Asociación Colombiana de Facultades de ingeniería

Revista Educación en Ingeniería
http://www.educacioneningenieria.org

Uso de Mininet y Openflow 1.3 para la enseñanza e investigación en redes IPv6 definidas por software

Line Yasmin Becerra-Sánchez [a], Bryan Valencia-Suárez [a], Santiago Santacruz-Pareja [a] & Jhon Jairo Padilla-Aguilar [b]

[a]*Facultad de Ciencias Básica e Ingeniería, Universidad Católica de Pereira, Pereira, Colombia. line.becerra@ucp.edu.co, bryan.valencia@ucp.edu.co, santiago.santacruz@ucp.edu.co*
[b]*Facultad de Ingeniería Electrónica, Universidad Pontificia Bolivariana, Bucaramanga, Colombia. jhon.padilla@upb.edu.co*

Resumen— Las redes definidas por software (SDN) son un nuevo paradigma en redes de datos que separan el plano de control del plano de datos, dicha separación proporciona grandes ventajas asociadas con el control y la gestión de redes; estas redes abren muchas oportunidades en investigación para proponer soluciones a los problemas existentes. Por otro lado, IPv6 es el protocolo de nueva generación y de la futura infraestructura global de Internet. El objetivo principal de este artículo es presentar una metodología sencilla para la emulación de una SDN configurada con el protocolo IPv6, usando OpenFlow 1.3 y el controlador RYU mediante la herramienta Mininet; esta combinación facilita la enseñanza e investigación en esta área. Para este propósito, en este documento se describe las características importantes de IPv6, SDNs el protocolo OpenFlow 1.3 y el controlador RYU. Además se describen los requerimientos y recomendaciones fundamentales para dicha emulación, y se presentan los resultados de pruebas sencillas con el fin de verificar la conectividad, transferencia de datos y operación sobre una topología de prueba.

Palabras Clave— mininet, openflow, redes definidas por software, RYU.

Recibido: 20 de abril de 2017. Revisado: 29 de junio de 2017. Aceptado: 7 de julio de 2017.

Use of Mininet and Openflow 1.3 for teaching and research in software defined IPv6 networks

Abstract— Software defined Network (SDN) is a new paradigm in data networks that separates the control plane from the data plane, which provides advantages associated with the control and management of networks; SDN opens many opportunities in researching to solve existing problems. On the other hand, IPv6 is the next generation protocol and the global internet infrastructure future. The main goal of this article is to present a simple methodology for the emulation of an SDN network configured with IPv6 protocol using OpenFlow 1.3 and RYU controller over Mininet tool; this is a combination facilities teaching and researching in this area. For this purpose, in this document the important features of SDNs, IPv6, OpenFlow 1.3 protocol and RYU controller are described. Similarly, the requirements and fundamental recommendations for such emulation, and the results of some tests in order to verify connectivity, data transfer and operating on a test topology are presented.

Keywords— mininet, openflow, software defined networks, RYU.

1. Introducción

IPv6 es un protocolo de gran importancia en las redes actuales debido a que Internet está migrando de la tecnología IPv4 hacia esta nueva versión del protocolo de Internet, conocida como IPv6. Esto se debe principalmente al agotamiento de las direcciones IPv4, pero también se introdujeron otras modificaciones que permiten una mayor eficiencia de los routers y una mayor seguridad, además de adaptarse a las características de los nuevos servicios de telecomunicaciones y a la movilidad de los usuarios.

De otro lado, las Redes Definidas por Software (SDN) son un nuevo paradigma que otorga nuevas características a las redes de datos, ya que hace que éstas puedan ser administradas y personalizadas a través de la programación de aplicaciones que, desde un controlador central, configuran el comportamiento de los dispositivos de conmutación. Los resultados de estudios de este nuevo paradigma y su funcionalidad con otros protocolos como IPv6 son muy significativos porque abren y posibilitan la creación de nuevas técnicas, herramientas software y algoritmos, entre otras opciones, para innovar y cambiar la administración y rendimiento de las redes convencionales. En Colombia, el estudio de las SDNs es incipiente, y su uso con IPv6 abre unas puertas de gran importancia para la investigación y aporte a nivel mundial de parte de nuestros grupos de investigación.

La meta principal de este artículo es presentar una metodología sencilla para la emulación de una red SDN configurada con el protocolo IPv6 usando Openflow 1.3 y el controlador RYU mediante la herramienta Mininet, lo cual es de suma importancia en la enseñanza y el aprendizaje del funcionamiento de este nuevo paradigma de las redes de datos. En la literatura se encuentran pocos trabajos alrededor de la combinación SDN-IPv6, a continuación se describirán algunos trabajos de investigación que se han desarrollado buscando esta convergencia.

En trabajos tales como [1], se presenta una arquitectura de comunicación de redes para SDN e IPv6. En [2] puede verse una propuesta para integrar IPv6 en SDN mediante mecanismos OpenFlow y en [3] los autores presentan la etapa inicial de un experimento para validar el soporte de OpenFlow para multicast en

Como citar este artículo: Becerra-Sánchez, L.Y., Valencia-Suárez, B., Santacruz-Pareja, S. and Padilla-Aguilar, J.J., Uso de Mininet y Openflow 1.3 para la enseñanza e investigación en redes IPv6 definidas por Software. Educación en Ingeniería, 12(24), pp. 89-96, Julio, 2017.

IPv6. En algunas publicaciones como en [4-6] pueden verse diferentes trabajos e investigaciones donde es utilizada la herramienta de emulación Mininet, en donde se puede evidenciar su gran utilidad y versatilidad para realizar estudios referentes a las redes definidas por software.

Por otro lado, el controlador RYU es realmente un Framework, lo que significa que a además de ser un controlador para SDN, está constituido por elementos como modelos, librerías, y otros recursos para el desarrollo de aplicaciones. Estas características y el hecho de ser de código libre lo convierten en una muy buena opción para la evaluación y emulación de SDNs.

