TÉCNICA

DESARROLLO DE **APLICACIONES SDN**

Por: Ing. Dayrene Frómeta Fonseca, Dra.C. Caridad Anías Calderón, Ing. Susana Ballester Macías e Ing. Sergio León González, ISPJAE.
dayrene.ff@electrica.cujae.edu.cu, cacha@tesla.cujae.edu.cu; sballesterm@electrica.cujae.edu.cu; sleong@cujae.edu.cu

RESUMEN

Las Redes Definidas por Software (SDN) han modificado la forma de diseñar y operar las redes de datos. Las SDN son aplicaciones basadas en software que determinan el comportamiento de la red, pudiendo modificar el estado de la misma de acuerdo a sus necesidades.

En este artículo se clasifican las aplicaciones SDN, se muestran las principales formas que existen hoy en día para su creación y se proponen herramientas que pueden ser usadas con este fin. Así mismo, se abordan algunos aspectos importantes para el desarrollo de aplicaciones SDN, los cuales sirven de guía a los programadores en el proceso de creación de dichas aplicaciones y en la selección de herramientas para su desarrollo.

Palabras clave: Redes Definidas por Software (SDN), Aplicaciones SDN, Interfaz de Programación de Aplicaciones (API)

ABSTRACT

Software defined networks (SDN) have modified the way of designing and operating data networks. In the SDN, the software-based applications are the ones that determine the network behavior, modifying its status according to its needs.

In this article, the applications SDN are classified, the existing main ways to create them are shown, and the tools used for that purpose are proposed. Likewise, the author makes reference to some crucial aspects to the development of SDN applications, which guide the programmers in the process of creation of such applications and selection of tools for their development.

Key words: Software Defined Networks (SDN), SDN Applications, Applications Programming Interface (API)

Introducción

Las Redes Definidas por Software (SDN) son un nuevo paradigma que promete dar solución a los problemas que enfrentan las redes de telecomunicaciones tradicionales, pues permiten la configuración personalizada de la red y sus servicios, a partir del uso de aplicaciones de software que pueden llegar a desarrollar los propios administradores de red.

En la figura 1 se muestra la arquitectura básica de las SDN, sus componentes fundamentales y las interacciones entre ellos.

Como puede observarse, las SDN proponen una arquitectura donde el **plano de control** (la inteligencia de los dispositivos de red) y el **plano de reenvío** (responsable del envío de los paquetes) se encuentran desacoplados, separando así las funciones de gestión de red y de conmutación de paquetes.

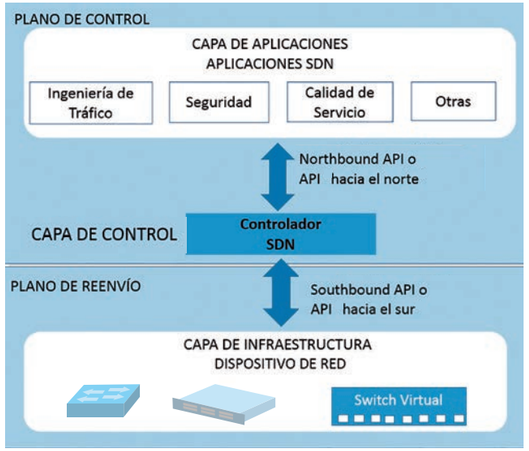

Figura 1. Arquitectura SDN. Fuente: [1].

Uno de los componentes más interesantes de la arquitectura SDN son las Interfaces de Programación de Aplicaciones —*Application Programming Interface* (API)—. La incorporación de estas API hace posible que los administradores puedan desarrollar sus propias aplicaciones SDN con relativa facilidad y que mediante ellas configuren, administren, aseguren y optimicen los recursos de la red de acuerdo a sus necesidades. En este contexto son las aplicaciones SDN y la posibilidad de que estas sean desarrolladas en donde se necesiten dos de las características más atractivas que presentan las Redes Definidas por Software.

Para lograr que las SDN sean redes altamente programables, algunos grupos de investigación han dirigido su atención al desarrollo de herramientas que permiten el despliegue de diversos servicios y aplicaciones para este tipo de redes, aunque las herramientas de programación tradicionales también pueden ser utilizadas para este fin.

Como resultado de lo antes expuesto existe una amplia gama de herramientas, tanto comerciales como de código abierto, que están a disposición de los desarrolladores de aplicaciones SDN, así como numerosas guías y tutoriales que los fabricantes ofrecen para la utilización de dichas herramientas.

