Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 2, pp. 21-33, Diciembre 2017
Selección de los Mejores Artículos de CoNCISa 2017

Lineamientos para el Despliegue de Redes SDN/OpenFlow

Gustavo Pereira [1], Eric Gamess [2]
gustavo.pereira.salas@gmail.com, egamess@jsu.edu

[1] Escuela de Computacion, Universidad Central de Venezuela, Caracas, Venezuela
[2] Department of Mathematical, Computing, and Information Sciences, Jacksonville State University, Jacksonville, AL, USA

Resumen: SDN (Software Defined Networking) es una arquitectura de red emergente que separa el plano de control del plano de datos de los dispositivos de la red y coloca el plano de control en uno o varios servidores de control capaz(ces) de gestionar las reglas de reenvío de tráfico de todos los dispositivos de comunicación bajo su(s) dominio(s). Este cambio de paradigma ofrece grandes beneficios en comparación con los métodos de red tradicionales debido a que el aprovisionamiento de políticas y servicios se efectúa en una sola entidad central, simplificando la administración, reduciendo los gastos de operación, acelerando el ciclo de innovación de nuevas tecnologías y ofreciendo la posibilidad de programar el comportamiento de la red. Esta investigación describe los principales componentes de una arquitectura SDN incluyendo componentes de hardware, software y protocolos y las consideraciones de diseño para el despliegue de redes SDN en el ámbito de campus empresariales. Las redes de campus empresariales están delimitadas a un conjunto de edificios o pisos de una edificación interconectados mediante redes Ethernet.

Palabras Clave: SDN; OpenFlow; Controladores SDN; Modelos de Despliegue SDN; Lineamientos Técnicos y Estratégicos; Fases para el Despliegue de Red SDN.

Abstract: SDN (Software Defined Networking) is an emerging network architecture that separates the control plane from the data plane of the network devices and places the control plane in one or several control servers that manage(s) the rules for traffic forwarding of all the communication devices under its/their domain(s). This new paradigm offers great benefits compared to traditional network methods, because the provision of policies and services is carried out in a single central entity, hence simplifying administration, reducing operating expenses, accelerating the innovation cycle of new technologies and offering the possibility of programming the behavior of the network. This research describes the main components of an SDN architecture including hardware components, software and protocols, and the design considerations for the deployment of SDN networks in business campuses. Business campus networks are limited to a set of buildings or floors of a building, interconnected by Ethernet networks.

Keywords: SDN; OpenFlow; SDN Controllers; SDN Deployment Models; Technical and Strategic Guidelines; Phases to Deploy SDN Networks.

I. Introducción

Un entorno de campus empresarial está conformado por un conjunto de edificios interconectados a través de medios de transmisión guiados y no guiados que enlazan los dispositivos de comunicación de la red y permiten el transporte del tráfico de los usuarios. En un campus empresarial tradicional el plano de control se encuentra distribuido en todos los dispositivos de la red y requiere la coordinación de los mismos para decidir cómo tratar los paquetes que ingresan a sus puertos. Esta naturaleza distribuida del plano de control obliga a los administradores de la red a configurar todos los dispositivos de red cada vez que se incorpora una nueva aplicación o política a la red, ralentizando el aprovisionamiento y la escalabilidad de servicios en ambientes con gran cantidad de dispositivos. Esta limitación ha impulsado a la comunidad científica a buscar nuevos enfoques y arquitecturas de redes alternativas que mejoren la agilidad, la flexibilidad y la escalabilidad en las redes de campus empresariales.

En un entorno SDN (Software Defined Networking) [1] uno o varios controladores se encargan de operar y gestionar el tráfico de la red desde un nodo lógico central. El controlador SDN instruye a los dispositivos SDN las reglas que se deben aplicar a los paquetes que circulan en la red en base a las políticas de una organización.

El despliegue de redes SDN requiere la aplicación de las mejores prácticas y principios de diseño que satisfagan los requerimientos de diseño de las organizaciones. Las mejores prácticas resultan de experiencias propias y de una investigación documental que abarca las siguientes categorías: (1) artículos académicos que describen la arquitectura de red SDN y el protocolo de red OpenFlow, (2) documentación de

implementaciones SDN en campus empresariales y (3) evaluaciones presentadas en artículos y notas técnicas.

II. Trabajos Relacionados

Los estudios relacionados son de carácter general y se encuentran algunos casos de estudio de implementaciones SDN en campus empresariales. La Universidad de Stanford [2][3] efectuó en el año 2010 una migración de parte de su red a una red SDN OpenFlow. El estudio de la migración a SDN en la Universidad de Stanford se divide en dos secciones: la primera sección muestra la red de arranque, las premisas de diseño y las herramientas de monitoreo utilizadas durante la migración. La segunda sección describe las fases de migración en la red de producción de los edificios considerados en la implementación.

Skorupa y Fabbi [4] describen los modelos de despliegue SDN existentes dependiendo de las características de la red de una organización.

La selección del tipo de controladores SDN y los dispositivos SDN para un despliegue se obtiene a partir de la revisión de las especificaciones de los productos abiertos y de carácter comercial disponibles por sus desarrolladores.

Los aspectos generales de diseño se encuentran en publicaciones y estudios [5][6] y textos de referencia [7][8] incluyendo principios de escalabilidad, alta disponibilidad, gestión y seguridad.

III. SDN

SDN [9] es un nuevo enfoque en la programación de redes que consiste en la capacidad de inicializar, controlar, cambiar y gestionar el comportamiento de reenvío del tráfico de una red mediante APIs abiertas. En una red SDN se separan los planos de control y datos de los dispositivos de red y se desplaza el plano de control a una unidad central llamada controlador SDN. Un controlador SDN se encarga de definir y comunicar las reglas de reenvío de tráfico a los dispositivos SDN y abstraer la infraestructura de red y su topología a las aplicaciones. Bajo este modelo las aplicaciones consideran a la red como un solo switch lógico central que provee servicios de conectividad a los usuarios y a las aplicaciones. Al separar los planos de datos y control, los switches de la red se convierten en dispositivos de reenvío simples y la lógica de control se implementa en un sistema operativo de red centralizado o NOS (Network Operating System). Con este nuevo enfoque se obtienen grandes beneficios: Aprovisionamiento de políticas simple, agilidad para reconfiguración e implementación de nuevos servicios y aceleración en la innovación de las redes [10].

A. *Arquitectura de Red SDN*

Una red SDN está conformada generalmente por tres capas: Capa de Aplicación, Capa de Plano de Control y Capa de Plano de Datos (ver Figura 1 tomada de [11]).

Capa de Aplicación: Está conformada por las aplicaciones del negocio y por las aplicaciones de servicios de red.

Capa del Plano de Control: Contiene los controladores SDN encargados de gobernar y dirigir la manera en que se transportan los datos en los dispositivos SDN. Se encuentra a cargo de todas las funciones complejas de enrutamiento, manejo de políticas, monitoreo y chequeos de seguridad de la red.

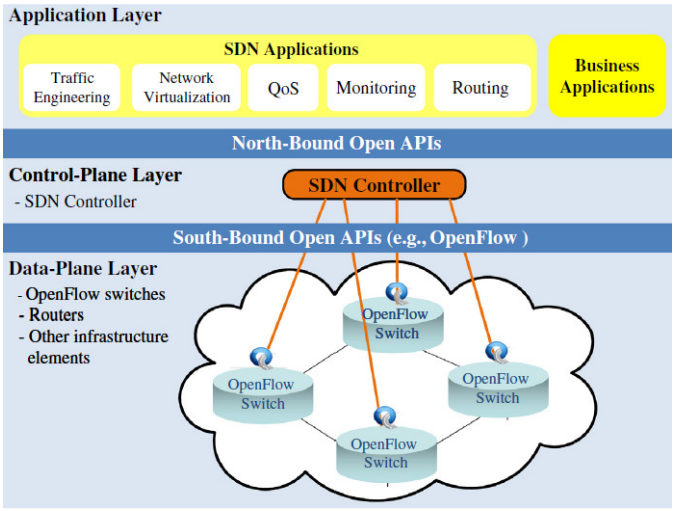

Figura 1: Representación Lógica de una Arquitectura de Red SDN

Capa del Plano de Datos: Está conformada por dispositivos SDN físicos y virtuales encargados de transportar datos en base a instrucciones recibidas por los controladores SDN de la red.

En una red SDN los controladores se comunican con las aplicaciones externas mediante APIs (Application Program Interface) Northbound abiertas y con los dispositivos SDN mediante APIs Southbound abiertas incluyendo el protocolo OpenFlow.

Una red SDN también puede ser vista como una composición de capas y sistemas (ver Figura 2 tomada de [12]). Algunas capas se encuentran presentes en todos los despliegues SDN incluyendo: controladores, infraestructura de red y APIs Southbound, mientras que otras son opcionales incluyendo APIs Northbound, aplicaciones, hipervisores y virtualización de redes.