Este documento está organizado de la siguiente manera: en la sección 2, se da una breve introducción y generalidades importantes sobre IPv6 y las Redes Definidas por Software; en la sección 3, se presentan las características de Mininet, RYU y en general de los elementos necesarios para la emulación de una SDN sobre Mininet. Posteriormente en la parte 4, se detalla el proceso y los pasos para emular un prototipo de red configurado con IPv6 y RYU; para esto se explican y muestran algunos elementos necesarios del protocolo OpenFlow para el soporte de IPv6, la utilización de Mininet y del controlador. En la sección 5, se muestran algunas pruebas realizadas en la emulación, y para finalizar, en la sección 6, se realizan algunas reflexiones y conclusiones alrededor del tema expuesto.

2. Protocolo IPv6 y redes definidas por software

2.1. Protocolo IPv6

Desde que la Internet viene siendo utilizada en gran medida y el número de equipos conectados a la red incrementa, la forma actual del Protocolo de Internet IPv4 [7] se ha quedado corta, pues no cuenta con el número suficiente de direcciones IP para identificar de forma única a cada dispositivo conectado a la red. En 1995 el grupo encargado de la IETF, recomendó la creación de IPv6, cuya especificación fue descrita en la RFC1752 [8] denominada "The Recommendation for the IP Next Generation Protocol". Luego esta recomendación fue evolucionando hasta la publicación de la especificación completa en 1998 con la RFC2460 [9].

IPv6 [9], tiene direcciones más grandes que el IPv4, son de 128 bits en oposición a los 32 bits respectivamente, lo que permite que existan muchas más direcciones asignables. Además de resolver el problema de la cantidad de direcciones, IPv6 simplifica el encabezado, que solo contiene 9 campos, a diferencia de IPv4 que contiene 13, esto posibilita que los enrutadores procesen mucho más rápido los paquetes y por tanto mejorar la velocidad de envío [10].

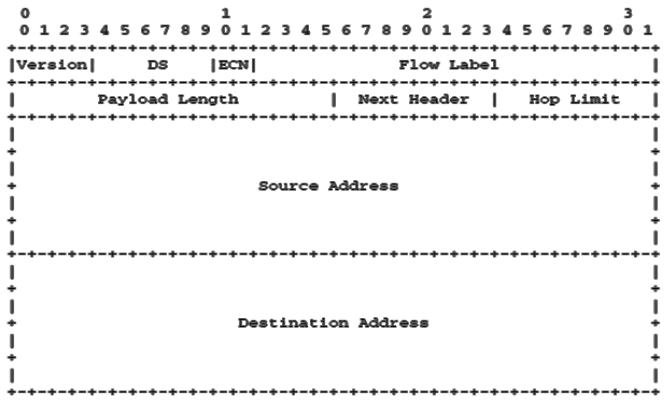

Figura 1. Cabecera IPv6.
Fuente: Adaptado de [1].

Como se puede apreciar en la Fig. 1, la mayoría de los campos en IPv6 fueron cambiados o modificados, el único campo que permanece sin cambio es el campo llamado "versión", esto se hizo para permitir a IPv4 e IPv6 coexistir en el mismo enlace local. El campo "Header length" de IPv4 es irrelevante para IPv6 porque todas las cabeceras IPv6 son de la misma longitud. El campo "Type of Service (ToS)/ Differentiated Services" de IPv4, ha cambiado significativamente desde que este fue especificado originalmente para etiquetar paquetes para diferentes clases de manejo de los paquetes por los routers. La RFC2474 [11], denominada "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", hace la especificación de este campo para ambos protocolos. Esta especificación requiere la sustitución del campo "ToS" de IPv4 y el campo "Traffic class" de IPv6 por el campo "DS. El campo "Total length" de IPv4, es el campo "payload length" en IPv6. El campo "Total lenght" de IPv4 específica la longitud del datagrama entero, incluyendo las cabeceras IP; de esta manera los routers pueden calcular la longitud de carga útil del datagrama IPv4 mediante la substracción de la longitud de la cabecera de la longitud del datagrama. En IPv6 este cálculo es innecesario, porque la longitud de carga útil de IPv6 incluye las cabeceras de extensión. El campo "Identification" es usado para identificar un datagrama como parte de un paquete fuente fragmentado. En IPv6 este campo no es necesario porque IPv6 no permite fragmentación de nodo intermedio. Los campos "Flags" y "Fragment offset" son también usados para permitir la fragmentación, por lo que tampoco se necesitan en IPv6. El campo "Time-to-live (TTL)" de IPv4, se transformó en el campo "hop limit" en IPv6. TTL fue originalmente concebido como límite superior del tiempo de vida (dado en segundos) de un paquete en Internet, sin embargo, se decidió interpretar como el número de saltos máximo que puede dar un paquete debido a que debe ser asignado un valor de tiempo de vida finito y entero que sea fácil de medir en diferentes routers. El campo "Protocol" se refiere al protocolo de la próxima capa más alta encapsulada dentro del paquete IPv4. Este campo se involucró en IPv6 dentro del campo "Next Header", donde este especifica la próxima cabecera, ya sea una cabecera de extensión IPv6 u otra cabecera de protocolos pertenecientes a la capa superior. El campo "Header checksum" es una aproximación robusta para evitar el procesamiento de paquetes que han tenido deterioros de las cabeceras en camino a su destino. Sin embargo, con los protocolos de capa superior como TCP y UDP que calculan sus propias sumas de comprobación sobre las cabeceras, entonces, el campo suma de comprobación de IPv4 fue considerada redundante y por tanto, desapareció en la cabecera IPv6. Los campos "Source/destination address", pasa de 32 bits para paquetes IPv4 a 128 bits para paquetes IPv6. El campo "IP option" en IPv4 es reubicado en IPv6. Las opciones de IPv6 existen como cabeceras separadas después de la cabecera principal pero antes de la carga útil del paquete.

Adicionalmente, se agregaron dos campos a IPv6, el campo "ECN" y el campo "Flow Label". El campo "ECN" es de dos bits y es usado como banderas de notificación de congestión explicita. Por su parte, el campo "Flow Label" es un valor usado para identificar paquetes pertenecientes al mismo flujo. La etiqueta de flujo y la dirección del nodo fuente podrían servir para identificar los flujos [10,12].