Sin embargo, los programadores tienen acceso a estos equipos y a su documentación solo de manera aislada. Durante el desarrollo de esta investigación no se contaba con ningún documento que recogiese las diferentes formas

que existen hoy en día para la creación de aplicaciones SDN, ni la gran variedad de herramientas que pueden ser usadas con este fin. Lo anterior impide que los desarrolladores comparen las herramientas disponibles y puedan determinar, de acuerdo a sus características, cuál es la más apropiada para la creación de una aplicación SDN específica. Esta situación influye negativamente en el desarrollo y despliegue de las Redes Definidas por Software.

Dado que en las SDN son las aplicaciones quienes determinan el comportamiento de la red, se considera de gran importancia toda investigación realizada en favor de agilizar el proceso para su desarrollo. El objetivo de este artículo es exponer algunos conceptos importantes relacionados con el desarrollo de aplicaciones SDN, presentar a los programadores alternativas y herramientas disponibles hoy en día para crear dichas aplicaciones y recomendar herramientas que pueden ser usadas para este fin. Por lo anterior, se considera que este documento constituye una guía a los programadores en el proceso de selección de herramientas para el desarrollo de aplicaciones SDN.

Aplicaciones SDN

Una aplicación SDN consiste en un programa de software diseñado para realizar una o varias tareas en el entorno de las Redes Definidas por Software, pudiendo reemplazar y ampliar las funcionalidades que se implementan en los dispositivos de hardware de una red tradicional.

Las aplicaciones SDN tienen la capacidad de cambiar el comportamiento de la red en tiempo real, de tal forma que esta pueda responder rápidamente a los objetivos de una empresa, como puede ser, reenviar paquetes por una ruta menos costosa, adaptar dinámicamente la calidad de servicio por disponibilidad del ancho de banda y solicitud de los usuarios, descartar paquetes sospechosos y rastrear su fuente para bloquearlos, entre otras.

Clasificación de las aplicaciones SDN

Las aplicaciones SDN pueden clasificarse atendiendo a tres parámetros fundamentales: la función que desempeñan en la red, la plataforma de software sobre la que corren y la complejidad de las mismas.

Las aplicaciones SDN pueden ser categorizadas atendiendo a la función específica que desempeñan, como aplicaciones de seguridad, calidad de servicio (QoS), ingeniería de tráfico (TE), gestión de las listas de control de acceso (ACL), balanceo de carga, etc. Otras aplicaciones SDN son desarrolladas para prestar servicios en ambientes específicos como la computación en la nube, las redes móviles, la virtualización de la red y la virtualización de las funciones de red (NFV) [1]. Atendiendo a la plataforma de software sobre la que corren, las aplicaciones SDN pueden clasificarse como **internas** o **externas** al controlador. Las aplicaciones internas son creadas como una extensión de las funcionalidades del controlador SDN y su código se ejecuta sobre la plataforma de software del mismo. Por otro lado, las aplicaciones externas pueden correr en una plataforma diferente a la del controlador, comunicándose con este a través de su API hacia el norte.

Por último, atendiendo a su complejidad, las aplicaciones SDN pueden clasificarse como aplicaciones **simples** o **complejas**. Las aplicaciones SDN simples son aquellas cuyo código cumple una sola función de red, mientras que las aplicaciones complejas poseen un código más sofisticado que hace que un mismo *switch OpenFlow* maneje eficiente y simultáneamente varias funciones de red.

Además de lo planteado anteriormente, algunas aplicaciones SDN son desarrolladas para cumplir objetivos específicos de un negocio y necesitan la interacción de varios módulos (internos y externos a la plataforma del controlador), para poder desempeñar las tareas que se proponen. Este tipo de aplicaciones son conocidas como **híbridas**, pues requieren de la interacción de aplicaciones SDN **internas** y **externas**. [2]

Ejemplos de aplicaciones SDN

Las aplicaciones SDN permiten a los controladores recopilar datos a través de servicios Web, leer, modificar, incorporar y eliminar entradas de las tablas de flujo de los dispositivos subyacentes; utilizar API para integrarse con otras aplicaciones; comprobar el estado de los dispositivos de red; activar y desactivar sus puertos y definir métricas que ayuden a la ingeniería de tráfico, entre otras funcionalidades.

No todas las aplicaciones SDN implementan funcionalidades completamente nuevas. Más bien, la mayoría de estas aplicaciones se basan o replican las funcionalidades de las aplicaciones que corren en los *routers* y *switche*s tradicionales. La novedad de las SDN es la manera en que integran estas aplicaciones y las facilidades que brindan para el desarrollo de otras nuevas.