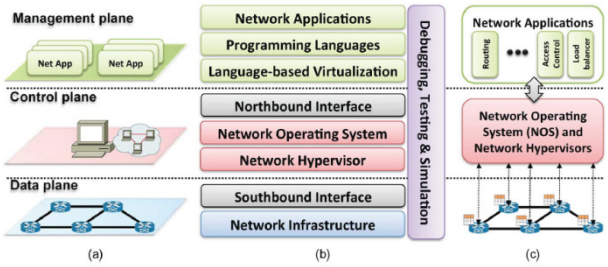

Figura 2: Arquitectura SDN basada en (a) Planos, (b) Capas y (c) Sistemas

Comenzando de abajo hacia arriba, se identifican los siguientes sistemas, planos y capas:

- Infraestructura de Red: Está conformada por los dispositivos SDN físicos o virtuales que manipulan y reenvían paquetes en base a reglas definidas por un controlador. Un dispositivo de red contiene un plano de datos (Forwarding Plane) encargado de transportar y manipular campos de cabecera de paquetes y un plano operacional (Operational Plane) encargado de efectuar tareas administrativas relacionadas con su funcionamiento.

Los dispositivos SDN pueden ser switches, routers o elementos de reenvío físicos o virtuales que soportan planos de datos y Soutbound SDN, como OpenFlow [2].

- APIs Southbounds: Constituyen el API de comunicación que facilita la comunicación entre controladores y dispositivos SDN. Estas APIs pueden ser abiertas o propietarias. El controlador SDN cuenta con las siguientes opciones de APIs Soutbound abiertas: OpenFlow [2], OVSDB [13] (Open vSwitch Database) y ForCES [14] (Forwarding and Control Element Separation). También existen las siguientes opciones de plugins: BGP [15] (Border Gateway Protocol), SNMP [16] (Simple Network Management Protocol) y NETCONF [17] (Network Configuration Protocol) entre otros. OpenFlow es el API Southbound estándar de la industria de las redes para entornos SDN. A través de OpenFlow un controlador puede crear, actualizar, modificar y eliminar entradas en las tablas de flujos de los dispositivos SDN y obtener información de estadísticas de estos.
- Plano de Datos (Forwarding Plane): El plano de datos se encuentra en los dispositivos de red y se encarga de la manipulación y del reenvío de los paquetes e incluye, pero no está limitado a filtros, medidores, marcadores, y clasificadores.
- Hipervisor de Red: En un ambiente de virtualización de servidores, un hipervisor es una plataforma de software que permite correr varias VMs sobre un mismo dispositivo de cómputo. Un hipervisor cuenta con switches virtuales, o vSwitches, que permiten la comunicación entre VMs. En una red SDN basada en vSwitches, un controlador le comunica a los vSwitches las reglas de reenvío de tráfico asociadas a la comunicación entre las VMs. Los vSwitches disponen de mecanismos de túneles para comunicarse con VMs hospedadas en otros hipervisores externos.
- Controlador SDN o NOS (Network Operating System): Es un software que corre en un servidor de red para gestionar las reglas de reenvío de tráfico de los dispositivos SDN. Un controlador SDN posee en su núcleo abstracciones, servicios esenciales y APIs comunes para comunicarse con el resto de los elementos de la arquitectura (ver Figura 3 tomada de [12]). Entre los principales servicios de red ofrecidos se encuentran: (1) gestión de la topología, encargado de descubrir la topología de la red, (2) gestión de estadísticas, encargado de recopilar información de contadores del tráfico de la red, (3) gestión de notificaciones, encargado de gestionar la comunicación del plano de control con los elementos de la red, (4) gestión de dispositivos, encargado de configurar y gestionar los elementos de la infraestructura de red, (5) reenvío de caminos más cortos, encargado de seleccionar los mejores caminos hacia los destinos y (6) mecanismos de seguridad, encargado de proveer mecanismos de protección a la red.
- Plano de Control: Es responsable de gestionar y dirigir el reenvío de tráfico de la red desde un nodo central.
- APIs Northbound: Presentan una interfaz común para el desarrollo de aplicaciones y permiten la interacción entre las aplicaciones externas y el controlador SDN. Actualmente los controladores de red ofrecen una variedad de APIs Northbound, como APIs RESTful [18][19], sistemas de archivos y lenguajes de programación.

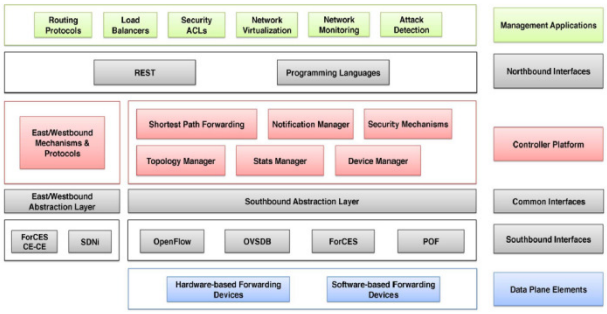

Figura 3: Componentes de un Controlador SDN

- APIs Eastbound/Westbound: Se utilizan en esquemas de controladores SDN distribuidos para permitir la comunicación entre los controladores y compartir información de alcanzabilidad y control. Entre las funciones de estas interfaces se encuentran: (1) importar o exportar datos entre controladores, (2) proveer algoritmos para modelos de consistencia de datos y (3) proveer capacidades de monitoreo y notificaciones. Para que exista compatibilidad e interoperabilidad entre diferentes controladores, se requiere disponer de APIs Eastbound/ Westbound estandarizadas, como SDNi [20], PCEP [21] y BGP [15].
- Virtualización de Red (Slicing): Consiste en la capacidad de compartir el plano de datos del hardware de red a múltiples redes lógicas, cada una con su propio direccionamiento y su propio mecanismo de reenvío. En una red SDN se puede virtualizar la capa de hardware de red en slices y asignar a cada slice recursos y espacios de direcciones. Un slice se define como una instancia de una red virtual, y dos redes virtuales distintas sobre el mismo hardware físico se conocen como slices. La virtualización de una red SDN se puede implementar siguiendo un esquema de virtualización mediante proxys o siguiendo un esquema de virtualización basado en lenguajes. La virtualización de redes SDN mediante proxys se logra con la colocación de un controlador SDN especial entre los controladores y los dispositivos SDN de la red. El proxy actúa como un multiplexor de tráfico y como un gestor de la asignación de recursos a slices independientes. En la actualidad se encuentran los siguientes virtualizadores de red basados en proxies: FlowVisor [22], OpenVirteX [23], y AutoSlice [24]. FlowVisor [22] es un controlador OpenFlow de propósito especial que actúa como un proxy transparente entre switches OpenFlow y controladores OpenFlow ofreciendo una capa de virtualización de red basada en OpenFlow. En FlowVisor los slices pueden ser definidos por una combinación de puertos de switches Capa 1, direcciones Ethernet origen/destino Capa 2, direcciones IP origen/destino de Capa 3 y puertos TCP/UDP origen/destino o códigos/tipos ICMP Capa 4. En la Figura 4 se muestra una topología de red conformada por tres controladores OpenFlow, un controlador FlowVisor y seis switches OpenFlow. FlowVisor recibe todos los comandos OpenFlow de los controladores hacia los switches OpenFlow y las

respuestas y notificaciones de estadísticas de los switches hacia los controladores respectivos garantizando el aislamiento de tráfico entre cada slice definido. En el ejemplo se puede observar como cada controlador tiene una vista particular y diferente de la misma red física. A través de FlowVisor un grupo de investigadores puede crear sus propias instancias de red lógicas corriendo sus propios protocolos de enrutamiento y ejecutarlas sobre una red real en paralelo con una red de producción y mantener el aislamiento y las velocidades de reenvío del hardware existente. La virtualización basada en lenguajes es el uso de lenguajes de propósito especial para virtualización incluyendo Pyretic [30] y Splendid [25].

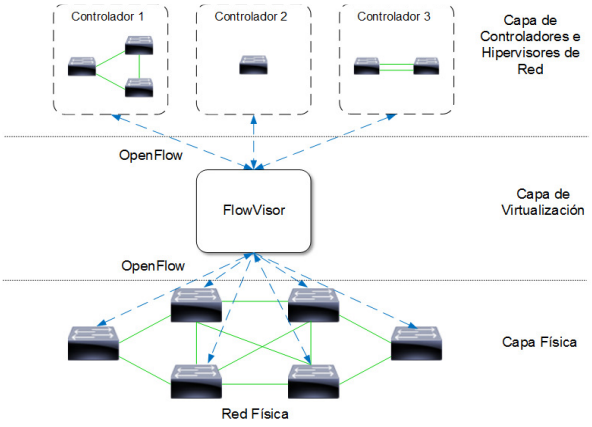

Figura 4: Topología de Red OpenFlow con FlowVisor

- Lenguajes de Programación: Se encuentran los lenguajes que pueden interactuar con el controlador SDN. Entre los más utilizados se encuentran: Python y Java y lenguajes de programación específicos para SDN, como Procera [26], FML [27] (Flow Based Management Language), Frenetic [28], NetCore [29] y Pyretic [30].
- Aplicaciones: Constituyen las aplicaciones del negocio de las organizaciones y aplicaciones externas de propósito especial como balanceadores de carga, sistemas de monitoreo y aplicaciones de QoS.
- Plano de Gestión: El plano de gestión es responsable de las funciones de monitoreo, configuración y mantenimiento de los dispositivos de red. En SDN la gestión de la red se implementa mediante una jerarquía de capas de protocolos, modelos y datos (ver Figura 5 tomada de [32]). Los protocolos de gestión utilizan un lenguaje de modelo de datos como YANG [33] (Yet Another Next Generation) para configurar y obtener datos del estado de los dispositivos. Entre los protocolos de gestión de red SDN se encuentran NETCONF [17] , RESTCONF [34], gRPC [35] y OF-CONFIG [36] (OpenFlow Management and Configuration Protocol).