2.2. Redes definidas por software

La idea de las SDN fue surgiendo desde aproximadamente 1996 con acercamientos como las Active Networks [13], DCAN [14], NETCONF [15] y Ethane [16]. Los conceptos alrededor de las Redes Definidas por Software permiten actualizar las características de un router ubicado dentro de una red de datos IP. Los routers tradicionalmente han desempeñado dos tareas fundamentales: la primera relacionada con la toma de decisiones de enrutamiento para cada paquete que llega, y la segunda es el envío de estos paquetes con base en estas decisiones. Inmersos en un router pueden ser vistos dos planos, uno que permite la toma de decisiones (plano de control) y otro que maneja el reenvío de paquetes (plano de datos). Las Redes Definidas por Software se refieren precisamente a la separación del plano de datos y el plano de control con el fin de simplificar las tareas de enrutamiento, [17,18].

En un entorno normal SDN, el router envía el primer paquete que recibe de un flujo al controlador. Después de procesar la cabecera del paquete, el controlador carga entradas adecuadas en las tablas de flujo que están "a ciegas" y que son utilizadas por los routers para reenviar el resto de los paquetes que pertenecen al mismo flujo [19]. Las políticas de reenvío de un router pueden ser cambiadas por el controlador en función de cada flujo, lo que no es posible en el enfoque tradicional ya que la única manera de cambiar una política de enrutamiento predefinida sería alterando la configuración del dispositivo, que por lo general viene definida en un firmware. Esto, evidentemente, no es deseable ya que limita la escalabilidad y personalización de la red.

Es importante también tener en cuenta que un componente esencial de los routers SDN es el protocolo de comunicaciones que permite el intercambio de datos entre el controlador y sus routers. Se han propuesto varios protocolos de comunicación para este propósito. Entre estos protocolos, OpenFlow [20] ha ganado una considerable popularidad entre otras propuestas como ForCES [21].

Como se detalla en [22], la arquitectura de un router SDN está compuesta por tres capas: capa de infraestructura, capa de control y capa de aplicación. La Fig. 2, muestra, además de estas capas, las interfaces de comunicación entre ellas como la interfaz sur (southbound) de la que hace parte OpenFlow.

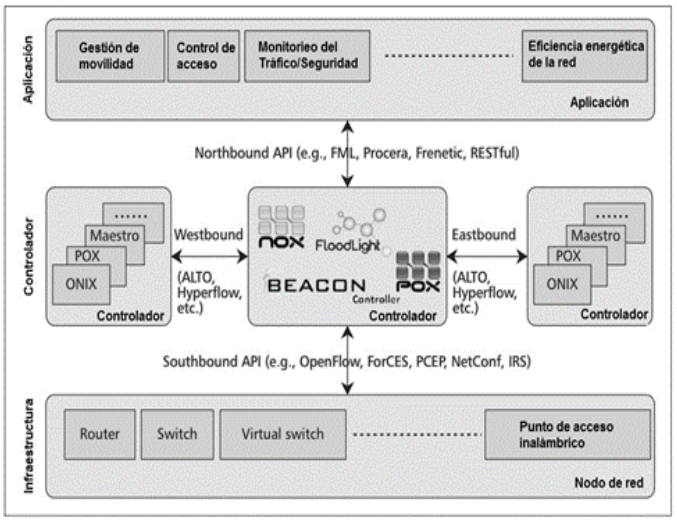

Figura 2. Arquitectura de una SDN.
Fuente: Adaptado de [23].

3. Emulación de redes definidas por software

La evaluación y la realización de pruebas de protocolos de red pueden realizarse a través de bancos de pruebas experimentales, simuladores y emuladores [2]. Cuando se utiliza emulación, el sistema a evaluar se representa con algunos elementos modificados pero otros se tratan exactamente igual que en un caso real, debido a esto, en la emulación todo se ejecuta en tiempo real [24].

Los emuladores y/o simuladores más utilizados actualmente para la evaluación de Redes Definidas por Software son ns-3 [25], EstiNet [26] y Mininet [27].

3.1. Mininet

Mininet es un emulador para el despliegue de redes sobre los limitados recursos de un ordenador sencillo simple o máquina virtual [6]. Éste utiliza el kernel de Linux y otros recursos para emular elementos de la SDN como el controlador, los switch OpenFlow y los host.

El funcionamiento de una SDN requiere de un conjunto de equipos de red (swtiches), un controlador, una aplicación software que se ejecute en el controlador, un software instalado en los switches y una coincidencia entre el protocolo y versión utilizada para la comunicación entre la capa de infraestructura y la de control, como por ejemplo OpenFlow. Mininet incorpora por defecto un controlador y un software switch para los procesos de emulación e instala en ellos la versión 1.0 de OpenFlow. Además instala y corre en el controlador una aplicación sencilla denominada *simple switch.*

Mininet también posibilita la generación de diferentes topologías de red entre las que pueden mencionarse las llamadas *minimal*, *single*, *linear* y *tree*. En cualquiera de éstas existe un controlador, varían en el número de hosts, switches y los enlaces entre éstos. La topología *minima*l, por ejemplo, está compuesta por dos hosts conectados a un switch; *single* también es una topología con un único switch pero la cantidad de host puede indicarse por medio de un parámetro [22].

Cabe resaltar en este punto que un controlador es básicamente un servidor que se encarga de la ejecución de aplicaciones de red que permitan gestionar y mantener la red a través de la comunicación de reglas de enrutamiento, seguridad y otros parámetros [1]. Por esto dentro de las características de Mininet está la utilización de controladores remotos, lo que significa que se puede realizar la emulación de una red que utilice un controlador que esté ejecutándose en la misma máquina o en cualquier parte del mundo [28]. Algunos de los controladores más conocidos son NOX, POX [29] [30], RYU [31], Foodlight [32] y Beacon [33].