Algunas de las aplicaciones que vienen incluidas en los controladores SDN constituyen módulos de conmutación simple, descubrimiento de topología, monitorización de la red, control de acceso a la red, cortafuegos, balanceo de carga, etc. También, se pueden mencionar otras aplicaciones SDN que son desarrolladas en función de las necesidades particulares de una empresa o para trabajar en ambientes específicos como son la computación en la nube, las redes móviles y la virtualización de las funciones de red, entre otros.

En [3], [4] y [5] se presentan varias aplicaciones SDN internas y Externas asociadas a los controladores NOX Classic, POX, Ryu, Floodlight, OpenDaylight, HP VAN Controller y Cisco XNC, siendo estos algunos de los controladores SDN comerciales y de código abierto disponibles actualmente.

Desarrollo de aplicaciones SDN

Las SDN poseen dos beneficios potenciales para mejorar las redes de computadoras: facilitar la innovación en las tecnologías de red y hacer que la creación, el despliegue y la composición de servicios y aplicaciones para las redes sea una tarea menos compleja [6]. Para explotar al máximo estos beneficios se encuentran a disposición de los desarrolladores múltiples herramientas, tanto comerciales como de código abierto, que permiten la creación de aplicaciones para las Redes Definidas por Software.

Muchos de los controladores SDN disponibles hoy en día extienden las funcionalidades de su plataforma al ofrecer a los programadores las herramientas necesarias para la creación de nuevos módulos de aplicaciones sobre ellos. Sin embargo, los controladores SDN de primera generación, tales como NOX [6], POX [6], Trema [7], Ryu [8], entre otros, no hacen de esto una tarea sencilla, pues ofrecen una interfaz de programación de bajo nivel de abstracción. Esto trae como consecuencia que los desarrolladores tengan que enfrentarse constantemente a dificultades que complican el proceso de creación de nuevas aplicaciones para la red. Una de las dificultades a las que pudieran enfrentarse los programadores al desarrollar aplicaciones sobre controladores SDN con la característica antes mencionada, es que deben tener en cuenta detalles de bajo nivel que poco tienen que ver con los servicios o las funciones que las aplicaciones van a desempeñar en la red. Por ejemplo, se ven obligados a especificar los patrones de comunicación entre el controlador y el *switch*, así como a negociar con aspectos de entendimiento difíciles, como la coordinación de eventos asíncronos, entre otros. [9]

Para evitar lo anterior han sido desarrollados nuevos lenguajes de programación que facilitan en gran medida el proceso de creación de aplicaciones para las SDN. Ejemplos de estos lenguajes son Frenetic [9], Pyretic [10], NetCore [11] y Procera, los cuales son considerados lenguajes específicos del dominio —*Domain Specific Language* (DSL)—, que permiten a los desarrolladores escribir programas con un estilo declarativo y composicional, y programar con un alto nivel de abstracción.

Además, como fue mencionado al presentar la arquitectura SDN, muchos controladores poseen hacia el norte una Interfaz de programación de Aplicaciones, que permite a los programadores crear aplicaciones SDN haciendo uso de herramientas de programación tradicionales que permitan consumir y/o exponer servicios a través de estas API.

Herramientas

A partir de lo expuesto anteriormente, se puede concluir que existen tres grupos de herramientas fundamentales para el desarrollo de aplicaciones SDN.

En el primero de los grupos se encuentran las herramientas que por defecto ofrecen los controladores SDN para la creación de aplicaciones sobre sus plataformas de software. Algunas de estas consisten en un lenguaje de programación (el lenguaje en el que haya sido desarrollado el controlador), Kits para el Desarrollo de Software —*Software Development Kit* (SDK)—, códigos de ejemplo, bibliotecas de Interfaces de Programación de Aplicaciones (API), entre otras herramientas que facilitan la creación de nuevas aplicaciones, como son los módulos de aplicaciones que ya vienen programados sobre la plataforma del controlador, cuyo código puede ser reutilizado para el desarrollo de nuevas aplicaciones.

Otro de los grupos de herramientas está constituido por las que ofrecen algunas plataformas de desarrollo de software tradicionales, las cuales permiten crear aplicaciones con la capacidad de consumir y/o exponer servicios a través de la API hacia el norte que usualmente presentan los controladores SDN. De esta forma, se pueden crear aplicaciones que, corriendo en una plataforma de software diferente a la del controlador, interactúen con los servicios que este ofrece y obtengan información del estado de la red y sean capaces de modificar el comportamiento de esta de acuerdo a los objetivos que se persigan.