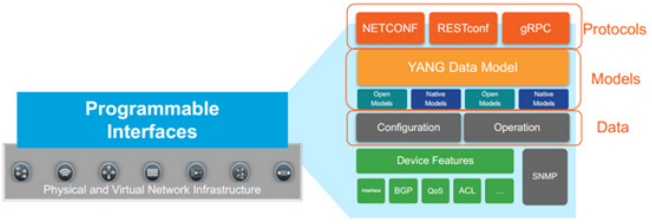

Figura 5: Arquitectura de Gestión basada en Modelos

- NETCONF es un protocolo de configuración de red estandarizado por el IETF en el RFC 6241 [17] que provee mecanismos simples para instalar, manipular, y eliminar la configuración de dispositivos de red. Sigue un paradigma solicitud-respuesta RPC (Remote Procedure Call) con codificación XML y transporte seguro SSH (Secure Shell). NETCONF define almacenes de datos que contienen la configuración de los dispositivos de red y operaciones CRUD (Create, Read, Update, Delete) que permiten recuperar, configurar, copiar y eliminar almacenes de datos. RESTCONF es un protocolo de configuración de red basado en HTTP estandarizado por el IETF en el RFC 8040 [34]. A través de RESTCONF las aplicaciones Web pueden acceder y modificar la configuración de un dispositivo de red siguiendo una arquitectura cliente/servidor basada en RPC con clientes y servidores RESTCONF. gRPC [35] es un framework para llamadas a procedimientos remotos RPC de código abierto y de alto desempeño liberado por la empresa Google para construir sistemas distribuidos masivos. OF-CONFIG [36] es un protocolo de gestión desarrollado por la ONF (Open Networking Foundation) para gestionar switches OpenFlow físicos y virtuales. Utiliza NETCONF para la administración, XML para la codificación y SSH para el transporte.

B. Modelos de Despliegue SDN

En la práctica se utilizan tres modelos de despliegue SDN [4]: (1) Modelo SDN basado en Dispositivos, (2) Modelo SDN Overlay, y (3) Modelo SDN Híbrido. El Modelo SDN basado en Dispositivos [4] se refiere a una red de switches físicos SDN que operan solo bajo las instrucciones de un controlador SDN. Se implementa con rapidez en despliegues nuevos como un nuevo complejo de oficinas dentro de un campus. En la Figura 6 se muestra un despliegue SDN con 6 switches dirigidos por las instrucciones de un controlador SDN. El modelo SDN Overlay [4] está basado en la superposición de redes sobre una infraestructura de red física subyacente. En una red SDN Overlay, los nodos finales SDN son dispositivos virtuales que forman parte de hipervisores en un ambiente de virtualización de servidores.

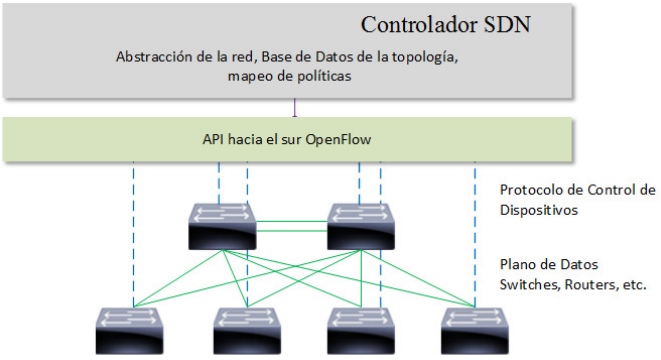

Figura 6: Ejemplo del Modelo SDN basado en Dispositivos

En este escenario el controlador controla el reenvío del tráfico de los switches lógicos que se encuentran definidos en los hipervisores y no altera la red física actual ni el plano de control distribuido de la red subyacente. Para crear la red virtual, los nodos SDN lógicos establecen túneles overlay entre sí a través de un protocolo de túneles: (1) VXLAN [37]

(Virtual Extensible LAN), (2) NVGRE [38] (Network Virtualization using Generic Routing Encapsulation), o (3) STT [39] (Stateless Transport Tunneling). Los túneles overlay usualmente terminan en los switches virtuales dentro de los hipervisores o en dispositivos físicos que actúan como gateways hacia la red existente. En la Figura 7 se muestra un despliegue SDN Overlay conformado por un controlador SDN, tres vSwitches hospedados en tres hipervisores y una red tradicional subyacente de cinco switches.

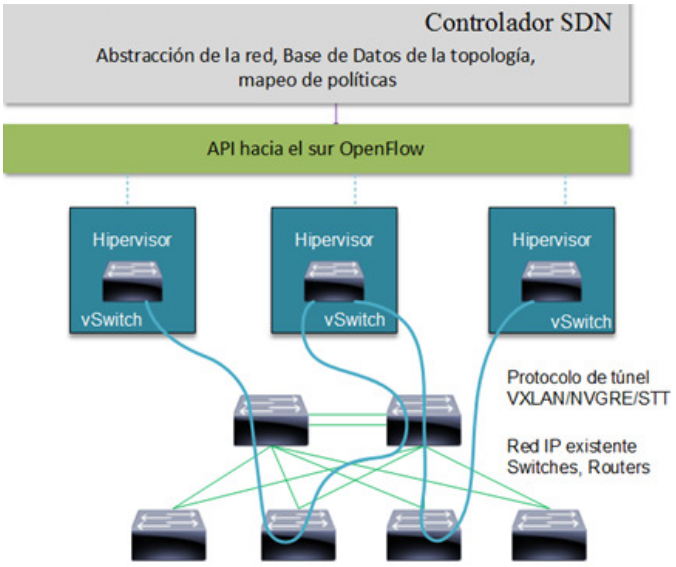

Figura 7: Modelo SDN Overlay

La red virtual consiste de switches lógicos interconectados por enlaces virtuales punto-a-punto. El controlador SDN se apoya en el protocolo hacia el sur OpenFlow para aprovisionar las reglas de reenvío a las tablas de flujos de los vSwitches presentes en los hipervisores de las VMs. El modelo se utiliza en escenarios donde se quiere hacer una implementación rápida de una solución SDN montada sobre una red IP existente. La limitante del modelo es la pérdida de visibilidad del tráfico en la red subyacente que restringe la aplicación de servicios diferenciados e ingeniería de tráfico y la complejidad de gestionar y operar ambas redes.

C. Modelo SDN Híbrido

El modelo SDN Híbrido está basado en la convivencia de tecnologías de redes tradicionales con tecnologías de red SDN en un mismo entorno. En este caso un gateway SDN corre tanto el modelo SDN Overlay, como el modelo SDN Basado en Dispositivos para esquemas híbridos de modelos SDN basado en dispositivos con modelos SDN Overlay. También existe la opción de disponer de un gateway SDN que corre protocolos legados tradicionales y protocolos SDN/OpenFlow. El gateway en un esquema híbrido se comunica con el controlador SDN OpenFlow y con los dispositivos de red Ethernet tradicionales y corre bajo ambos esquemas de red y protocolos.

IV. OpenFlow

OpenFlow [40] es la primera interfaz estandarizada diseñada específicamente para SDN, brindando alto desempeño, y control de tráfico granular en dispositivos de múltiples fabricantes. OpenFlow es un protocolo SDN abierto que se utiliza para controlar el plano de datos de los dispositivos SDN desde un nodo central. Constituye la primera interfaz de comunicación estándar definida entre los planos de datos y control de una arquitectura SDN y permite el acceso directo y la manipulación del plano de datos de routers y switches, tanto físicos, como virtuales. El protocolo se implementa en ambos lados de la interfaz entre los dispositivos de la infraestructura de red y el software de control SDN. Utiliza el concepto de flujos para identificar el tráfico de la red basado en reglas estáticas predefinidas o en reglas programadas dinámicamente por el software de control SDN.

A. Arquitectura OpenFlow

La arquitectura de red OpenFlow consiste de tres conceptos básicos: (1) la red está soportada por switches con capacidad OpenFlow que conforman el plano de datos, (2) el plano de control consiste de uno o más controladores OpenFlow que definen y publican las reglas de reenvío al plano de datos de los switches OpenFlow y (3) el controlador y los switches OpenFlow se comunican entre sí mediante un canal de control seguro (ver Figura 8 tomada de [40]). En una arquitectura OpenFlow, el reenvío de los datos se efectúa en los switches de la red, y las decisiones de reenvío se hacen en un programa de software de un controlador externo implementado en un servidor que se comunica con los switches a través del protocolo OpenFlow. A través del protocolo OpenFlow, un controlador puede añadir, actualizar, y eliminar entradas de flujos en las tablas de flujos, de manera proactiva o de manera reactiva en respuesta a la llegada de paquetes, modificando el comportamiento de reenvío del plano de datos de los switches.

