

Software Defined Networks

Priscilla Piedra y Martín Flores
 Escuela de Ingeniería en Computación
 Instituto Tecnológico de Costa Rica. Cartago, Costa Rica
 {ppiedra90, mfloresg}@gmail.com

Resumen—Aquí va el abstract

1. INTRODUCCIÓN

Las Redes Definidas por Software (o SDN por sus siglas en inglés) es una nueva tecnología o paradigma aplicada a las redes con el fin de aportar agilidad en redes que son estáticas o complejas para aplicar cambios que pueden estar dedicadas a brindar servicios individuales. El concepto de SDN busca la administración de diferentes servicios de forma dinámica aplicados en una infraestructura de hardware común. Por ejemplo, con una red SDN se podría controlar la red según la demanda que esta tenga y sacar provecho al optimizar las cargas de la red.

Algunas ventajas de las redes SDN se ven reflejadas en la disminución de gastos en recurso físico, la centralización de la información permitiendo que se pueda analizar la red y crear rutas óptimas de forma determinista, atender de forma dinámica las peticiones de las aplicaciones, optimizan el uso de la red sin sacrificar la calidad del servicio y puede filtrar paquetes que viajan por la red añadiendo seguridad a la red pues se pueden redireccionar paquetes maliciosos.

Las redes SDN surgen de la necesidad de una nueva arquitectura de redes. Parte de las necesidades que se querían cubrir con un nuevo paradigma de red eran: cambiar el tráfico según patrones (definir redes públicas, privadas, etc), manejo de redes ubicadas en diferentes regiones, acceso a la red desde diferentes dispositivos, inicio de los servicios cloud y aumento en almacenamiento de grandes cantidades de datos son algunas de las principales razones para comenzar a ver cambios en la implementación de las redes.

Las redes tradicionales tienen ciertas limitaciones como la jerarquía en la que se basa su construcción, la complejidad en escalabilidad de dispositivos que requieren nueva programación y configuraciones, dependencia de proveedores, diferencias entre los requerimientos del mercado y las capacidades de las redes. Estas son solo algunas de las principales razones para buscar una solución que permitiera inyectar dinamismo en la red.

2. BREVE RESEÑA

Aunque el éxito de SDN se ha hecho más palpable recientemente, muchas de las ideas de la tecnología subyacente ha evolucionado por más de 20 años (o más). De algún modo, SDN retoma ideas de las primeras redes telefónicas,

Este documento fue realizado durante el curso Redes de Computadoras Avanzadas, impartido por el profesor Luis Carlos Loaiza Canet. Programa de Maestría en Computación, Instituto Tecnológico de Costa Rica. Segundo Semestre, 2017.

que tenían una clara separación de los planos de control y datos para simplificar la administración de la red y la agregación de nuevos servicios. Aún así, interfaces abiertas tal y como OPENFLOW permiten mayor innovación en las plataformas de controladores y aplicaciones que lo que se podía hacer en redes cerradas diseñadas para un limitado rango de servicios telefónicos. De igual forma, SDN parece retomar investigación pasada en *active networking* que introdujo conceptos de para redes programables, aunque con un énfasis en planos de datos programables. SDN también se relaciona con otros trabajos previos en separación de los planos de control y datos en redes de computadoras.

La historia de SDN inició más de 20 años atrás, justo cuando Internet estaba despegando. Muchas de las innovaciones en el comunidad de investigación en redes fueron influenciadas de alguna forma u otra por el progreso de otras áreas tales como sistemas distribuidos, sistemas operativos y lenguajes de programación. Los esfuerzos para crear una infraestructura de red programable también estuvieron claramente relacionados con el trabajo en el soporte programable de procesamiento de paquetes a altas velocidades.