Por último, está el grupo de herramientas constituido por las implementaciones de los nuevos lenguajes de programación para las SDN. Algunos de estos lenguajes ya están disponibles como *frameworks* que generalmente consisten en el lenguaje de programación más las herramientas apropiadas para compilar y traducir las reglas escritas con un alto nivel de abstracción a un lenguaje entendible por el controlador SDN, que usa para la comunicación con los switches subyacentes. Es decir, estas implementaciones consisten en un nuevo controlador SDN, que permite escribir las reglas que dictan el comportamiento de la red con un lenguaje que proporciona un alto nivel de abstracción. Las implementaciones disponibles de los DSL para las Redes Definidas por Software son conocidas como *frameworks* de programación para las SDN.

Formas de desarrollo

Como resultado de la investigación realizada y teniendo en cuenta las herramientas existentes fue posible identificar tres formas fundamentales para el desarrollo de las aplicaciones SDN.

La primera forma es desarrollar estas aplicaciones como internas a la plataforma de software de los controladores SDN de primera generación, haciendo uso de las herramientas que estos ofrezcan para ello. Esta vía obliga a los desarrolladores a programar con un bajo nivel de abstracción.

La segunda forma es desarrollar las aplicaciones SDN como módulos internos a alguna de las plataformas de software que implementan los nuevos lenguajes de programación para las SDN, permitiendo a los programadores trabajar con un alto nivel de abstracción.

La última alternativa es crear aplicaciones SDN externas a la plataforma de software del controlador SDN que esté operando en la red, usando herramientas para el desarrollo de aplicaciones que permitan consumir y/o exponer servicios a través de la API hacia el norte que este ofrezca.

Como consecuencia, a la hora de desarrollar una aplicación SDN específica, los programadores pueden enfrentarse a interrogantes como: ¿cuál de los grupos de herramientas

disponibles es más apropiado para el desarrollo de una aplicación en cuestión y qué elementos se deban tener en cuenta para seleccionar entre esas herramientas? ¿es más factible desarrollar la aplicación en cuestión como interna o externa a la plataforma del controlador? y ¿qué parámetros se deben tener en cuenta al desarrollar aplicaciones internas o externas?

A continuación, se abordan algunos aspectos importantes que ayudan a los programadores a seleccionar las herramientas y la forma más apropiada para desarrollar una aplicación SDN específica, de esta forma, se dará respuesta a las interrogantes planteadas anteriormente.

Recomendaciones para el desarrollo de aplicaciones SDN y para la selección de las herramientas

Las recomendaciones que se presentan a continuación permiten que el programador, teniendo como base las características de la aplicación a desarrollar, sea capaz de determinar cuál de las formas de desarrollo es la más adecuada y de seleccionar el grupo de herramientas apropiadas para su creación. Consecuentemente, se propone como primer paso realizar una descripción de la aplicación que se quiere desarrollar.

Existen varios aspectos que pudieran tenerse en cuenta para caracterizar las aplicaciones SDN. Sin embargo, inicialmente, es suficiente con que el programador defina la función específica de red (enrutamiento, seguridad, calidad de servicio, gestión de dispositivos, etc.) que va a realizar la aplicación a desarrollar o si se va a implementar una combinación de esas funciones. Con esta información se puede clasificar la aplicación como simple o compleja y determinar si es necesario que interactúe frecuentemente con los *switches* subyacentes o con los servicios que ofrece el controlador SDN que esté operando en la red.

Anteriormente se dijo que los controladores SDN de primera generación ofrecen a los programadores una interfaz de programación de bajo nivel de abstracción. Esta condición se hace mucho más notable y perjudicial si la aplicación a desarrollar debe cumplir varias funciones de red al mismo tiempo. Muchas aplicaciones SDN requieren que el mismo tráfico sea procesado de diferentes formas, por ejemplo, una aplicación debe enrutar el tráfico basándose en la dirección IP destino, y a su vez debe monitorizarlo por la dirección IP fuente. Se debe aplicar una política de control de acceso para eliminar el tráfico de una fuente no deseada y luego enrutar el tráfico restante por la dirección IP destino. En estos casos, los programadores deben trabajar con especial cuidado para obtener los resultados que desean, pues las reglas creadas para cumplir con una función de red específica pueden anular las reglas de otra. Para garantizar el buen desempeño de estas aplicaciones complejas, los programadores deben combinar manualmente la lógica de reenvío para cada función que la aplicación va a desempeñar, lo cual es una tarea bastante tediosa y difícil cuando se trabaja con una interfaz de programación de bajo nivel de abstracción.