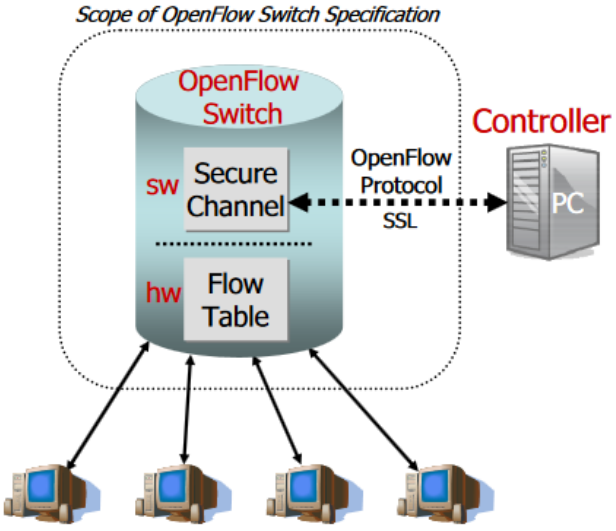

Figura 8: Arquitectura de Red OpenFlow

Dentro de un dispositivo OpenFlow, cuando un paquete ingresa a alguno de sus puertos se inicia un proceso de consulta en la tabla de flujos para encontrar una entrada de flujos que concuerde con el paquete. Las entradas de flujos son evaluadas en orden de prioridad y se utiliza la primera concordancia en la tabla. Si existe una concordancia, se ejecutan las acciones indicadas en la entrada de flujo y se actualizan los contadores de paquetes para la entrada de flujos correspondiente. En caso de no existir concordancia en alguna entrada de la tabla de flujos, el paquete se envía al controlador sobre el canal OpenFlow o se descarta el paquete.

Entre las instrucciones asociadas con cada entrada de flujo se encuentran: (1) reenviar el paquete a un puerto, (2) modificar los campos de cabecera del paquete y (3) descartar el paquete. Las entradas de flujos pudieran indicar enviar un paquete a un puerto físico, o a un puerto virtual definido por el switch, o a un puerto virtual reservado por la especificación OpenFlow. Los puertos virtuales reservados pudieran especificar acciones de reenvío genéricas como reenvío a un controlador, hacer una inundación de puertos, o reenviar los paquetes usando métodos no-OpenFlow, como el procesamiento de un switch Ethernet tradicional. Los puertos virtuales definidos por el switch permiten especificar grupos de agregación de enlaces, túneles o interfaces de loopback.

B. *Ejemplo de Tabla de Flujos OpenFlow*

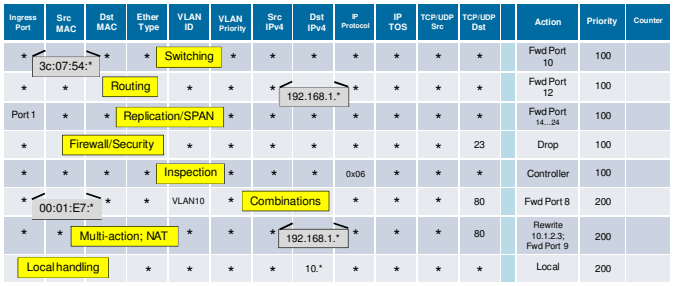

Figura 9: Ejemplo de Tabla de Flujos OpenFlow

En la Figura 9 tomada de [41] se presenta un ejemplo de una tabla de flujos de un switch OpenFlow v1.0.

En este ejemplo, se presentan las siguientes reglas para todos los paquetes que ingresan al switch:

- Switching: Todos los paquetes cuya dirección MAC origen sea 3c:07:54:* son enviados al puerto 10.
- Routing: Todos los paquetes cuya dirección IP destino sea 192.168.1.* son enviados al puerto 12.
- Replicación/SPAN: Todos los paquetes que ingresen por el puerto 1 son replicados en los puertos que van desde el puerto 14 hasta el puerto 24.
- Firewall/Seguridad: Todos los paquetes con número de puerto TCP/UDP destino 23 son descartados.
- Inspección: Todos los paquetes con protocolo IP 0x06, que corresponden a segmentos TCP, son reenviados al controlador para análisis ulterior.
- Combinaciones: Todos los paquetes cuya dirección MAC origen sea 00:01:E7:* y que pertenezcan a la VLAN 10 y que tengan puerto TCP/UDP destino 80 son enviados al puerto 8.
- Multi-acción/NAT: Todos los paquetes cuya dirección IP destino sea 192.168.1.* y con puerto TCP/UDP destino 80 tendrán nueva dirección IP origen 10.1.2.3 y son enviados al puerto 9.
- Manejo local: Todos los paquetes cuya dirección IP destino sea 10.*.*.* son manejados localmente.

C. *Tipos de Switches OpenFlow*

Existen dos categorías de switches OpenFlow: switches Solo-OpenFlow (OpenFlow-only), cuyo procesamiento de paquetes está basado únicamente en el protocolo OpenFlow y switches OpenFlow-híbrido (OpenFlow-hybrid), que soportan tanto la operación OpenFlow como la operación de switching Ethernet tradicional.

D. *Protocolo OpenFlow*

El protocolo OpenFlow describe el intercambio de mensajes que tiene lugar entre un controlador OpenFlow y un switch OpenFlow. Generalmente el protocolo es implementado sobre SSL (Secure Socket Layer) o TLS (Transport Layer Security), para proveer un canal OpenFlow seguro o sobre un canal TCP. Existen tres tipos de mensajes entre el controlador y los switches OpenFlow: (1) Controller to Switch, son mensajes enviados por el controlador a un switch para solicitar notificaciones de capacidades, estados de los puertos y estadísticas de paquetes, o para modificar el estado del reenvío del switch, como añadir, eliminar o modificar entradas en las tablas de flujos, (2) Asynchronous, son mensajes enviados de manera asíncrona por los switches al controlador para denotar la llegada de un nuevo paquete, notificar el cambio de estado en el switch, o informar la ocurrencia de algún error y (3) Symmetric, son mensajes enviados en cualquier dirección sin una solicitud expresa, como los mensajes Hello que notifican la disponibilidad de un dispositivo y los mensajes Echo Request y Echo Reply que pueden ser utilizados para medir la latencia en el canal de comunicación entre el controlador y el switch.

E. *Especificaciones OpenFlow*

OpenFlow ha evolucionado desde su aparición a finales de 2007 hasta la actualidad. En la siguiente sección se describen las características resaltantes de cada versión mayor:

OpenFlow 1.0.0 [42]: Especificación donde cada switch tiene una sola tabla de flujos. La tabla de flujos consta de tres grupos de campos: Header Fields, Counters y Actions. El grupo Header Fields contiene 12 campos: Ingress Port o puerto de ingreso del paquete, Ethernet Source Address o dirección Ethernet origen del paquete, Ethernet Destination Address o dirección Ethernet destino del paquete, Ethernet Type que indica cual es el protocolo encapsulado en el payload del frame Ethernet, VLAN ID para la VLAN del paquete, VLAN priority CoS que es un campo de prioridad de 3 bits, IPv4 Source Address o dirección IPv4 origen del paquete, IPv4 Destination Address o dirección IPv4 destino del paquete, IPv4 Protocol Number o número de protocolo IPv4, IP ToS o campo Type of Service de IPv4, TCP/UDP Source o número de puerto TCP/UDP origen o tipo ICMP (Internet Control Message Protocol), TCP/UDP Destination o número de puerto TCP/UDP destino o tipo ICMP.

OpenFlow 1.1.0 [43]: Modifica la nomenclatura de los campos a Match Fields, Counters e Instrucctions. Amplía los campos de cabecera incluyendo etiquetas para el soporte de MPLS. Incorpora múltiples tablas de flujos e implementa un mecanismo pipeline OpenFlow, en el cual un paquete atraviesa varias tablas de flujos en serie y recibe un procesamiento particular en cada tabla. Incluye una tabla de grupos para brindar funcionalidad común a un conjunto de paquetes. Permite disminuir el campo TTL (Time-To-Live) en la cabecera IP (ver Figura 10).

OpenFlow 1.2.0 [44]: Incluye campos para el soporte de IPv6. Soporta el análisis de campos de paquetes adicionales a través de una estructura TLV (Type-Length-Value), referenciada como OXM (OpenFlow Extensible Match). Soporta canales redundantes desde un switch a múltiples controladores.

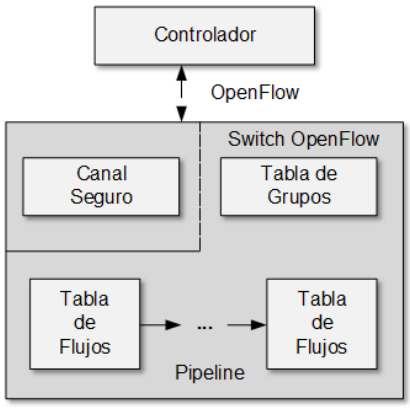

Figura 10: Componentes de un Switch OpenFlow 1.1.0

OpenFlow 1.3.0 [45]: Introduce nuevas funcionalidades para el soporte de funciones administrativas OAM (Operations Administration and Management) e incorpora la tabla especial de medidores al plano de datos del switch que recopilan estadísticas particulares como la tasa de bytes o paquetes de un flujo particular. Otra mejora importante es el soporte extendido de múltiples controladores auxiliares, permitiendo implementar mecanismos de balanceo de cargas hacia los controladores.

OpenFlow 1.4.0 [46]: Agrega estructuras TLV para puertos, tablas y colas. Incluye el soporte de puertos ópticos. Adicionalmente, los controladores soportan el envío de mensajes en lotes.

OpenFlow 1.5.0 [47]: Introduce el mecanismo de Tablas de Egreso permitiendo que el procesamiento se realice en el contexto de puertos de salida.

V. CONTROLADORES SDN

En esta sección se analizan 3 controladores de código abierto disponibles en la industria: OpenDaylight [48], Floodligth [49] y Ryu [50]. Los criterios de selección de los controladores utilizados fueron: (1) tiempo en el mercado mayor a 3 años, (2) foco activo en su desarrollo, (3) soporte del API OpenFlow 1.3 o superior y (4) disponibilidad de documentación y soporte en línea con FAQs (Frequently Asked Questions).