En [4] se divide la historia de SDN en tres etapas:

1. ***active networks*** mediados de los 90 a inicios del 2000: que introdujo funciones programables en la red, lo cual condujo a una mayor innovación.
2. **Separación de los planos de control y datos** alrededor del 2001 al 2007, en donde se desarrolló interfaces abiertas entre los planos de control y datos.
3. **El API de OPENFLOW y sistemas operativos de red** del 2007 al 2010, que representó el primera adopción generalizada de una interfaz abierta y desarrolló forma para hacer la separación de los planos de control y datos escalable.

La virtualización de redes también jugó un rol importante a través de la evolución histórica de SDN como uno de los primeros casos de uso significativos para SDN.

3. SOFTWARE DEFINED NETWORKS

El *Open Network Foundation* – la organización encargada de promover la implementación de redes SDN y estándares para las mismas - define SDN como la separación física del plano de control (*control plane*) de la red del plano de datos

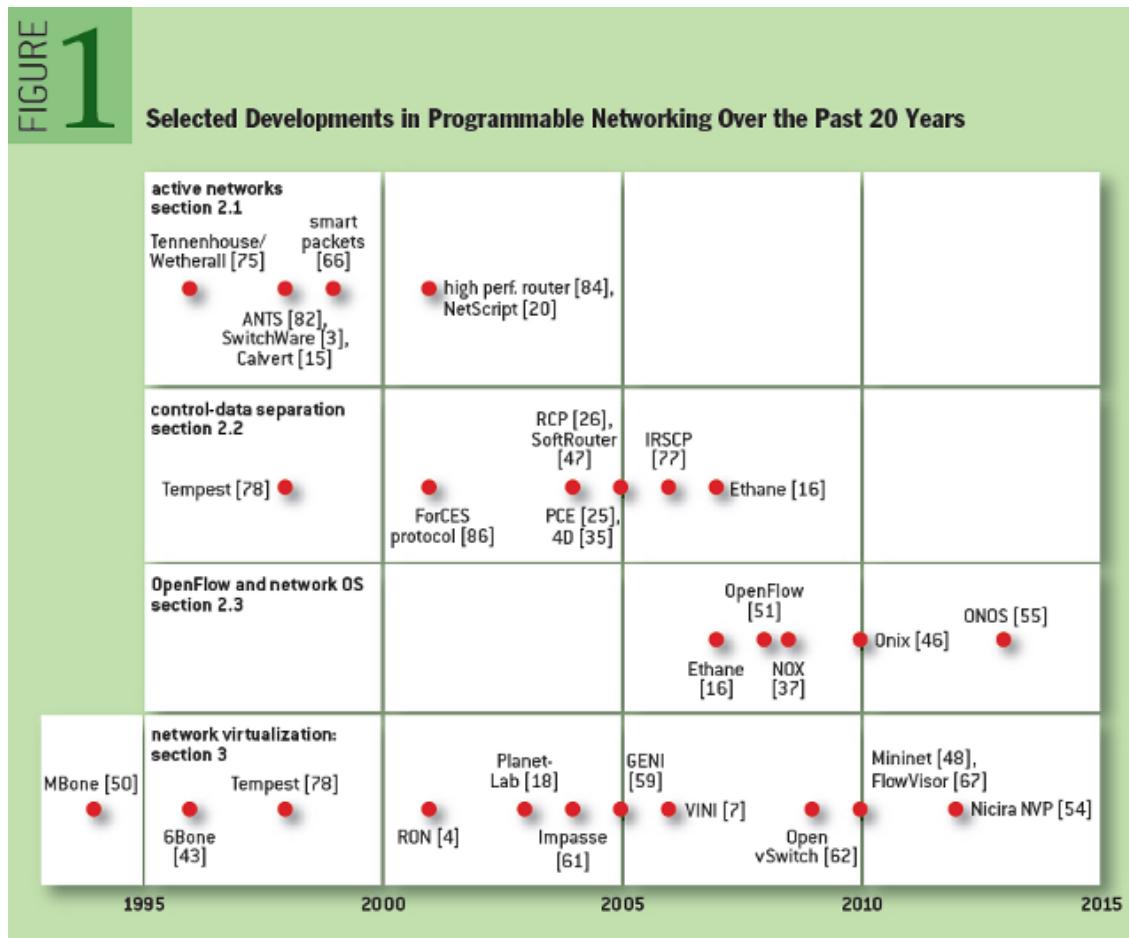


Figura 1. Desarrollos selectos en el campo de redes programables en los últimos 20 años. Tomado de *The Road to SDN* [4].