Teniendo en cuenta lo antes planteado, si la aplicación a desarrollar es clasificada como compleja, se recomienda que sea creada usando alguno de los lenguajes de programación desarrollados especialmente para las SDN. Sin embargo, estos nuevos *frameworks* de programación para las SDN, no son distribuciones independientes del controlador SDN sobre el cual están implementadas. Es decir, estos *frameworks* no pueden ser instalados como *plug-ins* sobre el controlador SDN que esté operando en la red, y para poder escribir nuevas aplicaciones de red haciendo uso de los lenguajes que estos *frameworks* implementan, habrá que usarlos como el nuevo controlador de la red. Además, es importante mencionar que el empleo de los mismos también tiene sus inconvenientes. Una de las desventajas de su uso es que las aplicaciones escritas con los nuevos lenguajes de programación para las SDN no podrán exponer sus servicios a otras aplicaciones, solamente podrán determinar el estado de los dispositivos subyacentes haciendo uso de la interfaz que ofrezca el controlador SDN hacia el sur (el protocolo OpenFlow).

Por lo anterior, como puede observarse en la figura 2, donde se muestran las recomendaciones para el desarrollo de aplicaciones SDN complejas, lo primero es que el programador decida si creará la aplicación usando los nuevos *frameworks* de programación para las SDN o no. Algunos aspectos que se pudieran considerar para tomar esta decisión pueden ser: si está dispuesto a instalar un nuevo controlador para la red, la experiencia que posea en el uso del lenguaje de programación que el *framework* implementa y las desventajas que tiene el uso de estos nuevos lenguajes.

Si el programador decide desarrollar la aplicación usando los nuevos lenguajes de programación para las SDN, ya que estos permiten escribir complejos programas de red sin preocuparse por los detalles de los *switches* subyacentes, entonces, debe proceder a instalar la implementación del

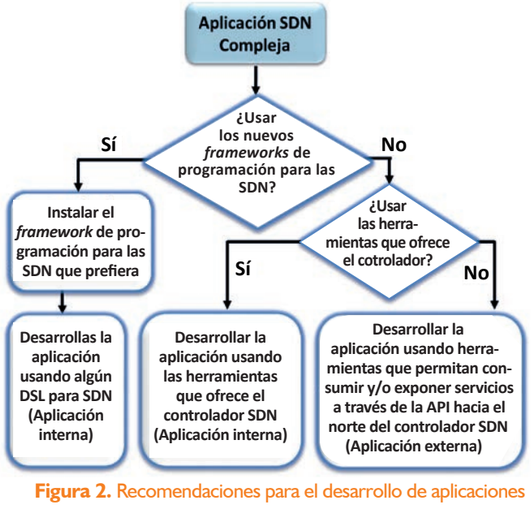

Figura 2. Recomendaciones para el desarrollo de aplicaciones SDN complejas . Fuente: Elaboración propia.

lenguaje que prefiera. A partir de ese momento, todas las aplicaciones para la red (simples y complejas) tendrán que ser desarrolladas usando el lenguaje de programación que el nuevo controlador implemente.

Las implantaciones de los nuevos lenguajes de programación Pyretic (sobre los controladores POX y Ryu) y Frenetic (sobre el controlador NOX) están disponibles en GitHub [12].

Por otro lado, como se muestra en la figura 2, si la aplicación a desarrollar es compleja, pero el programador prefiere no usar los nuevos *frameworks* de programación para las SDN, entonces se proponen otras dos opciones para el desarrollo de la aplicación: crearla usando las herramientas que para ello el controlador SDN provee, pues a pesar de ofrecer una interfaz de programación de bajo nivel que dificulta el desarrollo de aplicaciones complejas, lo cierto es que no lo hace imposible; o crearla en una plataforma para el desarrollo de software diferente a la del controlador, usando herramientas que permitan que este consuma y/o exponga servicios a través de la API hacia el norte que ofrece.

Los controladores SDN vienen programados con la capacidad de brindar ciertos servicios de red. Muchos de ellos incluyen módulos para descubrimiento de topología, gestión de dispositivos y enrutamiento, entre otros, que pueden ser usados como base para el desarrollo de nuevas aplicaciones para las SDN.

Las aplicaciones SDN, generalmente, necesitan interactuar con algunos de los servicios antes mencionados. Por ejemplo, puede que necesiten conocer la topología de la red en un momento determinado, el estado de los enlaces (activos o inactivos), cuáles son las computadoras que están conectadas a la red, entre otras informaciones que pueden ofrecer los controladores. Algunas aplicaciones SDN, para desempeñar correctamente sus funciones, necesitan interactuar frecuentemente con dichos servicios o constantemente tendrán que intercambiar información con los *switches* subyacentes. Tal es el caso de aplicaciones que requieren el reenvío de paquetes, la manipulación de eventos *Packet-in*, o ejercer un control muy granular sobre los flujos de datos. Asímismo, algunas aplicaciones SDN deben responder rápidamente ante los eventos que ocurren en la red, pues un *switch* no puede retener un paquete esperando a que llegue la respuesta de cómo procesarlo, desde una aplicación con alta latencia.