A. OpenDaylight

La Tabla I resume las características principales del controlador OpenDaylight.

Tabla I: Caracteristicas de OpenDaylight

Concepto
Controlador de código abierto que sigue una arquitectura modular e incluye su propia máquina virtual Java, lo cual le permite ser desplegado en cualquier plataforma de hardware y software con soporte Java.
Software
Última versión y fecha de liberación: Sexta versión denominada Carbon, mayo, 2017. Lenguaje de Núcleo: Está basado en un framework modular basado en Java 1.7 y soporta el framework de programación OSGI (Open Specifications Group Initiative). APIs Northbound: Soporta REST bidireccional. Servicios y Orquestación de Aplicaciones de Red: Incluye aplicaciones del negocio y aplicaciones de control, aprovisionamiento y gestión de la red, aplicaciones de nube, aplicaciones de centros de datos, funciones de red y servicios de virtualización. Plataforma de Controladores: Incluye los siguientes módulos: gestión de topología (Topology Manager), gestión de estadísticas (Statistics Manager), gestión de switches (Switch Manager), gestión de reglas de reenvío (Forwarding Rules Manager) y rastreador de hosts (Host Tracker). Plugins de protocolos y APIs Southbound: OpenFlow 1.0, OpenFlow 1.3, OF-CONFIG, NETCONF, LISP, OVSDB, BGP, CAPWAP y otros. Soporte de Comunicación C2C (Controlador-a-Controlador): Incluye la aplicación ODL-SDNi para proveer el establecimiento de la interfaz Este-Oeste (SDNi communication) entre múltiples controladores OpenDaylight.
Usabilidad
Documentación de instalación y documentación operativa disponible en la wiki del proyecto.
Implementación
El controlador OpenDaylight corre sobre una máquina virtual Java. Para mejores resultados, se recomienda una distribución Linux y Java 1.7.
Funcionalidades
Soporta OpenFlow 1.0 y 1.3. Soporta el framework OSGi y Java como lenguaje de desarrollo y soporta REST bidireccional empleando JSON y API DOM empleando XML. Incluye aplicaciones: OpenDOVE y VTN para implementar virtualización de red overlay. Integración con OpenStack a través de un API Neutron.
Licenciamiento
Licencia: Eclipse Public License (EPL-1.0).
Impacto Potencial
Popularidad: Alta. Tiene fuerte soporte de la industria y de la comunidad de código abierto.

B. Floodlight

La Tabla II resume las características principales del controlador Floodlight.

Tabla II: Caracteristicas de Floodlight

Concepto
Controlador OpenFlow extensible basado en Java con licencia Apache desarrollado por una comunidad abierta de programadores y patrocinado por Big Switch Networks y un conjunto de aplicaciones construidas en el tope del controlador.
Software
Última versión y fecha de liberación: Floodlight v1.2, febrero. 2016. Lenguaje de Núcleo: Java 1.7. APIs Northbound: REST con formato de datos JSON. Aplicaciones: Circuit Pusher, utiliza APIs REST Floodlight para crear un circuito bidireccional entre 2 puntos terminales IP. OpenStack Quantum Plugin, permite que el controlador Floodlight corra como una red de backend para el sistema operativo de nubes OpenStack utilizando el plugin Neutron. Virtual Network Filter, módulo para virtualizar redes basadas en direcciones MAC capa 2. Learning Switch, switch de aprendizaje capa 2 común. Firewall, implementa reglas ACL a switches OpenFlow. Hub, aplicación que siempre inunda un paquete de ingreso a todos los puertos activos de un switch. Load Balancer, balancea cargas de flujos TCP, UDP y ping. Servicios/Funciones del Core: Inventario de dispositivos (hosts). Administrador de la topología. Módulo para insertar flujos y grupos en la red OpenFlow (Static Entry Pusher). Monitoreo del desempeño del controlador. Capacidad de reenvío Dijkstra que permite interconectar islas de switches OpenFlow con switches no-OpenFlow. APIs Southbound: Solamente OpenFlow desde la v1.0 hasta la v1.4. Soporte de Comunicación C2C: Floodlight no provee soporte para ningún tipo de comunicación C2C, por lo cual no soporta alta disponibilidad, ni interfaces este-oeste, ni controladores jerárquicos.
Usabilidad
Documentación de la instalación y documentación operativa disponible.
Implementación
Se recomienda una distribución Linux Ubuntu 14.0.4 TLS Trusty Tahr o Ubuntu 16.04.1 LTS Xenial Xerus. Para instalar Floodlight en Linux se requiere un cliente Git, Python y la herramienta Apache Ant.
Funcionalidades
Soporta OpenFlow v1.0 a v1.4. Se integra con sistemas de gestión de nube a través del módulo Virtual Network Filter (VNF), el cual provee un API REST para integrarse con Quantum/OpenStack. Se integra con aplicaciones externas a través de APIs REST.
Licenciamiento
Licencia: Apache 2.0.
Impacto Potencial
Uso Comercial: Representa el núcleo del controlador comercial Big Switch Network Controller.

C. Ryu

La Tabla III resume las características principales del controlador Ryu.

Tabla III: Caracteristicas de Ryu

Concepto
Ryu es un framework SDN basado en componentes con APIs bien definidas que facilitan la creación de aplicaciones de control y gestión de redes.
Software
Última versión y fecha de liberación: Ryu v4.8, noviembre. 2016. Lenguaje de Núcleo: Python 2.6+, greenlets, pseudo-multihilo cooperativo (utilizando un sólo núcleo). Interfaces/Lenguaje de Aplicaciones: Python 2.6+ para las aplicaciones del controlador y REST para las aplicaciones externas. Arquitectura de Software: El framework SDN Ryu está conformado por un conjunto de componentes, librerías e interfaces para el intercambio de información. La capa de aplicaciones contempla las siguientes aplicaciones de red: switch capa 2, firewall, IDS (Snort), abstracciones de túneles GRE, VRRP (Virtual Router Redundancy Protocol) y los servicios de descubrimiento de la topología y manejo de estadísticas entre otros. La capa de control incluye el soporte de protocolos hacia el sur y los siguientes componentes principales: manejador de eventos, analizador de mensajes OpenFlow, gestor de memoria, gestor de aplicaciones, servicios de infraestructura y un conjunto de librerías que incluyen NETCONF, sFlow y NetFlow. Soporte de Comunicación C2C: Es capaz de soportar servicios de alta disponibilidad a través del componente Zookeeper.
Usabilidad
Documentación de la instalación y documentación operativa disponible.
Implementación
Sistema Operativo Soportado: Distribución GNU/Linux reciente. Dependencias en Tiempo de Ejecución: Paquetes Python (python-eventlet, python-routes, python-webob y python-paramiko).
Funcionalidades
Versión OpenFlow Soportada: 1.0, 1.2, 1.3, 1.4, 1.5 y extensiones Nicira. APIs Southbound: NETCONF, OF-CONFIG, SNMP, OVSDB y OpenFlow. Integración con Sistemas de Gestión de Nube: Incluye el plugin OpenStack. Utiliza APIs REST. Servicios/Funciones del Core: Incluye analizadores y generadores de paquetes de diferentes protocolos de red. Incluye un plugin para integrarse al controlador de redes de nube OpenStack Neutron que soporta túneles overlay basados en GRE y configuraciones de VLANs.
Licenciamiento
Licencia: Apache 2.0.
Impacto Potencial
Popularidad: Es utilizado principalmente en campos de investigación debido a que es fácil de aprender y fácil de expandir. Uso comercial: Ha tenido poco uso comercial debido en parte a que varias evaluaciones de desempeño [51][52] han reportado lentitud del controlador en comparación con otros controladores.

La selección de un controlador para un despliegue SDN depende de factores como el tamaño de la red, el costo de adquisición, el desempeño de la red requerido por las aplicaciones, la confiabilidad, la facilidad de gestión, el nivel de escalabilidad y las APIs soportadas. Para ambientes de escala grande y mediana se puede optar por un conjunto de controladores OpenDaylight en alta disponibilidad en base a su robustez, escalabilidad y amplio soporte. Para ambientes de escala pequeña se puede considerar una implementación basada en un controlador Floodlight y para entornos de experimentación y desarrollo de nuevas tecnologías se pueden utilizar controladores Ryu.

En la industria existen alternativas de controladores SDN/OpenFlow comerciales incluyendo: (1) Big Network Controller [53], basado en el controlador Floodlight, (2) Brocade SDN Controller [54], distribución comercial de OpenDaylight por Brocade, (3) Cisco Open SDN Controller [55], distribución comercial de OpenDaylight por Cisco Systems, (4) Aruba VAN SDN Controller [56] y (5) NEC ProgrammableFlow PF6800 [57]. La selección del controlador comercial dependerá de factores como el costo de capital, el servicio técnico, la documentación, el apoyo durante la migración o implementación de la solución, la facilidad de la gestión, los APIs soportados para interactuar con el controlador y las aplicaciones embebidas en las propuestas de cada fabricante.

VI. Fases para el Despliegue de SDN

En esta sección se describen las fases para desplegar redes SDN basado en las mejores prácticas y recomendaciones recopiladas de los casos de estudio de SDN descritos por el Grupo de Trabajo para Migración hacia SDN de la ONF, específicamente considerando las pautas relacionadas con la migración hacia SDN en el campus de la Universidad de Stanford [2][3].