o reenvío (*forwarding plane*), en donde un plano de control controla varios dispositivos [2].

El objetivo de las redes SDN es brindar una interfaz abierta que pueda permitir el desarrollo de software para controlar la conectividad dentro de una red, el flujo y el tráfico. La idea central es desacoplar el control de la red del de datos y que además sea programable. Al tener una separación entre la parte lógica y la física se busca abstraer los servicios y las aplicaciones los cuales pueden tratar la red como una entidad lógica o virtual.

3.1. Arquitectura

Los dispositivos de red tradicionales, como un *switch*, tiene 3 planos: el plano de administración, el plano de control y el plano de datos. El plano de administración permite la configuración del dispositivo y el de control, que controla el enruteamiento del tráfico. La tabla de ruteo la llena el plano de control (por un algoritmo de enruteamiento) que a su vez determina cómo el tráfico debería ser enruteado por el hardware. Finalmente, la capa de datos se encarga de este enruteamiento. Uno de los problemas con las redes tradicionales, el cual SDN está intentando cambiar, es que cada dispositivo en la red tiene su propio plano de administración, control y de datos. Si la red tiene 100 *switches*, entonces se tienen 100 planos de administración, 100 de control y 100 de datos. Se tiene que administrar cada dispositivo

por separado, cada uno ejecutando un algoritmo distribuido de enruteamiento como *Open Shortest Path first*(OSPF) que tiene que determinar la mejor ruta de un punto de la red al otro y que, con suerte tendrá conocimiento completo de la red o al menos uno bueno de la misma. OSPF y otros algoritmos distribuidos de enruteamiento son complejos y no necesariamente eligen la mejor ruta porque no tienen una vista del tamaño total de la red.

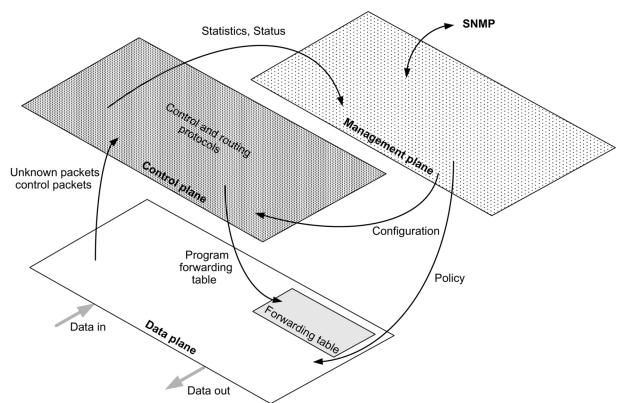


Figura 2. Arquitectura de un dispositivo tradicional. Fuente: *Software Defined Networks* [5].

Con SDN, la idea es remover el “cerebro” o la “inteli-

gencia” del dispositivo de red y llevarlo a un dispositivo diferente como un **controlador**: un dispositivo que actúa como un punto de control estratégico – probablemente compuesto de varios servidores físicos- que controla varios dispositivos de red tanto físicos como virtuales y que tiene mejor conocimiento de la red porque es un único ente controlando la totalidad de red o una porción de la misma. Protocolos como OPENFLOW permiten al plano de control manipular el enrutamiento de tráfico a través de la red realizando mejores decisiones a la hora de calcular rutas y de gestionar de la red.