Por otra parte, existen aplicaciones que no están pensadas para ejercer un control tan granular sobre los flujos de datos, ni es necesario que reaccionen rápidamente ante los eventos que ocurren en la red. Tal es el caso de aplicaciones para el abastecimiento de rutas, la inspección de flujos u otras que permiten monitorear el comportamiento de la red y del controlador SDN de forma remota.

Por ejemplo, se puede necesitar una aplicación para mover cinco Terabytes de datos entre dos extremos de un Centro de Datos, teniendo estos que atravesar una gran malla de *switches* para llegar a su destino. La aplicación a desarrollar debe proveer una ruta que garantice la mejor manera de que esta gran cantidad de datos atraviese la red. En función de esto, debe inspeccionar la topología para planificar la mejor ruta, instalar las nuevas entradas en las tablas de flujo de los *switches* subyacentes y especificar que estas entradas deben permanecer por varias horas o días. Este tipo de aplicaciones pueden programar todas las reglas necesarias desde una plataforma diferente a la del controlador SDN y comunicarse con este a través de su API hacia el norte, pues por la función va a desempeñar la función de abastecimiento de ruta. La aplicación no necesita interactuar frecuentemente con los servicios ofrecidos por la plataforma del controlador, ni con los *switches* subyacentes. Además, estas aplicaciones pueden demorar en realizar todas sus funciones sin que constituya un inconveniente. [2]

Las aplicaciones de este tipo no necesitan cumplir sus tareas con inmediatez, por lo que no es necesario que posean una baja latencia. Por el contrario, suelen ser muy efectivas por el simple hecho de actuar preventivamente, y para lograrlo, no necesitan interactuar frecuentemente con los servicios ofrecidos por el controlador SDN, ni intercambiar información constantemente con los *switches* subyacentes.

Como se muestra en la figura 3, donde se resumen las recomendaciones para el desarrollo de aplicaciones SDN simples, lo primero es que el programador determine si es necesario que la aplicación a desarrollar interactúe frecuentemente con los servicios ofrecidos por el controlador SDN o con los *switches* subyacentes. Si este determina que no es necesario que la aplicación interactúe frecuentemente con los servicios del controlador SDN ni con los *switches* subyacentes, se recomienda que sea creada sobre una plataforma para el desarrollo de software diferente a la del controlador (como una aplicación externa), y haciendo uso de herramientas que permitan consumir y/o exponer servicios a través de la API hacia el norte que este ofrezca generalmente a través de una API REST, aunque puede ser otra, pues la interfaz para la comunicación entre ambos aún no ha sido estandarizada. De esta forma, la aplicación no tiene que compartir innecesariamente con otros módulos los recursos del controlador, como puede ser la memoria y la capacidad de procesamiento. Otra de las ventajas del desarrollo de las aplicaciones SDN externas es que no obliga al programador a trabajar en el lenguaje de programación en el que se basa el controlador, sino que podrá hacerlo usando cualquier otro lenguaje que prefiera (Java, C, Python, etc), siempre que le permita consumir y/o exponer servicios a través de la API hacia el norte que el controlador ofrezca.

Actualmente, son muchos los lenguajes de programación que ofrecen herramientas para consumir y/o exponer servicios a través de APIs basadas en la arquitectura REST. Entre ellas, se encuentran el *framework* ASP. NET Web API para la Plataforma .NET de Microsoft [18], el *framework* Spring (a partir de la versión 3.0) para la plataforma Java, las bibliotecas Django REST *Framework* 3.0 y Python Request de Python y el *framework* jQuery de JavaScript, por solo mencionar algunos.

Por otro lado, si la aplicación a desarrollar es simple y el programador determina la necesidad de que esta interactúe frecuentemente con el controlador SDN o con los *switches* subyacentes por la función que va desempeñar en la red, (Figura 3) se recomiendan dos opciones para el desarrollo de la aplicación: usar las herramientas que ofrece por defecto la plataforma del controlador que esté operando en la red o utilizar alguno de los nuevos *frameworks* de programación para las SDN. En ambos casos, la aplicación desarrollada va a ejecutarse sobre la plataforma software del controlador, por lo que puede ser clasificada como una aplicación SDN interna.

En este punto, el programador debe determinar de acuerdo a sus preferencias la forma de desarrollar la aplicación, teniendo en cuenta las desventajas que pudiera traer el uso de los nuevos lenguajes y que para usarlos debe instalar alguno de los controladores que lo implementan.