3.2. Controlador RYU

RYU es un framework para Redes Definidas por Software basado en componentes y escrito completamente en Python. RYU ofrece elementos software con una API bien definida que hace fácil a los desarrolladores crear nuevas aplicaciones de control y gestión de red. RYU soporta varios controladores y dispositivos de gestión de red como OpenFlow, Netconf, OF-config, entre otros. Este controlador también admite completamente las versiones 1.0, 1.1, 1.2, 1.3, 1.4 y 1.5 de OpenFlow [34].

```
root@ubuntu:~# cd ryu
root@ubuntu:~/ryu# ./bin/ryu-manager --verbose ryu/app/rest_router.py
loading app ryu/app/rest_router.py
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu/app/rest_router.py of RestRouterAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK dpset
  PROVIDES EventDP TO {'RestRouterAPI': set(['dpset'])}
  CONSUMES EventOFPStateChange
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPPortStatus
BRICK RestRouterAPI
  CONSUMES EventOFPPacketIn
  CONSUMES EventOFPStatsReply
  CONSUMES EventOFPFlowStatsReply
  CONSUMES EventDP
BRICK ofp_event
  PROVIDES EventOFPPacketIn TO {'RestRouterAPI': set(['main'])}
  PROVIDES EventOFPFlowStatsReply TO {'RestRouterAPI': set(['main'])}
```

Figura 3. Ejecución del controlador RYU.
Fuente: Los autores.

Tabla 1
Campos de Ipv6 Soportados por Openflow 1.3 [41]

Campo	Descripción
OXM_OF_IPV6_SRC	Dirección IPv6 de origen.
OXM_OF_IPV6_DST	Dirección IPv6 de destino.
OXM_OF_IPV6_FLABEL	Etiqueta de flujo de IPv6
OXM_OF_ICMPV6_TYPE	Tipo de ICMPv6
OXM_OF_ICMPV6_CODE	Código de ICMPv6
OXM_OF_IPV6_ND_TARGET	La dirección objetivo en un mensaje de descubrimiento de vecinos en IPv6.
OXM_OF_IPV6_ND_SLL	La opción de dirección de capa de enlace del origen en un mensaje de descubrimiento de vecinos en IPv6.
OXM_OF_IPV6_ND_TLL	La opción de dirección de capa de enlace de destino en un mensaje de descubrimiento de vecinos en IPv6.
OXM_OF_IPV6_EXTHDR	Pseudo-campo de cabecera de extensión IPv6.

Fuente: Adaptado de [41]

Al ser una herramienta de código libre, cuenta con diferentes comunidades de ayuda como el *Mailing List* [35], manejo de repositorio de código fuente libre *GitHub* [36] y varios libros con información para el desarrollo de nuevas aplicaciones y el uso de las que incorpora de manera predeterminada.

La instalación de RYU puede realizarse de dos formas, directamente con el instalador de paquetes *pip*, en la que en el terminal se escribe la instrucción **pip install ryu,** o puede preferirse instalar RYU desde el código fuente; para eso se clona el código con el comando **git clone git://github.com/osrg/ryu.git** y luego se ejecuta la instalación con python de la siguiente manera **cd ryu; python ./setup.py install** [37]**.**

Para administrar y personalizar el comportamiento de los dispositivos de una red (switches, routers, etc.) debe escribirse una aplicación de RYU. La aplicación le dice a RYU cómo administrar los dispositivos de red a través de la configuración de los mismos utilizando el protocolo OpenFlow. Estas aplicaciones deben ser codificadas en lenguaje python [38].

En la Fig. 3, puede verse la ejecución del controlador RYU en el terminal del sistema operativo Ubuntu. Puede observarse en la tercera línea que el controlador es inicializado con una aplicación denominada rest_router.py.

De manera general puede entonces decirse que para emular una red es necesario correr un controlador que a su vez ejecute una aplicación, ejecutar Mininet con una topología y asegurarse de la utilización de un software switch. Opcionalmente si fuese necesario la utilización de una versión específica de OpenFlow debe comprobarse que la versión sea soportada tanto por el controlador, la aplicación y el software switch.

4. Emulación del protocolo IPv6 en una red definida por software en Mininet usando el controlador RYU

Para lograr una emulación de una SDN que emplee IPv6, además de utilizar una versión de OpenFlow que admita y reconozca todos los campos de la cabecera de este protocolo, es necesario comprobar que el *software switch* utilizado también haga uso de la misma versión de OpenFlow. El protocolo OpenFlow, permite la comunicación entre los dispositivos de red y el controlador e identifica los campos de la cabecera IP para programar y crear aplicaciones que hagan uso de estos. Dependerá de la versión de este protocolo el soporte para IPv6 y los campos específicos de su cabecera.

Por otra parte, las aplicaciones del controlador cumplen un papel fundamental en las SDN, pues son en éstas donde se encuentra la programación y configuración de la red. La aplicación que se instala en el controlador también debería estar codificada de tal manera que se aprovechen los campos específicos de la cabecera de IPv6, para lo que sería necesario hacer uso en su desarrollo de la versión específica de OpenFlow que lo permita.

4.1. Openflow para soportar IPv6

La versión de OpenFlow 1.0 [39] sólo soporta campos de la cabecera de IPv4, mientras que OpenFlow 1.2 [3] ya provee algún soporte para IPv6. Por su parte, en la versión OpenFlow 1.3 [41], un switch puede especificar reglas de coincidencia utilizando elementos de IPv6 como: dirección IPv6 de origen y destino, Protocolo de Mensajes de control de Internet versión 6 (ICMPv6), cabeceras de extensión IPv6, entre otros. En la Tabla 1, pueden verse de manera detallada los campos de IPv6 soportados por la versión 1.3 de OpenFlow.

4.2. Emulación de una SDN con IPv6 y usando el controlador RYU

De manera general, las herramientas y elementos necesarios para emular una Red Definida por Software que trabaje con el protocolo IPv6 mediante Mininet y RYU son:

- Mininet debe estar instalado en una máquina Ubuntu o Fedora. Los procesos de instalación pueden consultarse en [42]. Dos de las formas más utilizadas de instalar y utilizar Mininet son a través de una máquina virtual configurada con esta herramienta o realizando la instalación de manera nativa.
- Tener RYU instalado en una máquina con conexión al computador donde se instaló Mininet o instalarlo en el mismo. El proceso de instalación de RYU puede verse en [37].
- Configurar y constatar el soporte de OpenFlow 1.3 por el software switch seleccionado y el controlador.
- Configurar las interfaces de cada host de la topología con direcciones IPv6.