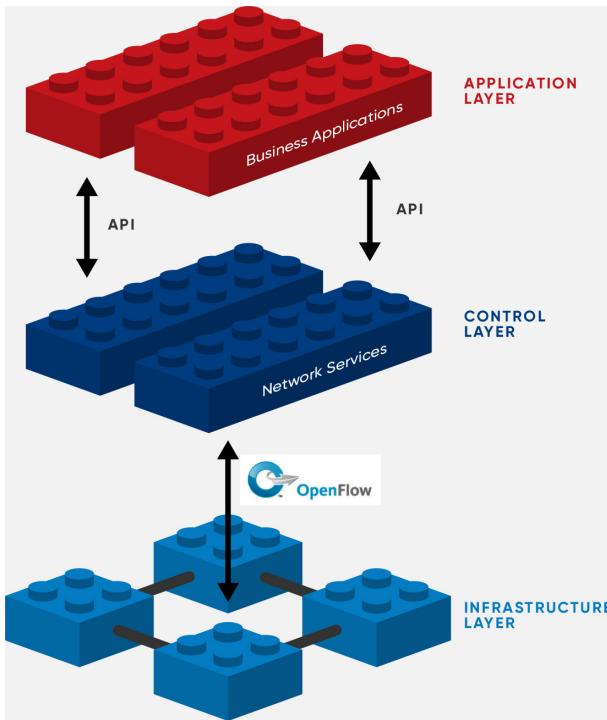


Figura 3. Arquitectura SDN. Fuente: *Open Networking Foundation*

SDN define una arquitectura formada por tres capas distintas como se muestra en la figura 5:

Capa de infraestructura

Se ubica el equipo físico de transferencia (*routers*, *switches*, etc).

Capa de control

Se encuentran los **controladores** SDN y aplicaciones escritas por programadores que interactúan con el controlador utilizando lo que se conoce como la interfaz *Northbound*. Para interactuar con los dispositivos de la red, el controlador usa la interfase *Softbound*. El controlador envía mensajes a los dispositivos usando varios protocolos – Uno de ellos es OPENFLOW - en la interfase *Softbound*. A diferencia de un *switch* o *router* tradicional, en un ambiente SDN puro el cerebro del dispositivo de red es removido del mismo y se coloca en la capa de control en un dispositivo por separado.

Capa de aplicación

Las aplicaciones en esta capa manipulan el flujo de tráfico a través de la red. SDN permite control sobre toda

la red desde un solo punto lo cual simplifica el diseño y el mantenimiento de la red. También reduce el trabajo entre dispositivos de red pues solo necesitan recibir instrucciones de la capa de control en vez de entender y procesar distintos protocolos.

El trabajo de operadores de red también se ve reducido a solo mantener y configurar la capa de control, lo cual les da tiempo para aprovechar en la mejora de la red al implementar inteligencia para alterar el comportamiento de la misma de forma dinámica y responder a las necesidades de clientes.

La arquitectura SDN soporta un conjunto de interfaces de programación (API) para incluir servicios como ruteo, seguridad, control de acceso, manejo de rutas, distribución de cargas, entre otros. SDN promueve el uso de estas APIs que pueden ser de código abierto, dejando de lado el soporte de proveedores. Si por ejemplo se quisiera implementar un nuevo algoritmo de enrutamiento y ponerlo a disposición en los *routers* de un cierto fabricante, este podría negarse a incluirlo. De la misma forma en la que el *Apple Store* o *Google Play* permiten a los programadores de aplicaciones a escribir sus propios programas al proveer interfaces de programación de alto nivel para tener acceso a los recursos de los teléfonos, una de las ideas de SDN es que un programador pueda escribir su propio programa y por medio de la interfaz *Northbound* se pueda controlar el flujo del tráfico de la red. El controlador abstrae la complejidad detrás de la programación de la red y la interfaz abierta en los dispositivos de red permite que el programador controle, por medio del programa que escribió, el flujo del tráfico. Se podría utilizar un lenguaje de alto nivel como Python para programar los circuitos de los *switches* sin tener que enviar mensajes de bajo nivel a dispositivos individuales en la red y cambiarle la configuración. La complejidad de la red se le oculta al programador: se puede escribir una aplicación de alto nivel que manipule el flujo de los datos a través de la red utilizando un protocolo como OPENFLOW pero, la complejidad de OPENFLOW y la complejidad de los dispositivos de la red están ocultos o abstraídos. De esta forma el programador solamente tiene que tener conocimiento una interfaz de alto nivel para escribir un programa manipule el envío de tráfico de un punto *a* a un punto *b*.