La Universidad de Stanford migró dos edificios de su campus a SDN en el año 2010 [2]. La meta fue promulgar la tecnología SDN para facilitar la experimentación en redes de producción, comprender y verificar la tecnología SDN y contribuir con el desarrollo del protocolo OpenFlow. La implementación estuvo basada en un despliegue SDN Híbrido con switches OpenFlow híbridos de múltiples fabricantes con asignación de una VLAN OpenFlow y una VLAN legada. También se consideraron Access Points con switches de software de referencia OpenFlow basados en Linux. El plano de control se hizo con controladores abiertos disponibles para la fecha y se virtualizó la red con FlowVisor en slices para tráfico de datos de experimentación y slices para tráfico de datos de producción. Para garantizar el éxito de la migración se implementó una infraestructura de monitoreo que recopiló estadísticas del plano de control antes, durante y después de la migración.

En general un despliegue SDN consta de tres fases importantes: (1) Preparación, (2) Implementación y (3) Validación.

A. Preparación

La fase de preparación contempla (1) el análisis de los requerimientos del negocio, (2) el diseño de alto nivel de la solución, (3) la preparación de las pruebas de concepto, (4) el diseño de bajo nivel de la solución y (4) la planificación de la implementación.

Análisis de Requerimientos: Esta etapa consiste en la identificación de las metas y objetivos del negocio que se deben alcanzar con la nueva red y en el reconocimiento del estado actual de la red. Particularmente se debe recopilar y analizar la siguiente información: (1) metas del negocio, (2) caracterización de las aplicaciones del negocio y los servicios de red incluyendo tipo de aplicación, volumen, prioridades, requerimientos de ancho de banda y reglas de acceso, (3) topología de red incluyendo usuarios, sistemas, servidores, equipos de red, tipo y velocidad de enlaces de conexión, (4) identificación de segmentos de red incluyendo definiciones de VLANs, direccionamiento IP y protocolos de red, (5) identificación de políticas del negocio, políticas de seguridad, políticas de calidad de servicio de la red y normas de cumplimiento como regulaciones y (6) identificación de los sistemas de gestión de la red.

La evaluación de la red se puede hacer levantando la información con informes del cliente y/o corriendo herramientas de monitoreo y análisis de tráfico que permitan identificar métricas de utilización de la red identificando las aplicaciones, usuarios y servicios con mayor uso y volumen de solicitudes. Esta etapa también contempla la identificación de riesgos, cuellos de botella de la red o problemas de desempeño.

Diseño de Alto Nivel: Muestra un diagrama de alto nivel de la solución identificando los componentes de la arquitectura de red SDN de la solución incluyendo controladores, dispositivos de red, aplicaciones y sistemas de gestión. La selección de los controladores y los dispositivos de red se realizan en base a los resultados del análisis de los requerimientos y considera aspectos como costos, facilidad de uso, desempeño, soporte y APIs soportadas. Se debe seleccionar el tipo de switch OpenFlow solo-OpenFlow/Híbrido o virtual, dependiendo del modelo de despliegue de la solución seleccionado.

Pruebas de Concepto: Consiste en la preparación de un banco de pruebas para determinar si la solución a implementar satisface las metas del negocio. La implementación del banco de pruebas se puede realizar mediante la utilización de herramientas de simulación SDN de código libre de prototipado rápido incluyendo las herramientas Mininet [58], VT-Mininet [59] y OFNet [60] o herramientas comerciales incluyendo EstiNet [61]. La herramienta OFNet permite hacer emulaciones de red gráficas y provee capacidades de depuración visual, monitoreo de desempeño y capacidades de generación de tráfico sintético que facilitan la depuración de la solución y la optimización del diseño de la solución.

Diseño de Bajo Nivel: Consiste en la definición de las reglas de control de tráfico que serán implementadas en el controlador SDN/OpenFlow en base a los requerimientos y las políticas del negocio.

Planificación: Se establece la línea de tiempo de las fases de ejecución del despliegue SDN y los entregables e informes de cada fase.

B. Implementación

La fase de implementación consiste en la migración de la red legada, los usuarios y servicios a la red SDN/OpenFlow de la solución. El esquema de migración dependerá del análisis de requerimientos donde se determinará el modelo de despliegue SDN. En el caso de una red nueva totalmente OpenFlow, se implementa el modelo SDN basado en dispositivos. En el caso de escenarios de migración de red legada a SDN/OpenFlow, se implementa un modelo SDN Híbrido.

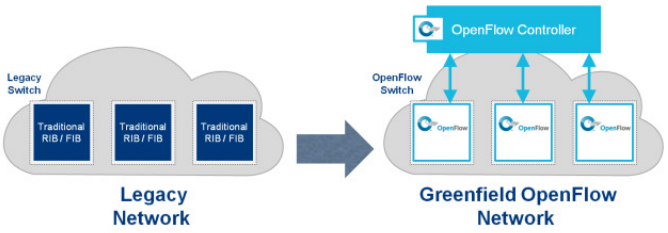

Figura 11: Despliegue Greenfield

Despliegue Greenfield: La ONF denota al modelo SDN basado en Dispositivos con el nombre de despliegue Greenfield (ver Figura 11). En este caso la red existente es actualizada para convertirse en OpenFlow y el control de los dispositivos de la red es reemplazado con un esquema de control de tráfico a través del controlador SDN. Los dispositivos SDN pueden ser dispositivos que aceptan un plugin a los cuales se le instala un agente OpenFlow o switches OpenFlow nativos. El despliegue de esta red consiste en la conexión de red entre el controlador y los dispositivos de red. A partir de este momento el controlador se encargará de hacer un descubrimiento de la red y publicará las reglas de las tablas de flujos a los dispositivos de red. Posteriormente se definen las reglas de filtrado y conectividad específicas en base a las políticas de seguridad, calidad de servicio y políticas de red recopiladas durante la fase de preparación.

Despliegue Mixto (Mixed): La ONF denota el modelo SDN Híbrido con el nombre Despliegue Mixto. Este enfoque considera que se desplegarán nuevos dispositivos OpenFlow los cuales coexistirán con los dispositivos legados que operan bajo protocolos tradicionales distribuidos. En este caso el controlador SDN/OpenFlow y los dispositivos tradicionales intercambiaran información de enrutamiento entre sí a través de protocolos de red tradicionales.

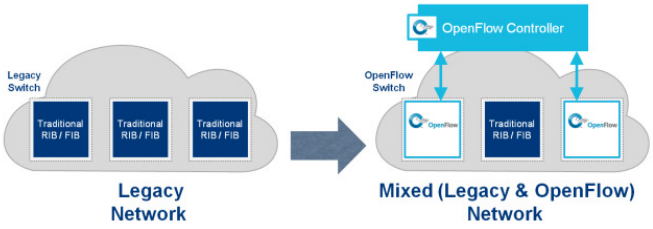

Figura 12: Despliegue Mixto

Enfoque de Migración: El enfoque de migración recomendado por la ONF para este escenario consiste en mover usuarios individuales y VLANs individuales a la red de control OpenFlow para gestionar el riesgo asociado en desplegar la nueva tecnología. Se consideran 4 fases fundamentales:

- Añadir soporte OpenFlow al hardware: Se hace una actualización del firmware a los dispositivos de red para soportar OpenFlow.
- Verificar el soporte OpenFlow en los dispositivos de red: Se verifica la funcionalidad OpenFlow incorporando una VLAN experimental gestionada por el controlador SDN.
- Migración de usuarios a la nueva red: Se crea una nueva red no-OpenFlow y se migran los usuarios a esta nueva red antes de utilizar OpenFlow para el tráfico de producción. Esta tarea involucra las siguientes actividades:
  - Añadir una nueva subred de producción.
  - Añadir/Mover usuarios gradualmente a la nueva subred.
  - Verificar la alcanzabilidad dentro de la nueva subred.
- Habilitar OpenFlow en la nueva subred: Una vez verificado que la nueva subred es funcional, se habilita el control OpenFlow a la subred configurando la regla correspondiente en el controlador.

C. Validación

La validación es una fase fundamental en el proceso de despliegue de la red ya que permite verificar el cumplimiento de los objetivos del diseño y de las metas del negocio. Para

efectuar la validación, el despliegue debe contar con una infraestructura de monitoreo que permita recopilar, analizar y verificar el correcto funcionamiento y la alcanzabilidad, desempeño y estabilidad de la nueva red.

Infraestructura de Monitoreo: La infraestructura de monitoreo está conformada por un conjunto de herramientas de captura, almacenamiento y análisis de tráfico y por encuestas de satisfacción al cliente. La infraestructura de monitoreo recopila información en 2 planos:

- Plano de Control: Se recopila información a nivel de flujos basado en mensajes de control entrantes *packet-in* y *flow-exp*. Otras estadísticas importantes a recopilar son la tasa de arribo de flujos *flow_arrival_rate* y los flujos activos *active_flows*.
- Plano de Datos: Se pueden implementar nodos de monitoreo dedicados a la recolección de estadísticas ejecutando comandos ping y wget entre sí que permitan recopilar información sobre la utilización de CPU del switch, el tiempo de establecimiento de flujos, RTT, retardos wget y tasa de pérdidas de paquetes.