Para el caso puntual que aquí se presenta, la emulación se realiza con Mininet instalado de manera nativa en una máquina virtual

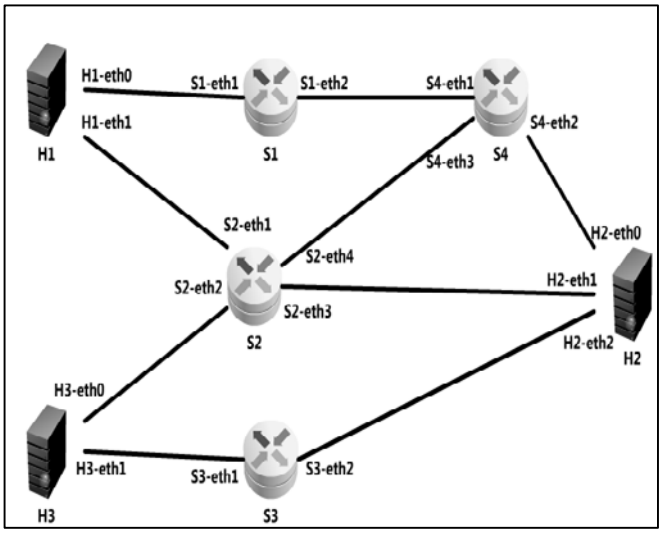

Figura 4. Topología de SDN emulada con IPv6.
Fuente: Los autores.

Tabla 2.
Direcciones Ipv6 Asignadas a la Topología Emulada.

HOST	INTERFAZ	DIRECCIÓN IPv6
H1	h1-eth0	fc00::1/64
H1	h1-eth1	fc00::2/64
H2	h2-eth0	fc00::3/64
H2	h2-eth1	fc00::4/64
H3	h2-eth2	fc00::5/64
H3	h3-eth0	fc00::6/64
H3	h3-eth1	fc00::7/64

Fuente: Los autores.

Ubuntu 12.04, donde se codificó una topología personalizada en python para lograr realizar la configuración de red mostrada en la Tabla 1. La línea de comando utilizada para iniciar la topología es la siguiente: **sudo mn –custom topoIPv6.py --topo mytopo --switch ovsk --controller remote.** Donde "***topoIPv6.py***" es el nombre del script de python donde se codificó la topología personalizada. Nótese también que se hace uso de un controlador remoto.

Antes de esta ejecución es necesaria la asignación de la versión 1.3 de OpenFlow al software switch utilizado, en este caso el llamado Open vSwitch [43]. Una opción para esto es utilizar la función *xterm* y en cada switch escribir la siguiente instrucción: **ovs-vsctl set Bridge s1 protocols=OpenFlow13**.

Posteriormente se asignan direcciones IPv6 a cada una de las interfaces de red. Para este caso las direcciones utilizadas, que pueden ser vistas en la Tabla 2, son del tipo Unicast Local Única [44]. Estas direcciones se utilizan para comunicaciones locales.

En Mininet la asignación de direcciones IPv6 se realiza con la instrucción:

<identificadorHost> ifconfig <identificadorInterfaz> inet6 add <direcciónIPv6AAsignar>. En la Fig. 4, se muestra un ejemplo con varias de estas configuraciones.

Como se comentó anteriormente, el controlador RYU puede ser inicializado dentro de la misma máquina donde se corre Mininet. Un ejemplo de esto es ilustrado en la Fig. 5. Para el caso que se presenta, se utilizó una aplicación de RYU que se llama *simple_switch_13.py*. El número al final indica que esta aplicación utiliza la versión 1.3 de OpenFlow.

```
*** Starting CLI:
mininet> h1 ifconfig h1-eth0 inet6 add fc00::1/64
mininet> h1 ifconfig h1-eth1 inet6 add fc00::2/64
mininet> h2 ifconfig h2-eth0 inet6 add fc00::3/64
mininet> h2 ifconfig h2-eth1 inet6 add fc00::4/64
mininet> h2 ifconfig h2-eth2 inet6 add fc00::5/64
mininet> h3 ifconfig h3-eth0 inet6 add fc00::6/64
mininet> h3 ifconfig h3-eth1 inet6 add fc00::7/64
```

Figura 5. Asignación de direcciones IPv6 en Mininet.
Fuente: Los autores.

Esta aplicación hace que los switches se comporten como hubs; OpenFlow ha denominado a este comportamiento Learning Switch. Con esta aplicación el switch examinará cada paquete y aprenderá el puerto origen que le corresponde. A partir de este momento, la dirección MAC origen será asociada con ese puerto. Si el destino de un paquete ya está asociado a algún puerto, el paquete será enviado al puerto dado, de lo contrario se inundarán todos los puertos del switch con éste [45].

En general, lo que hace esta aplicación es que a medida que el switch va recibiendo tramas, va creando una tabla donde se relacionan direcciones MAC con puertos, esto se debe a que el controlador enviará un mensaje OpenFlow al switch instaurando una nueva entrada en la tabla de flujo [46].

5. Pruebas

Se realizaron algunas pruebas sencillas con el fin de comprobar conectividad, transferencia de datos y funcionalidad de la red configurada. Como experimento inicial se realizaron pruebas de conectividad con *ping*, con el fin de verificar que los host se conectan a la red y a sus recursos. En la Fig. 6, se muestran estas pruebas entre el host 1 y el host 2 para comprobar conectividad en la red. De 17 paquetes enviados, fueron recibidos en su totalidad los 17.

Una herramienta muy útil para poder comprobar la versión de openflow que se está trabajando y otras funcionalidades es Wireshark [47]. Con Wireshark se puede verificar si quedó bien instalado openFlow 1.3 y si se está trabajando con dicha versión, esta comprobación es importante teniendo en cuenta que Mininet trabaja con openflow 1.0 por defecto y esta versión no soporta IPv6. Wireshark también facilita la comprobación de cualquier proceso de envío de información, protocolos usados y pruebas que se realicen en la red.

En otras pruebas, y con ayuda de iPerf [48] se crearon flujos UDP entre los dos host tratados en la prueba anterior para tomar algunas mediciones como el Jitter, la pérdida de datagramas y el ancho de banda. Como se mostró en la explicación de la aplicación software utilizada en el administrador, las tablas de flujo de los swiches se van completando a medida que el controlador envía mensajes a través de OpenFlow para este fin. La Tabla 3, muestra los resultados de la prueba con iPerf en el primer envío de datos (cuando las tablas de flujo están vacías) como también después de que éstas están completas.