3.2. Características Fundamentales de SDN

3.2.1. Separación de planos

La primer característica fundamental de SDN es la separación de los planos de control y datos. Funcionalidades de reenvío (*forwarding*) incluyendo las tablas lógicas y las tablas para escoger como lidar con los paquetes entrantes, basados en características tales como una dirección MAC, dirección IP o una VLAN ID, reside en el plano de datos. Las acciones fundamentales realizadas por el plano de datos pueden ser descritas de acuerdo cómo manejan los paquetes que llegan. Se puede hacer *forward*, *drop*, *consume* o *replicate* un paquete entrante. También se puede transformar el paquete de alguna forma antes de tomar una acción posterior.

La lógica y los algoritmos que son usado para programar el plano de datos reside en el control de datos. Muchos de estos protocolos y algoritmos requieren conocimiento global de la red. El plano de control determina cómo las

tablas de reenvío (*forwarding tables*) y la lógica en el plano de datos debería ser programada o configurada. En una red tradicional cada dispositivo tiene su propio plano de control, la tarea primaria de ese plano de control es ejecutar protocolos de ruteo y *switching* de tal forma que todas las tablas de reenvío en los dispositivos en toda la red están sincronizadas.

3.2.2. Dispositivos simples y control centralizado

Los dispositivos son ahora controlados por un sistema centralizado ejecutando software de administración y control. En lugar de cientos o miles de líneas de complicado software de plano de control ejecutándose en el dispositivo y que le permite comportarse de forma autónoma, ese software es removido del dispositivo y puesto en un controlador centralizado. Este controlador basado en software puede entonces gestionar la red. El controlador provee instrucciones primitivas a los dispositivos simplificados cuando sea apropiado con el fin de permitirles hacer decisiones más rápidas acerca de cómo manejan los paquetes entrantes.

3.2.3. Automatización y virtualización de redes

El controlador basado en software en SDN provee una interfaz abierta en el mismo para permitir el control automático de la red. En el contexto de Open SDN¹ los términos *northbound* y *southbound* se usan frecuentemente para distinguir si la interfaz es para las aplicaciones o para los dispositivos. Estos términos se derivan del hecho que en la mayoría de los diagramas las aplicaciones se muestra arriba (al norte) del controlador mientras que los dispositivos se muestra en la parte de abajo (al sur) del controlador. El controlador ofrece una *northbound API* que permite que se pueda proporcionar algoritmos y protocolos que pueden hacer que la red trabaje eficientemente. La *northbound API* del controlador pretende brindar una abstracción de los dispositivos de red y la topología, de esta forma, las aplicaciones pueden ser desarrolladas para que trabajen sobre una amplia variedad equipos/fabricantes los cuales pueden diferir substancialmente en los detalles de su implementación.

Uno de los resultados de este nivel de abstracción es proveer la habilidad de virtualizar la red, desacoplar el servicio de la red de la red física subyacente. Esos servicios están aún presentes en los dispositivos *host* de tal forma que esos *hosts* no están concientes que los recursos de la red que están usando son virtuales o no los físicos para los cuales fueron diseñados originalmente.