Entre las herramientas de monitoreo convencionales se encuentran ping [62], tcpdump[1] y wget[2]. También es conveniente disponer de herramientas más completas con interfaces gráficas y soporte de reportes como NetFlow [63], sFlow [64], JFlow [65] o herramientas de monitoreo especializadas que verifiquen el comportamiento de la red y la correctitud del funcionamiento OpenFlow. Algunas herramientas especiales disponibles son:

PayLess [66]: Es un framework de monitoreo basado en consultas para redes SDN que provee un API RESTful flexible para recopilar estadísticas a diferentes niveles de agregación, como flujos, paquetes y puertos. PayLess ejecuta la recopilación de información con alta precisión en tiempo real sin incurrir en un alto overhead de red.

OpenTM [67]: Es una arquitectura de monitoreo que lleva la traza de los flujos activos en una red OpenFlow. Adicionalmente obtiene la información de enrutamiento de la aplicación de enrutamiento del controlador y sondea periódicamente los contadores de cantidad de bytes y número de paquetes de los flujos activos en los switches a lo largo del camino de datos. Para reducir la sobrecarga en la red se pueden sondear de maneara aleatoria un subconjunto de los switches seleccionados cuidadosamente para no afectar la precisión de las estadísticas recopiladas por la herramienta.

FlowSense [68]: Es una arquitectura de monitoreo que permite estimar el desempeño de una red OpenFlow a un bajo costo. Utiliza un método pasivo que captura y analiza el intercambio de los mensajes de control entre los switches y el controlador de una red OpenFlow asociados a cambios en el tráfico de la red, como ocurre con los mensajes *PacketIn* y *FlowRemoved*. FlowSense utiliza los mensajes *PacketIn* que notifican la llegada de un nuevo flujo y los mensajes *FlowRemoved* que notifican la expiración de un flujo, para estimar la utilización de un enlace por un flujo.

[1] http://www.tcpdump.org
[2] https://www.gnu.org/software/wget

VII. Lineamientos Generales

El éxito de una implementación SDN se encuentra alineado a los siguientes lineamientos generales:

A. Factibilidad Técnica y Comercial

La red SDN/OpenFlow de la solución debe satisfacer los requerimientos del negocio y el siguiente conjunto mínimo de objetivos generales para proveer factibilidad técnica:

- La disponibilidad de la red debe ser superior a un objetivo definido. Por ejemplo, en la red de la Universidad de Stanford, la disponibilidad esperada era superior a 99.9%.
- Debe existir un mecanismo de reversión a la red legada en caso de fallas durante la migración.
- El desempeño de la red debe estar cercano o ser superior al desempeño de la red legada.
- La experiencia de los usuarios no debe ser afectada durante la migración. La red objetivo debe ser programable mediante APIs abiertos y extensibles.
- La red objetivo debe ser utilizable y soportar actualizaciones automatizadas de software con mínima interrupción de servicio y debe tener mecanismos de reversión.
- La red objetivo debe ser interoperable con dispositivos de red de diferentes fabricantes.
- La red objetivo debe ser administrable con software, herramientas y simuladores disponibles.
- La red de arranque puede requerir la preparación y transformación a un estado intermedio seguro desde el cual se pueda proceder a una migración total de la solución.
- La red final debe ser validada con respecto a los requerimientos y expectativas del negocio.

Los costos de la solución deben respetar el presupuesto asignado al proyecto para satisfacer la factibilidad comercial.

B. Modelo de Despliegue

La selección del modelo de despliegue es un factor determinante en los costos y la gestión de riesgo de la solución. El modelo SDN basado en Dispositivos debe implementarse en despliegues Greenfield, mientras que los modelos SDN Overlay o SDN Híbridos deben implementarse en despliegues Mixtos.

C. Selección de los Componentes de Hardware y Software

La selección de los componentes de la solución dependerá de los costos y los requerimientos de criticidad de la solución. El tipo y modelo de controladores se determina en base a las cargas de tráfico actual, al crecimiento esperado de la red y a las funcionalidades soportadas. Para la selección de los controladores se toman en cuenta los siguientes criterios: (1) desempeño, (2) escalabilidad, (3) alta disponibilidad, (4) modularidad, (5) flexibilidad, (6) interoperabilidad, (7) soporte de delegación del control, (8) portabilidad de aplicaciones y (9) licencias.

D. Fases

Para el despliegue de una red SDN en un campus empresarial, se deben cumplir una serie de fases y un conjunto de consideraciones y lineamientos técnicos y estratégicos que

garanticen el éxito de la implementación. En un despliegue SDN se deben cumplir las siguientes fases de ejecución: (1) Preparación, (2) Implementación y (3) Validación.

E. Validación

El éxito de un despliegue SDN depende del cumplimiento de la solución con las expectativas del negocio. Se deben implementar pruebas de validación en los planos de control y datos durante la migración para verificar que se cumplen los objetivos del negocio.

F. Disponibilidad

En una red SDN/OpenFlow, el controlador es el único responsable del funcionamiento de la red lo cual ocasiona la condición de un solo punto de falla, en la cual al fallar el controlador o al fallar su comunicación ocasiona un fallo general en la red. Para evitar esta condición se pueden implementar los siguientes mecanismos:

Redundancia de controladores: La plataforma de controladores debe ser redundante para continuar el funcionamiento de la configuración de políticas en caso de que falle el controlador principal a cargo de la red.

Datos de controladores en espejo: La base de datos utilizada por el controlador principal para efectuar la toma de decisiones debe ser replicada y estar disponible para su uso por otros controladores de la red, de tal forma, que los controladores puedan configurar los dispositivos de red de una manera consistente.

Redundancia de caminos hacia los controladores: Se deben establecer enlaces y caminos redundantes hacia los controladores para asegurar que exista al menos un camino de comunicación disponible entre un switch y un controlador.

G. Escalabilidad

La solución debe ser capaz de escalar con el crecimiento de usuarios, aplicaciones y dispositivos en la red. En caso de superarse la capacidad, se debe contar con un plan de acción para soportar las nuevas cargas de tráfico, incluyendo mecanismos de redundancia o segmentación de la red en dominios de control que compartan alcanzabilidad entre sus controladores.

H. Gestión

Para garantizar el éxito de la implementación de la solución se debe contar con una infraestructura de monitoreo y gestión robusta capaz de dar seguimiento y control a los aspectos de alcanzabilidad, desempeño, estabilidad y correctitud del funcionamiento SDN/OpenFlow.

I. Seguridad

Se deben proveer mecanismos de seguridad al tráfico de la red a través de la definición de reglas de filtrado en los controladores y en el desarrollo de programas de control que interactúen con el controlador y permitan al controlador manipular los caminos de flujos basado en el análisis de tráfico y en las estadísticas de la red.

J. Transferencia de Conocimientos

Para garantizar la operación de la nueva red es fundamental impartir el conocimiento del funcionamiento y operación de la red OpenFlow a los administradores de la red. Se debe hacer énfasis en los aspectos de configuración, actualización y validación de reglas en el controlador SDN, en la integración con aplicaciones externas, en los mecanismos de redundancia de los controladores y en los aspectos de programación de los controladores de la red.

VIII. CONCLUSIONES Y TRABAJOS FUTUROS

SDN es una arquitectura de red emergente que desacopla el plano de control y el plano de datos de los dispositivos y centraliza la inteligencia y el control de la red en un controlador central. La centralización de la inteligencia en una entidad central provee grandes beneficios en términos de operación, control y gestión de la red. El controlador SDN dispone la visión global de la red lo cual facilita y acelera los tiempos de respuesta en la resolución de problemas en la red y permite a los administradores de red implementar optimizaciones y aprovisionar nuevos servicios de manera ágil y expedita.

El análisis de los resultados reportados en la investigación de la arquitectura SDN/OpenFlow, permitió fundamentar un conjunto de consideraciones para apoyar a los administradores de red en el despliegue de redes SDN. Entre los aspectos más relevantes a considerar se encuentran los siguientes:

- Una solución SDN debe estar alineada con los objetivos y metas del negocio para la cual fue diseñada.
- Invertir una mayor cantidad de esfuerzo durante las pruebas de concepto de la solución permite detectar fallas a tiempo y agilizar los tiempos de despliegue en etapas posteriores de este.
- Un mecanismo para garantizar el funcionamiento de una nueva red SDN radica en contar con una buena infraestructura de monitoreo de red que permita efectuar chequeos y controles durante todas las etapas de un despliegue SDN.

Como trabajo futuro, planificamos desarrollar una herramienta que medición de desempeño para controladores SDN. Esta herramienta daría el tiempo de respuesta de un controlador cuando surge un cambio en un switch ya que este último debe comunicarse con el controlador para que le especifique las acciones a tomar. Similarmente, la herramienta podría inundar un controlador SDN de peticiones para estresarlo, y se reportaría la cantidad de peticiones que el controlador sería capaz de atender en condiciones de carga extrema. En este último caso, la herramienta se desarrollaría con una arquitectura distribuida de tal forma que se pueda someter una mayor carga de peticiones al controlador SDN.

REFERENCIAS

[1] Open Networking Foundation. *Software Defined Networking: The New Norm for Networks*. ONF White Paper. April 2012.

[2] Open Networking Foundation. *Migration Use Cases and Methods*. Migration Working Group. 2014.

[3] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. Van Reijendam, P. Weissmann, and N. Mckeown. *Maturing of OpenFlow and Software Defined Networking through Deployments*. Computer Networks: The International Journal of Computer and Telecommunications Networking. vol. 61, pp. 151-175. March 2014.

[4] J. Skorupa, M. Fabbi, and M. Fabbi. *Ending the Confusion about Software-Defined Networking: A Taxonomy*. Gartner Research, G00248592. March 2013.