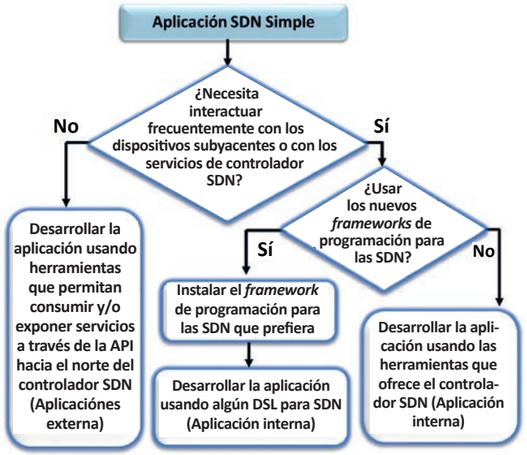

Figura 3. Recomendaciones para el desarrollo de aplicaciones SDN simples. Fuente: Elaboración propia.

Una vez presentadas algunas recomendaciones para la selección de herramientas para el desarrollo de aplicaciones SDN, es importante hacer notar que aunque en algunos casos la utilización de ciertas herramientas sea lo más apropiado, realmente la selección final que se haga va a depender en gran medida de las necesidades, los conocimientos y la experiencia que posean los programadores en el trabajo con las mismas, así como del tiempo que dispongan para el desarrollo de las aplicaciones.

Por otra parte, el desarrollo de aplicaciones SDN depende en gran medida de las características y posibilidades que brinde el controlador que opera en la red. En particular, la creación de aplicaciones internas depende del lenguaje que este ofrezca y de los servicios que brinde de forma predeterminada (descubrimiento de topología, enrutamiento, configuración de dispositivos, etc), los cuales pueden ser utilizados por las nuevas aplicaciones. Por otro lado, el desarrollo de aplicaciones externas depende de la interfaz de Programación de Aplicaciones que el controlador ofrezca hacia el norte, pues, en dependencia del tipo de API, se seleccionarán las herramientas necesarias para garantizar que las aplicaciones externas puedan consumir y/o exponer servicios a través de la misma.

Por ejemplo, las herramientas mencionadas para el desarrollo de aplicaciones SDN externas están pensadas para controladores que presenten una API REST hacia el norte.

Adicionalmente, cuando fueron clasificadas las aplicaciones SDN se planteó que algunas de ellas son conocidas como híbridas, pues para desempeñar eficientemente las tareas que se proponen necesitan la interacción de varios módulos internos y externos a la plataforma del controlador. Las recomendaciones realizadas en este apartado también pueden ser consideradas para la selección de las herramientas para el desarrollo de este tipo de aplicaciones. En este caso se debe aplicar a cada módulo que conforma la aplicación híbrida, ayudando a seleccionar entonces las herramientas adecuadas para el desarrollo de cada uno de ellos por separado.

Características del desarrollo de aplicaciones SDN Internas

Una característica común de las aplicaciones SDN internas, independientemente de las herramientas con las que hayan sido desarrolladas, es el hecho de que necesitan interactuar frecuentemente con algunos servicios que corren en la plataforma del controlador, como pueden ser los módulos para reenvío de paquetes, intercambio de mensajes *OpenFlow* con los *switches* subyacentes y descubrimiento de topología, entre otros que pudiera brindar el controlador SDN que se esté usando en la red. Además, estas aplicaciones deben compartir con los módulos existentes en la plataforma del controlador los recursos de la misma, tales como la memoria y el espacio de procesamiento.

Es importante destacar que las aplicaciones creadas, haciendo uso de las herramientas que ofrece por defecto la plataforma del controlador, deben ser desarrolladas en el lenguaje de programación en el que este haya sido implementado. Además, estas aplicaciones no solo pueden consumir los servicios que ofrece el controlador SDN, sino que también pueden hacer públicos sus propios servicios para que sean utilizados por otras aplicaciones internas (generalmente a través de API Java) o externas a la plataforma del controlador a través de la API hacia el norte.

Por medio de la API hacia el norte del controlador, una aplicación interna también puede consumir los servicios que ofrezcan otras aplicaciones externas. Por ejemplo, una aplicación que cumpla la función de cortafuegos, puede actualizar su base de información cada cierto tiempo desde una base de datos que resida en un servidor remoto, lo que puede hacer a través de la API REST que muchos controladores SDN ofrecen.