3.2.4. Apertura

Una característica de las redes SDN basadas en OPENFLOW es que las interfaces deberían de permanecer estándar, bien documentadas y no ser propietarias. Las APIs que son definidas deberían dar al software el suficiente control de varios planos de control. La premisa es que al mantener las interfaces *northbound* y *southbound* abiertas en el controlador SDN se permitirá la investigación de nuevos e innovadores métodos para operar una red.

3.3. Operativa de SDN

A nivel conceptual, el comportamiento y operación de una red SDN es sencillo. En la figura 4 se proporciona una representación gráfica de la operación de los componentes básicos de SDN: los dispositivos SDN, el controlador y las aplicaciones. La forma más fácil de entender la operación es mirándola de abajo hacia arriba, iniciando por los dispositivos. Los dispositivos SDN contienen funcionalidad de reenvío (*forwarding*) para decidir qué hacer con cada paquete entrante. Los dispositivos también contienen los datos para tomar esas decisiones de reenvío. Los datos como tal son representados por los flujos definidos por el controlador, como se muestra en la parte superior izquierda de cada dispositivo.

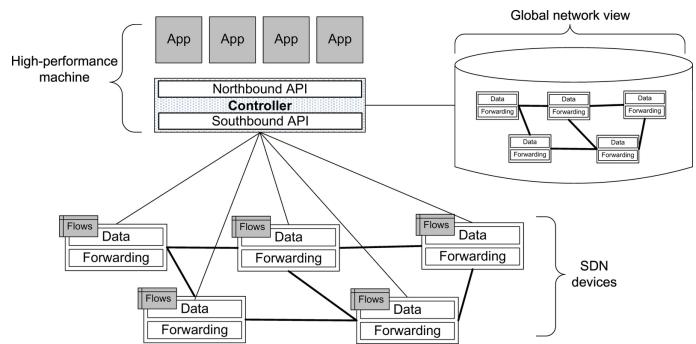


Figura 4. Visión general de la operativa SDN. Fuente: Open Networking Foundation

Un flujo describe un conjunto de paquetes transferidos desde un punto final de la red (*network endpoint*) hacia otro punto final de la red (o conjunto de puntos). Un conjunto de reglas describe las acciones que el dispositivo debería tomar para todos los paquetes que le pertenecen a ese flujo. Un flujo *flow* es unidireccional en que los paquetes que fluyen entre los mismos dos *endpoints*, en la dirección opuesta podrían constituir cada uno un flujo separado. Los flujos se representan en un dispositivo como una entrada de flujo (*flow entry*).

Una tabla de flujos reside en el dispositivo de red y consiste de una serie de entradas de flujo y las acciones a realizar cuando un paquete entrante coincide con un flujo. Cuando el dispositivo SDN recibe un paquete, consulta su tabla de flujos en busca de una coincidencia. Estas tablas de flujos se han construido previamente cuando el controlador descargó las reglas de flujo apropiadas al dispositivo. Si el dispositivo SDN encuentra una coincidencia entonces realiza la acción configurada apropiada que usualmente implica reenviar un paquete. Si no encuentra una coincidencia, el dispositivo puede soltar (*drop*) el paquete o pasarlo al controlador, dependiendo de la versión del protocolo y la configuración que tenga. La definición de un flujo es una expresión de programación relativamente simple de lo que lo que puede ser un cálculo de plano de control muy complejo realizado previamente por el controlador.

El controlador SDN es responsable de abstraer la red de los dispositivos SDN que controla y presentar una abstracción de estos recursos de red a las aplicaciones SDN ejecutándose por encima de sí. El controlador le permite a las aplicaciones SDN a definir flujos en los dispositivos

1. Decir algo de esto - OpenSDN

y ayuda a la aplicación a responder a paquetes que fueron reenviados al controlador por los dispositivos SDN. En la figura 4 se puede ver en la parte derecha del controlador que se mantiene una vista de la totalidad de la red que controla. Esto permite calcular soluciones de reenvío óptimas de una forma determinística y predecible. Debido a que un controlador puede control un gran número de dispositivos, estos cálculos son normalmente realizados en un equipo de alto rendimiento el cual posee mejor CPU y memoria que los que vienen típicamente en los dispositivos de red.