Para el caso específico de estas muestras, iPerf se configuró para utilizar UDP con un ancho de banda de 10 Mbits/s y con un total de muestras en 10 intervalos.

```
mininet> h1 ping6 fc00::3 -I h1-eth0
PING fc00::3(fc00::3) from fc00::1 h1-eth0: 56 data bytes
64 bytes from fc00::3: icmp_seq=1 ttl=64 time=229 ms
64 bytes from fc00::3: icmp_seq=2 ttl=64 time=0.337 ms
64 bytes from fc00::3: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from fc00::3: icmp_seq=4 ttl=64 time=0.076 ms
64 bytes from fc00::3: icmp_seq=5 ttl=64 time=0.045 ms
64 bytes from fc00::3: icmp_seq=6 ttl=64 time=0.072 ms
64 bytes from fc00::3: icmp_seq=7 ttl=64 time=0.074 ms
64 bytes from fc00::3: icmp_seq=8 ttl=64 time=0.079 ms
64 bytes from fc00::3: icmp_seq=9 ttl=64 time=0.077 ms
64 bytes from fc00::3: icmp_seq=10 ttl=64 time=0.079 ms
64 bytes from fc00::3: icmp_seq=11 ttl=64 time=0.074 ms
64 bytes from fc00::3: icmp_seq=12 ttl=64 time=0.109 ms
64 bytes from fc00::3: icmp_seq=13 ttl=64 time=0.079 ms
64 bytes from fc00::3: icmp_seq=14 ttl=64 time=0.073 ms
64 bytes from fc00::3: icmp_seq=15 ttl=64 time=0.076 ms
64 bytes from fc00::3: icmp_seq=16 ttl=64 time=0.071 ms
64 bytes from fc00::3: icmp_seq=17 ttl=64 time=0.077 ms
^C
--- fc00::3 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16001ms
rtt min/avg/max/mdev = 0.045/13.604/229.798/54.048 ms
```

Figura 6. Prueba de conectividad con *ping*.
Fuente: Los autores.

Tabla 3
Resultados de las Muestras Obtenidas con Iperf.

Estado tablas de flujo	Parámetros evaluado	Resultados
Incompletas	Transferencia	4.31 MBytes
	Jitter	1432.044 ms
	Porcentaje de datagramas perdidos	1082 (26%)
	Total Datagramas	4155
Completas	Transferencia	5.96 MBytes
	Jitter	0.032 ms
	Porcentaje de datagramas perdidos	0 (0%)
	Total Datagramas	4249

Fuente: Los autores.

Como puede observase, no hubo pérdida de datagramas y hubo un jitter muy pequeño en el test realizado cuando los switches contaban con sus tablas de flujo configuradas, por el contrario en el proceso llevado a cabo con las tablas incompletas hubo una pérdida de datagramas del 26% y un Jitter de 1432.044ms.

6. Conclusiones

En este artículo se presentó el uso del controlador RYU para la emulación de redes IPv6 en SDNs, usando la herramienta Mininet y el protocolo Openflow 1.3. Se proporciona una metodología sencilla y recomendaciones importantes que facilitan el desarrollo emulaciones en el estudio de IPv6 y SDNs. La combinación del controlador RYU para el estudio de IPv6 sobre redes definidas por software no ha sido explotada en la literatura actual, por tanto esto hace que sea un aporte importante para el desarrollo de investigaciones en el tema. RYU permite de forma predeterminada darle a los dispositivos de la capa de infraestructura un comportamiento de switch de aprendizaje (Learning Switch), el cual fue usado en la emulación presentada. Es importante resaltar aquí, que para generar otro tipo de comportamientos en la red y personalizar otros elementos es necesaria la codificación de aplicaciones para situaciones específicas. Esto permite también descubrir que la personalización, no sólo del enrutamiento, sino de las políticas de seguridad, rendimiento, ingeniería de tráfico, etc., es realmente posible pero dependerá de la construcción de aplicaciones propias y ajustadas a las necesidades de una organización o a procesos de investigación. Se pudo comprobar la facilidad de uso y se observaron las diferentes funcionalidades que tiene este conjunto de herramientas para implementar emulaciones de redes IPv6.

Por otro lado, en esta investigación se encontró que el protocolo Openflow 1.3 ofrece la ventaja de poder manejar diferentes campos de la cabecera IPv6, estos campos están descritos en la Tabla 1. El manejo de dichos campos abre un sinnúmero de oportunidades en investigaciones y trabajos futuros. Por ejemplo, en IPv6 se han realizado muchas propuestas para el uso de la etiqueta de flujo IPv6 para diferentes propósitos, las cuales están descritas en [12]; la metodología propuesta aquí, junto con la combinación con la herramienta Mininet, el controlador RYU y Openflow 1.3, constituye un aporte fundamental que permitiría hacer pruebas para las propuestas de investigación donde se manipula el contenido de la etiqueta de flujo IPv6, agregando algunas APIS sencillas para situaciones específicas de estudio. Trabajos futuros podrían estar enfocados a la emulación de funcionalidades específicas del protocolo IPv6 sobre SDNs que proporcionen por ejemplo nuevas soluciones independientes de calidad de servicio o ingeniería de tráfico en internet [49], en movilidad con IPv6 móvil o IPv6 móvil jerárquico [50] o en combinación con otras tecnologías como de MPLS [51].

Documentación como [52] publicada por RYU, las comunidades de ayuda en línea mediante el *Mailing List* [35] y el manejo de repositorio de código fuente libre *GitHub* [36] son algunas de las características que demuestran que el controlador RYU representa una gran oportunidad para la evaluación y despliegue de Redes Definidas por Software. En trabajos como [4], [53] y [54], se evidencian ejemplos de su uso y aportes específicos en el desarrollo y avance de este nuevo paradigma. También cabe resaltar que al ser RYU un completo marco de trabajo proporciona todo lo necesario para la creación de aplicaciones y posibilita la apropiación y entendimiento global del papel del controlador en SDN.

Referencias

[1] Tseng, C.W., Yang Y.T. and Chou L.D., An IPv6-enabled software-defined networking architecture, 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), Hiroshima, 2013.