Además, los programadores pueden diseñar aplicaciones SDN internas que permitan a los administradores de red interactuar con las funcionalidades que las mismas brindan. Estos podrán hacerlo a través de una API REST, como fue mencionado anteriormente, o a través de una GUI, siempre que el controlador SDN la posea. Muchos controladores, a fin de hacer seguras estas interacciones, incluyen *frameworks* para la autenticación que pueden ser usadas por las nuevas aplicaciones desarrolladas.

Según lo planteado anteriormente, algunos aspectos a tener en cuenta para el desarrollo de aplicaciones SDN internas son:

−El lenguaje de programación en que se desarrollará, lo cual depende del lenguaje en el que se basa el controlador SDN sobre el que va a correr.

−Si va a exponer o no sus servicios para que sean utilizados por otras aplicaciones internas.

−Si va a consumir la API hacia el norte del controlador para usar los servicios que ofrecen otras aplicaciones externas.

−Si va a permitir que aplicaciones externas hagan uso de sus funcionalidades a través de la API hacia el norte o GUI que el controlador ofrezca.

−Si va a hacer uso, para cumplir sus funcionalidades, de algunos servicios que se ofrezcan de forma predeterminada en la plataforma del controlador.

−Si se va a integrar con algún *framework* que ofrezca el controlador para la autenticación, o va a usar algún otro mecanismo para hacer segura la interacción con otras aplicaciones a través de la red.

Además, es importante hacer notar que no todos los controladores ofrecen las mismas posibilidades para el desarrollo de aplicaciones internas, pues depende de las prestaciones que sus fabricantes hayan decidido agregarles, las cuales pueden facilitar, en mayor o menor medida, el desarrollo de nuevas aplicaciones.

Conclusiones

En las SDN las aplicaciones son notables, por ello es muy importante conocer cómo desarrollarlas. En función de esto, la investigación presentada clasifica las aplicaciones SDN de acuerdo a la función que desempeñan en la red (seguridad, calidad de servicio, ingeniería de tráfico, etc.); a la plataforma de software sobre la que corren (Internas y Externas al controlador SDN) y a la complejidad de las mismas (Simple o Complejas).

Teniendo como base tal clasificación se presenta un conjunto de recomendaciones útiles para el desarrollo de aplicaciones SDN (Internas o Externas, Simples o Complejas) y para la selección de las herramientas necesarias para ello, destacándose las posibilidades de los nuevos lenguajes de programación para la creación de aplicaciones SDN complejas.

Referencias bibliográficas

[1] Y. Jarraya, T. Madi, and M. Debbabi, "A Survey and a Layered Taxonomy of Software-Defined Networking," IEEE COMMUNICATION SURVEYS & TUTORIALS, vol. 16, p. 26, 2014.

[2] HP, "HP SDN Controller Architecture," in Technical Solution Guide, ed, 2013, p. 14.

[3] M. Sidana. (2014). Northbound applications on SDN controllers. Available: http://opensdnetworking.blogspot.com/2014/09/north bound-applications-on-sdn.html

[4] Cisco. (2014). Cisco Extensible Network Controller Data Sheet. Available: http://cisco.com/c/en/us/products/collateral /cloud-systems-management/extensiblenetwork-controller-xnc/data_sheet_c78-729453.html

[5] HP. (2014, Data sheet HP VAN SDN Controller Software.

[6] S. Rao. (2015). SDN Series Part Three: NOX, the Original OpenFlow Controller. Available: http://thenewstack.io/sdn-seriespart-iii-nox-the-original-openflowcontroller/

[7] S. Rao. (2015). SDN Series Part Two: Trema, a Framework for Developing OpenFlow Controllers in Ruby and C - The New Stack. Available: http://thenewstack.io/sdn-series-part-iitrema-a-framework-for-developingopenflow-controllers-in-ruby-and-c/

[8] T. F. Rui Kubo, Yuji Agawa and Hikaru Suzuki. (2015). Ryu SDN Framework. Available: https://www.nttreview.jp/archive/ntttechnical.php?contents=ntr201408fa4.html

[9] N. Foster, R. Harrison, M. J. Freedman, J. Rexford, and D. Walker. (2012, Frenetic: A High-Level Language for OpenFlow Networks. 10.

[10] C. M. JOSHUA REICH, NATEFOSTER, and A. D. W. JENNIFER REXFORD. (2013, Modular SDN Programming with Pyretic.

[11] C. Monsanto, N. Foster, R. Harrison, and D. Walker, "A Compiler and Run-time System for Network Programming Languages," p. 14, 2013.

[12] (2015). Frenetic-lang/frenetic-vm. Available: https://github.com/frenetic-lang/frenetic-vm

(Artículo recibido en noviembre de 2015 y aprobado en febrero de 2016)