www.euchina-fire.eu/onf-fall-plugfest-2014/.*

[6] NEC Networks, Casos de Éxito de Implementación de redes bajo el paradigma SDN *Sitio Web Oficial - http://www.nec.com/en/case/sub/prdslt$_p$43.html*.

[7] Á. L. Valdivieso Caraguay, A. Benito Peral, L. I. Barona López, L. J. García Villamba, *SDN: "Evolution and Opportunities in the Development IoT Applications*, December 2013.

[8] N. McKeown, T. Anderson, H. Balakrishnan et al, *"OpenFlow: Enabling innovation in campus networks"*, .ACM SIGCOMM Computer Communication Review, VOL, 38, PP, 69-74, 2008.

[9] A. Valdivieso, L. Barona, and L. Villalba, *"Evolution and Challenges of Software Defined Networking"*, in Proceedings of the 2013 Workshop on Software Defined Networks for Future Networks and Services, pp. 61-67, IEEE, November 2013.

[10] S. Sezer, S. Scott-Hayward, p. K. Chouhand et al, *"Are we Ready for SDN? Implementation Challenges for Software-Defined-Networks"*, in IEEE Communications Magazine, vol. 51, pp. 36-43, 2013.

[11] S. Kim and N. Feamster, *"Improving Network Maganement with Software Defined Networking"*, in IEEE Communications Magazine, vol. 51, pp. 114-119, 2013.

[12] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, *"Abstractions for Network Update"*, in ACM SIGCOMM Computer Communication Review, vol. 42, pp. 323-334, 2012.

[13] N. Gude, T. Koponen, J. Pettit et al, *"NOX: Towards an Operating System for Networks"*, ACM SIGCOMM Computer Communication Review, vol. 38, no. 3, pp. 105-110, 2008.

[14] D. Erickson, *"The Beacon OpenFlow Controller"*, in Proceedings of the 2nd ACM SIGCOMM Workshop on Hot topics in Software Defined Networking, pp, 13-18, ACM, August 2013.

[15] N. Foster, R. Harrison, M. J. Freedman et al, *"Frenetic: A Network Programming Language"*, ACM SIGPLAN Notices, vol. 46, no. 9, pp. 279-291, 2011.

[16] N. Foster, A. Guha, M. Eitblatt et al, *"Languages for Software Defined Networks"*, in IEEE Communications Magazine, vol. 51, pp. 128-134, 2013.

[17] C. Monsanto, N. Foster, R. Harrison, and D. Walker, *"A Compiler and Run-Time System for Network Programming Languages"*, ACM SIGPLAN Notices, vol. 47, no. 1, pp. 217-230, 2012.

[18] A. Voellmy, and J. Wang, *"Scalable Software Defined Network Controllers"*, in ACM SIGCOMM Computer Communication Review, vol. 42, pp. 289-290, 2012.

[19] ONF, *.ºpenFlow Switch Specification - version 1.5.0"*, Open Networking Foundation, diciembre 2014.

[20] L. Henao *"Guía de Implementación y Uso del emulador de Redes Mininet"*, Universidad Tecnológica de Pereira , 2015.

[21] Liu. Ting, *"Implementing Open Flow Switch Using FPGA Based Platform"*, Master Thesis, Norwegian University of Science and Tehnology - NTNU - Rondheim, June 2014.

[22] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, *"A Demonstration of Fast Failure Recovery in Software Defined Newtworking"*, in Testbeds and Research Infrastructure. Development of Networks and Communities, vol. 44, pp. 411-414, Springer, Berlin, Germany, 2012.

[23] Ola Salman and Imad H., Ayman Kayssi and Ali Chehab *"SDN Controllers: A Comparative Study"*, IEEE - Proceedings of the 18th Mediterranean Electrotechnical Conference MELECON 2016, Limassol, Cyprus, April 18-20, 2016.