Las aplicaciones SDN están construidas por encima del controlador. Estas aplicaciones no deberían de confundirse con la capa de aplicación definida en el modelo OSI de siete capas para redes de computadoras. Las aplicaciones SDN se interconectan con el controlador usando un conjunto de flujos proactivos (*proactive*) en los dispositivos y para recibir paquetes que han sido enviados al controlador. Los flujos proactivos los establece la aplicación: típicamente la aplicación va a configurar estos flujos cuando la aplicación inicia. Los flujos van a persistir hasta que algún cambio en la configuración se haga. Esta clase de flujo se conoce como flujo estático. Otro tipo de flujo proactivo es cuando el controlador decide modificar un flujo basado en la carga del tráfico que se está dando actualmente a través del dispositivo de red.

Además de los flujos definidos proactivamente por la aplicación, algunos flujos son definidos para responder a un paquete enviado al controlador. Al recibir los paquetes entrantes que han sido reenviados al controlador, la aplicación SDN dará instrucciones al controlador para darle a conocer cómo responder al paquete y, si es apropiado, establecerá nuevos flujos en el dispositivo con el fin de permitir que el dispositivo responda localmente la próxima vez que vea un paquete que le pertenezca a ese flujo. Tales flujos se llaman flujos reactivos. De esta forma, ahora es posible escribir aplicaciones de software que implementan envío, ruteo, superposición (*overlay*), *multipath*, funciones de control de acceso, entre otros.

3.3.1. Dispositivos SDN

Un dispositivo SDN está compuesto por un API para comunicación con el controlador, una capa de abstracción y una función de procesamiento de paquetes. En el caso de un *switch* físico, la función de procesamiento de paquetes está contenida en el hardware para procesamiento lógico de paquetes

La capa de abstracción incorpora una o más tablas de flujos. La lógica de procesamiento de paquetes consiste de mecanismos para tomar acciones basado en los resultados de la evaluación los paquetes entrantes y de buscar la coincidencia de prioridad más alta. Cuando una coincidencia se encuentra, los paquetes entrantes son procesados localmente, a menos que explícitamente se usa una regla para enviar al controlador. Cuando no se encuentra una coincidencia, el paquete puede ser copiado al controlador para procesamiento posterior. En el caso de un *switch* de hardware, estos mecanismos son implementados por hardware especializado. En el caso de un *switch* de software, estas mismas funciones son hechas por software.

3.3.1.1. Tablas de flujos: son estructuras de datos fundamentales en un dispositivo SDN. Estas tablas de flujos

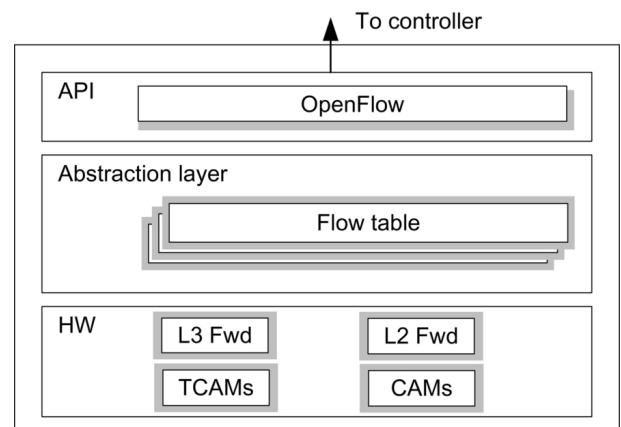


Figura 5. Anatomía del hardware un *switch* SDN. Fuente: *Open Networking Foundation*

permiten al dispositivo evaluar paquetes entrantes y llevar a cabo una acción apropiada basada en los contenidos del paquete que se acaba de recibir. Los paquetes han sido tradicionalmente recibidos por dispositivos de red y evaluados en función de ciertos campos. Dependiendo de la evaluación, se realiza una acción. Las acciones que se pueden tomar fueron anteriormente comentadas en la sección 3.3. Un dispositivo SDN no es fundamentalmente diferente, excepto que esta operación básica se ha vuelto más genérica y más programable a través de las tablas de flujo y su lógica asociada.