[5] W. Xia, Y. Wen, C.H. Foh, D. Niyato, and H. Xie. *A Survey on Software-Defined Networking*. IEEE Communication Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, First Quarter 2015. 2015.

[6] A. Lara, A. Kolasani, and B. Ramamurthy. *Network Innovation using OpenFlow: A Survey*. IEEE Communications Survey & Tutorials. August 2013.

[7] P. Goransson and C. Black. *Software Defined Networks, A Comprehensive Approach*. Morgan Kaufmann. May 2014.

[8] P. Morreale and J. Anderson. *Software Defined Networking Design and Deployment*. CRC Press. 2015.

[9] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi Salim, D. Meyer, and O. Koufopavlou. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. RFC 7426. Internet Research Task Force. January 2015.

[10] H. Kim and N. Feamster. *Improving Network Management with Software Defined Networking*. IEEE Communications Magazine. vol. 51, no. 2. pp. 114-119. 2013.

[11] A. Porxas. *Virtualization-enabled Adaptive Routing for QoS-aware Software-Defined*. Master thesis. Universitat Politècnica de Catalunya, Spain, December 2014.

[12] D. Kreutz, F. Ramos, et al. *Software-Defined Networking: A Comprehensive Survey*. Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76. January 2015.

[13] B. Pfaff and B. Davie. *The Open vSwitch Database Management Protocol*. RFC 7047. Internet Research Task Force. December 2013.

[14] A. Doria, J. Hadi, R. Hass, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. *Forwarding and Control Element Separation (ForCES) Protocol Specification*. RFC 5810. Internet Research Task Force. March 2010.

[15] Y. Rekhter, T. Lid, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. Internet Research Task Force. January 2006.

[16] J. Case, M. Fedor, M. Schoffstall, and C. Davin. *A Simple Network Management Protocol (SNMP)*. RFC 1098. Internet Research Task Force. May 1990.

[17] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. *Network Configuration Protocol (NETCONF)*. RFC 6241. Internet Research Task Force. June 2011.

[18] L. Richardson and S. Ruby. *RESTful Web Services. Web Services for the Real World*. O'Reilly Media. May 2007.

[19] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California. Irvine, CA, USA, 2000.

[20] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi. *SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains*. Internet Engineering Task Force, Internet Draft. June 2012.

[21] JP. Vasseur and JL. Le Roux. *Path Computation Element (PCE) Communication Protocol (PCEP)*. RFC 5440. Internet Engineering Task Force. March 2009.

[22] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. *FlowVisor: A Network Virtualization Layer*. Deutsche Telekom Inc. R&D Lab, Stanford University, Nicira Networks. October 2009.

[23] A. Al-Shabibi, M. De Leenheer, A. Koshibe, G. Parulkar, B. Snow, M. Gerola, and E. Salvadori. *OpenVirteX: Make Your Virtual SDNs Programmable*, in proceedings of the Third Workshop on Hot Topics in Software Defined Networks (HotSDN 2014). Chicago, IL, USA, August 2014.

[24] Z. Bozakov and P. Papadimitriou. *AutoSlice: Automated and Scalable Slicing for Software-defined Networks*, in proceedings of the 2012 ACM Conference on CoNEXT Student Workshop (CoNEXT Student 2012), Nice, France, December 2012.

[25] S. Gutz, A. Story, C. Schlesinger, and N. Foster. *Splendid Isolation: A Slice Abstraction for Software-defined Networks*, in proceedings of the First Workshop on Hot Topics in Software Defined Networks (HotSDN 2012), Helsinki, Finland, August 2012.

[26] H. Kim and N. Feamster. *Improving Network Management with Software Defined Networking*. IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, 2013.

[27] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker. *Practical Declarative Network Management*, in proceedings of the First ACM Workshop on Research on Enterprise Networking (WREN 2009), Barcelona, Spain, August 2009.

[28] N. Foster, R. Harrison, M. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. *Frenetic: A Network Programming Language*, in proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP 2011), Tokyo, Japan, September 2011.

[29] C. Monsanto, N. Foster, R. Harrison, and D. Walker. *A Compiler and Run-time System for Network Programming Languages*, in proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2012), Philadelphia, PA, USA, January 2012.

[30] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker. *Modular SDN Programming with Pyretic*. Usenix, The Advanced Computing Systems Association, vol. 38, no. 5, October 2013.

[31] ITU. *Management Framework for Open Systems Interconnection (OSI) for CCITT Applications. ITU Recommendation X.700*. September 1992.

[32] F. Maccioni. *Network Automation with Ansible 2.1 and Beyond* [PowerPoint slides]. Cisco. September 2016.

[33] M. Bjorklund. *The YANG 1.1 Data Modeling Language*. RFC 7950. Internet Research Task Force. August 2016.

[34] A. Bierman, M. Bjorklund, and K. Watsen. *RESTCONF Protocol*. RFC 8040. Internet Research Task Force. January 2017.

[35] gRPC. gRPC *A High Performance, Open-Source Universal RPC Framework*. http://www.grpc.io.

[36] Open Networking Foundation. *OF-CONFIG 1.2. OpenFlow Management and Configuration Protocol. ONF TS-016*. 2014.

[37] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348. Internet Engineering Task Force. August 2014.

[38] P. Garg and Y. Wang. *NVGRE: Network Virtualization using Generic Routing Encapsulation*. RFC 7637. Internet Engineering Task Force. September 2015.

[39] B. Davie and J. Gross. *A Stateless Transport Tunneling Protocol for Network Virtualization (STT)*. Internet Engineering Task Force, Internet Draft. April 2016.

[40] N. McKeown. *OpenFlow: Enabling Innovation in Campus Networks*. Stanford University. March 2008.

[41] J. Davis. *Introduction to Software-Defined Networking (SDN) and Network Programability*. CiscoLive BRKSDN-1014 [Power Point-Slides]. 2015.

[42] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.0.0 Implemented (Wire Protocol 0x01)*. December 2009.

[43] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0x02)*. February 2011.

[44] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.2.0 Implemented (Wire Protocol 0x03)*. December 2011.

[45] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.3.0 Implemented (Wire Protocol 0x04)*. June 2012.

[46] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.4.0 Implemented (Wire Protocol 0x05)*. October 2013.

[47] Open Networking Foundation. *OpenFlow Switch Specification, Version 1.5.0 Implemented (Wire Protocol 0x06)*. December 2014.

[48] Linux Foundation. *OpenDaylight: Open Source SDN Platform*. https:// www.opendaylight.org.

[49] Project Floodlight. *Open Source Software for Building Software-Defined Networks*. http://www.projectfloodlight.org/floodlight.

[50] Ryu SDN Framework Community. *Component-Based Software Defined Networking Framework. Build SDN Agilely*. http://osrg.github.io/ryu.

[51] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou. *Feature-Based Comparison and Selection of Software Defined Networking (SDN) Controllers*, in proceedings of the 2014 World Congress on Computer Applications and Information Systems (WCCAIS 2014), Hammamet, Tunisia, January 2014.

[52] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky. *Advanced Study of SDN/OpenFlow Controllers*, in proceedings of the

9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR 2013), Moscow, Russian Federation, October 2013.

[53] Big Switch Networks. *SDN Controller Unified Network Control Plane*. http://www.bigswitch.com/products/SDN-Controller.

[54] Brocade. *Brocade SDN Controller Data Sheet*, 2016.

[55] Cisco. *Cisco Open SDN Controller 1.2 Data Sheet*. October 2015.

[56] HPE. *Aruba VAN SDN Controller Software Data Sheet*. 2017.

[57] NEC. PF6800 ProgrammableFlow Controller. https://www.necam.com/sdn/Software/SDNController.

[58] B. Lantz, B. Heller, and N. McKeown. *A Network in a Laptop: Rapid Prototyping for Software-Defined Networks*, in proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets IX), Monterey, CA, USA, October 2010.

[59] J. Yan and D. Jin. *VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation*, in proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Network Research (SOSR 2015), Santa Clara, CA, USA, June 2015.

[60] B. Linkletter. *OFNet SDN Network Emulator*. http://www.brianlinkletter.com/ofnet-a-new-sdn-network-emulator.

[61] S. Wang, C. Chou, and C. Yang. *EstiNet OpenFlow Network Simulator and Emulator*, IEEE Communications Magazine, vol. 51, no. 9. September 2013.

[62] J. Postel. *Internet Control Message Protocol*, RFC 792, Internet Engineering Task Force, September 1981.

[63] Cisco Systems. *Introduction to Cisco IOS NetFlow - A Technical Overview*, May 2012.

[64] sFlow. *Traffic Monitoring using sFlow*. 2003.

[65] A. Myers. *JFlow: Practical Mostly-Static Information Flow Control*, in proceedings of the 26th ACM Symposium on Principles of Programming Languages (POPL 1999), San Antonio, TX, USA, January 1999.

[66] S. Chowdhury, Md. F. Bari, R. Ahmed, and R. Boutaba. *PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks*, in proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS 2014), Krakow, Poland, June 2014.

[67] A. Tootoonchian, M. Ghobadi, and Y. Ganjali. *OpenTM: Traffic Matrix Estimator for OpenFlow Networks*, in proceedings of the 11th International Conference on Passive and Active Measurement (PAM 2010), Zurich, Switzerland, April 2010.

[68] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. Madhyastha. *Flowsense: Monitoring Network Utilization with Zero Measurement Cost*, in proceedings of the 14th International Conference on Passive and Active Measurement (PAM 2013), Hong Kong, China, March 2013.