[2] Tseng, C.W., Chen, S.J., Yang, Y.T., Chou, L.D., Shieh C.K. and Huang, S.W., IPv6 operations and deployment scenarios over SDN of Asia-Pacific, Network Operation and Management Symposium (APNOMS) 2014, Asia, 2014.

[3] Newman, D., Technology validation experiment: IPv6 and multicast support on OpenFlow, 2014. [online]. [Acceded: 08 mayo 2015]. Available at: http://users.ecs.soton.ac.uk/drn/ofertie/tve_ipv6_and_multicast.pdf

[4] Olaya-Yandun, M.E., Diseño e implementación de una aplicación para balanceo de carga para una Red Definida por Software (SDN), Quito, Tesis, Escuela Politécnica Nacional, 2015.

[5] Santos, R., Schweitzer C., Shinoda A. and Rodrigues L., Using MiniNet for emulation and prototyping Software-Defined Networks, IEEE Colombian Conference on Communications and Computing (COLCOM), Bogotá, pp. 1-6, 2014, DOI: 10.1109/ColComCon.2014.6860404.

[6] Kaur, K., Singh, J. and Ghumnan, N., MiniNet as Software Defined Networking testing platform, International Conference on Communication, Computing & Systems (ICCCS–2014), 2014.

[7] Postel, J., Internet Protocol, IETF RFC791, 1981.

[8] Bradner, S. and Mankin, A., The recommendation for the IP next generation protocol, IETF RFC1752, 1995.

[9] Deering, S. and Hinden, R., Internet Protocol Version 6 (IPv6) specification, IETF RFC2460, 1998.

[10] Loshin, P., IPv6: Theory, protocol and practice, Elseiver Morgan Kaufmann Publishers, pp. 123-140, 2004.

[11] Nichols, K., Blake, S., Baker F. and Black, D., Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 Headers, IETF RFC2474, 1998.

[12] Becerra, L.Y. and Padilla, J.J., Review of approaches for the use of the label flow of IPv6 header, IEEE Latin America Transactions, 12(8), 6 P., 2014.

[13] Tennenhouse, D. and Wetherall, D., Towards an active network architecture, Proceedings DARPA Active Networks Conference and Exposition, 2002, pp. 2-15, 1996.

[14] University of Cambridge, DCAN project devolved control of ATM Networks, [online]. [acceded: 2015 mayo 2015]. Available at: https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/dcan.

[15] Enns, E.R., NETCONF Configuration Protocol, IETF RFC 4741, December 2006.

[16] Casado, M., Freedman, M., Pettit, J., Luo, J., McKeown N. and Shenker, S., Ethane: Taking control of the enterprise, ACM SIGCOMM Computer Communication Review, 37(4), pp. 1-12, 2007. DOI:10.1145/1282427.1282382.

[17] Heller, S. and McKeown, The controller placement problem, ACM Hot SDN, 2012. DOI:10.1145/2342441.2342444.

[18] Hasan, S.F., A discussion on Software-Defined handovers in hierarchical MIPv6 networks, 10th IEEE Conference on Industrial Electronics and Applications, Auckland, 2015. DOI: 10.1109/ICIEA.2015.7334099.

[19] Hasan, S.F., Emerging trends in communication networks, Palmerston North: Springer, 2014.

[20] Rexford, J. et al., OpenFlow: Enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, 38(2), pp. 69-74, 2008. DOI:10.1145/1355734.1355746.

[21] Haleplidis, E., Salim, J., Halpern, J., Hares, S., Pentikousis, K., Ogawa, K., Weiming, W., Denazis S. and Koufopavlou, O., Network programmability with ForCES, Communications Surveys & Tutorials, IEEE, 17(3), pp. 1423-1440, 2015. DOI: 10.1109/COMST.2015.2439033.

[22] Becerra, L.Y., Valencia, B., Santacruz, S. and Padilla, J.J., Mininet: Una herramienta para el prototipado y emulación de Redes Definidas por Software. Entre Ciencia e Ingeniería, 9(17), pp. 62-70, 2015.

[23] Sezer, S., Scott, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M. and Rao, N., Are we ready for SDN? Implementation challenges for software-defined networks, IEEE Communications Magazine, 51(7), pp. 36-43, 2013. DOI: 10.1109/MCOM.2013.6553676.

[24] Saldaña, J.M., Murillo, J., Julián, F.N., Ruiz-Mas J. and Viruete, E.A., Emulación de escenarios de red mediante un testbed, 2014. [En línea]. [Último acceso: December 6th 2015]. Disponible en: http://diec.unizar.es/~jsaldana/personal/testbed_URSI_2010_in_proc.pdf

[25] ns-3, [online]. [acceded: may 30th 2015]. Available at: https://www.nsnam.org/.

[26] EstiNet, EstiNet Technolgies, 2015. [online]. [acceded: may 30th 2015]. Available at: http://www.estinet.com/.

[27] MiniNet, MiniNet: An Instant Virtual Network on your Laptop (or other PC) 2015. [online]. [acceded: may 26th 2015]. Available at: www.mininet.org.

[28] Mininet, Using a remote controller, [online]. [acceded: may 31th 2015]. Available at: http://mininet.org/walkthrough/#using-a-remote-controller.

[29] NOX, 2015. [online]. [acceded: may 30th 2015]. Available at: http://www.noxrepo.org/.

[30] Gude, N., Pfaff, B., Koponen, T., Casado, M., Shenker, S., Pettit, J. and McKeown, N., NOX: Towards an operating system for networks, Computer Communication Review, 38(3), pp. 105-110, 2008. DOI:10.1145/1384609.1384625.

[31] RYU, Component-based software defined networking framework. Build SDN Agilely, 2014. [online]. [acceded: may 30th 2015]. Available at: http://osrg.github.io/ryu/.

[32] Floodlight, Floodlight OpenFlow controller, [online]. [acceded: may 30th 2015]. Available at: http://www.projectfloodlight.org/floodlight/.

[33] Erickson, D., The beacon OpenFlow controller, [online]. [acceded: may 31th 2015]. Available at: http://yuba.stanford.edu/~derickso/docs/hotsdn15-erickson.pdf.

[34] RYU, RYU 25 agosto 2015. [online]. [acceded: August 25th 2015]. Available at: http://osrg.github.io/ryu/.