Las tablas de flujos consisten de un número de entradas de flujos con prioridades asociados, cada de los cuales consiste típicamente de dos componentes: *match fields* y *match actions*. Los *match fields* son usados para compararlo contra los paquetes entrantes. Un paquete entrada se compara contra un *match field* en orden prioritario y la primera coincidencia completa se selecciona. Las acciones son las instrucciones que el dispositivo de red debería de realizar si un paquete entrante coincide con los *match fields* especificados para una entrada de flujo.

3.3.2. Controlador SDN

3.3.3. Aplicaciones SDN

4. EL PROTOCOLO OPENFLOW

OPENFLOW es el primer protocolo estándar que trabaja entre la capa de control. Permite el acceso directo y el manejo de los dispositivos físicos o virtuales. El protocolo especifica primitivas básicas que pueden ser usadas por aplicaciones externas para programar los dispositivos de red. Por ejemplo, un programa que se cree para controlar la carga de red puede implementar OpenFlow para crear la comunicación entre los dispositivos de la red y el software, similar a las instrucciones de CPU que interactúan entre el sistema operativo y el hardware.

El protocolo permite también definir como el tráfico de la red en base a parámetros como patrones de uso, aplicaciones y recursos de la nube. Provee un control granular que permite a la red responder a cambios en tiempo real a nivel de aplicación, usuario y sesión.

Actualmente es el único protocolo de SDN estandarizado que permite la manipulación de la red.

Parte de los beneficios de usar este protocolo está la centralización del control de ambientes de diferentes proveedores de dispositivos físicos/virtuales, reducir la complejidad del mantenimiento de la red mediante la automatización de procesos, innovación al permitir al programador crear aplicaciones según las necesidades del negocio, aumenta la seguridad de la red pues permite una configuración de alto nivel de políticas de seguridad, configuraciones dinámicas aplicables en pocas horas que reducen el “*time-to-market*” del negocio, control granular de la red al aplicar políticas o configuraciones según diferentes niveles de distribución y carga.

4.1. Otros protocolos e iniciativas

5. CONCLUSION

The conclusion goes here.

APÉNDICE A

PROOF OF THE FIRST ZONKLAR EQUATION

Appendix one text goes here.

APÉNDICE B

Appendix two text goes here.

ACKNOWLEDGMENTS

The authors would like to thank...

REFERENCIAS

- [1]
- [2] Open Networking Foundation. <https://www.opennetworking.org/sdn-definition/>
- [3] Open Networking Foundation. *Software-Defined Networking: The new Norm for Networks*. White Paper. Abril, 2012.
- [4] N. Feamster, J. Rexford, and E. Zegura. 2013. *The Road to SDN*. Queue 11, 12. December 2013.
- [5] P. Goransson and C. Black. *Software Defined Networks: A Comprehensive Approach*. Elsevier Science. 2016.



Priscilla Piedra es Ingeniera de Computación del Tecnológico de Costa Rica. Actualmente es estudiante del programa de Maestría en Ciencias de la Computación en la misma universidad. Sus principales intereses son: *cloud computing* y automatización.



Martín Flores es Ingeniero en Informática de la Universidad Nacional. Actualmente, realiza sus estudios de Maestría en Ciencias de la Computación del Tecnológico de Costa Rica. Sus principales intereses son: lenguajes de programación, ingeniería de software y *DevOps*.