[35] RYU, RYU Devel, [online]. [acceded: December 6th 2015]. Available at: https://lists.sourceforge.net/lists/listinfo/ryu-devel.

[36] RYU, RYU component-based software defined networking framework, 2013. [online]. [acceded: December 6th 2015]. Available at: https://github.com/osrg/ryu.

[37] RYU, Getting Started, [online]. [acceded: December 7th 2015]. Available at: https://ryu.readthedocs.org/en/latest/getting_started.html.

[38] RYU, The first application, 2014. [online]. [acceded: December 7th 2015]. Available at: http://ryu.readthedocs.org/en/latest/writing_ryu_app.html.

[39] Open Networking Foundation, OpenFlow switch specification Version 1.0.0, 2009. [online]. [acceded: December 8th 2015]. Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf.

[40] Open Networking Foundation, OpenFlow switch specification Version 1.2.0, 2010. [online]. [acceded: December 8th 2015]. Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.pdf.

[41] Open Networking Foundation, OpenFlow switch specification Version 1.30,2012. [online]. [acceded: December 9th 2015]. Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.

[42] Mininet, Download/Get Started with Mininet, [online]. [acceded: may 30th 2015]. Available at: http://mininet.org/download/.

[43] Open vSwitch, Open vSwitch, 2010. [online]. [acceded: August 27th 2015]. Available at: http://openvswitch.org/.

[44] Hinden, R., Unique local IPv6 unicast addresses, IETF RFC4193, 2005.

[45] Create a learning switch 2015. [online]. [acceded: September 27th 2015]. Available at: https://github.com/mininet/openflow-tutorial/wiki/Create-a-Learning-Switch.

[46] Roncero, O., Software defined networking 2014. [online]. [acceded: May 08th 2015]. Available at: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf.

[47] Team, W., Wireshark [online]. [acceded: February 02nd 2015]. Available at: www.wireshark.org.

[48] , iPerf - The network bandwidth measurement tool, [online]. [acceded: December 08th 2015]. Available at: https://iperf.fr/.

[49] Becerra, L.Y. y Padilla, J.J., Estudio de propuestas para soportar ingeniería de tráfico en Internet, Entre Ciencia e Ingeniería, 6(11), pp. 53-76, 2012.

[50] Becerra, L.Y. y Padilla, J.J., (HMIPv6-BI) Propuesta de modificación al protocolo HMIPv6 para mejorar el ancho de banda en el canal radio, Entre Ciencia e Ingeniería, 3(5), pp. 72-91, 2009.

[51] Rosen, E., Viswanathan, A. and Callon, R., Multiprotocol label switching architecture, RFC3031, 2001.

[52] RYU project team, RYU SDN Framework, 2014. [online]. [acceded: December 20th 2015]. Available at: http://osrg.github.io/ryu-book/en/Ryubook.pdf.

[53] Fernandez, C., and Muñoz, J., Software Defined Networking (SDN) with OpenFlow 1.3, Open vSwitch and Ryu, 2015. [online]. [acceded: December 20th 2015]. Available at: http://upcommons.upc.edu/bitstream/handle/2117/77684/sdnbook.pdf. zip.

[54] Lopez-Rodriguez, F. and Campelo, D., A robust SDN network architecture for service providers, Global Communications Conference (GLOBECOM), Austin, 2014. DOI: 10.1109/GLOCOM.2014.7037086.

L.Y. Becerra-Sánchez, es Ing. Electrónica de la Universidad Pontificia Bolivariana (1999). Esp. en Telecomunicaciones de la Universidad Pontifica Bolivariana (2005). MSc. de la Universidad Pontificia Bolivariana (2009). Actualmente es estudiante de doctorado en ingeniería en el área de

telecomunicaciones de la misma universidad, es docente de la Universidad Católica de Pereira y pertenece al Grupo de Investigación Entre Ciencia e Ingeniería. Sus áreas de interés son: ingeniería de tráfico, enrutamiento, redes móviles IP, MIPv6, HMIPv6, simulación de redes, Internet, IPv6.
ORCID: 0000-0003-0514-3919

B. Valencia-Suárez, es Ing. de Sistemas y Telecomunicaciones de la Universidad Católica de Pereira-UCP (2016). Fue integrante del semillero de investigación inscrito al Grupo de Investigación GEMA-UCP en el cual, participó en un proyecto sobre la utilización de realidad aumentada como herramienta en la enseñanza-aprendizaje de geometría. Actualmente hace parte del Semillero de Investigación en Telecomunicaciones (SIT) inscrito al Grupo de Investigación Entre Ciencia e Ingeniería de la Universidad Católica de Pereira. Labora en VC@Soft: Bogotá, Cundinamarca, Colombia, como Developer BPM – DataPower. Sus áreas de interés son: redes definidas por software, IPv6, desarrollo de aplicaciones para dispositivos móviles.
ORCID: 0000-0003-0747-2601

S. Santacruz-Pareja, es Ing. de Sistemas y Telecomunicaciones de la Universidad Católica de Pereira (2016). Fue integrante del semillero de investigación inscrito al Grupo de Investigación GEMA participó en un proyecto sobre la utilización de realidad aumentada como herramienta en la enseñanza-aprendizaje de geometría. Actualmente hace parte del Semillero de Investigación en Telecomunicaciones (SIT) inscrito al Grupo de Investigación Entre Ciencia e Ingeniería de la Universidad Católica de Pereira. Labora en E-global: Medellín, Colombia como Ingeniero de soporte (Antioquia). Sus áreas de interés son: redes definidas por software, IPv6, enrutamiento, redes de datos.
ORCID: 0000-0003-2024-171X

J.J. Padilla-Aguilar, es Ing. Electrónico de la Universidad del Cauca (1993). Obtuvo su grado de MSc. en Informática de la Universidad Industrial de Santander (1998) y es Dr. en Ingeniería Telemática por la Universidad Politécnica de Cataluña, España, (2008). Actualmente es docente de la Facultad de Ingeniería Electrónica de la Universidad Pontificia Bolivariana y coordina el Grupo de Investigación en Telecomunicaciones (GITEL) de dicha universidad. Sus áreas de interés son: ingeniería de tráfico, Internet, calidad de servicio en Internet, redes inalámbricas, IPv6.
ORCID: 0000-0002-8552-2873