Redes Definidas por Software

Software Defined Networks

Gastón Alejandro Borja Perazzi, Ing. Giovanardi Ezequiel, Ing. Galleguillo Juan, Ing. Marcela Busnardo.
gborja333@alumnos.iua.edu.ar
Depto. de Electrónica y Telecomunicaciones
Instituto Universitario Aeronáutico – Facultad de ingeniería
Córdoba, Argentina

***Resumen*—En este artículo se pretende informar sobre esta arquitectura de red emergente, sus ventajas sobre la red tradicional y componentes básicos. También se explicará el protocolo OPEN FLOW, un protocolo emergente y abierto que facilita la implementación de SDN y los Switch OpenFlow. Por último se dará una breve explicación acerca de MININET, un entorno de red virtual que consta de un conjunto de hosts, switch, routers, links y controladores (que son el cerebro de esta arquitectura) para crear redes en máquinas virtuales sobre un simple kernel Linux para terminar de entender todo el concepto de Software Defined Networks.**

***Palabras Clave*—SDN, Openflow, Mininet, Switch Openflow, Controlador.**

I. INTRODUCCIÓN

Las redes de comunicación actuales enfrentan problemas que se van volviendo más críticos con el paso de los años, y hace tiempo se está buscando soluciones que se adapten tanto a las necesidades de los clientes como a las de las grandes empresas. Dentro de estos problemas podemos encontrar la gran cantidad de datos que se necesitan enviar por la red, la necesidad de seguridad avanzada, mayor eficiencia en el procesado de paquetes para no generar cuellos de botellas en los enrutadores ni colisiones, propiedades de escalabilidad acordes al crecimiento de la población y más incógnitas para las cuales las redes actuales resultan obsoletas [5].

Aquí es donde entran en juego las Redes Definidas por Software (SDN) con respuestas a todas las interrogantes que presentan las redes actuales, brindando un cambio de paradigma con respecto, no solo a las tecnologías nuevas a utilizar, sino también, como se deben aplicar.

II. REDES DEFINIDAS POR SOFTWARE

Esta nueva arquitectura de red presenta una característica clara que es la separación de las funciones de control y gestión de las funciones de "forwarding", dando como resultado una red dinámica, manejable, centralizada, adaptable y rentable, ideal para las necesidades de las empresas y clientes de hoy en día.

Al entender este concepto de la separación del plano de control y el plano de datos, podemos imaginar a la red como un gran conjunto de dispositivos de conmutación (switch, routers, etc), conectados entre sí por líneas de transmisión y, separado de estos, la capa de control por encima compuesta por un controlador que es el "cerebro" de la red y toma las decisiones sobre los paquetes que circulan por ella.

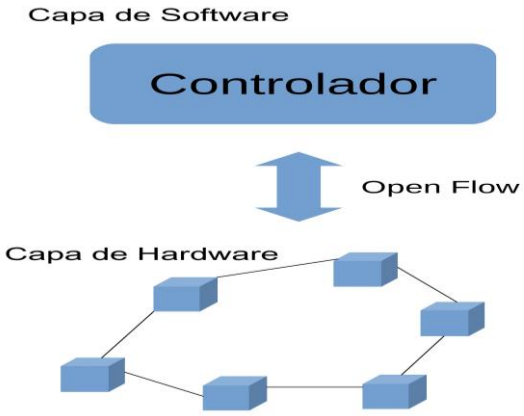

Figura 1: Visión básica de las SDN.

Dentro de las características que presenta esta nueva arquitectura y que sacan ventaja a las redes convencionales se encuentran [1]:

- **Gestión centralizada:** Permite gestionar la totalidad de la red desde un único punto, eliminando fallas de seguridad, cuellos de botellas, errores de configuración, etc. que traen consigo los múltiples puntos de decisión.
- **Agilidad:** Se puede reservar ancho de banda para servicios especiales o QoS con mayor rapidez y adaptarse a los cambios y prioridades de la red.
- **Bajos costos:** Al no tener que optar por la homogeneidad de productos y tener un proveedor genérico, se abaratan los costos de la red.
- **Programación automática:** SDN permite a los administradores, por medio de programas automatizados que ellos mismos pueden escribir ya que no depende de software propietario, gestionar y

optimizar los recursos de la red de forma rápida y dinámica.

- **Fácilmente programable**: Brinda la posibilidad de programar o configurar el plano de control con mayor facilidad al estar separado del plano de red.
- **Escalabilidad:** Con esta arquitectura, es muy fácil proyectar una red que crezca en gran medida sin perder ninguna de sus características.
- **Basado en estándares abiertos y proveedores neutrales:** Como se implementa a través de estándares abiertos, simplifica el diseño de la red y las operaciones porque las instrucciones son proporcionadas por el controlador, en lugar de múltiples dispositivos, proveedores específicos o protocolos.
- **Seguridad:** Al estar la red centralizada por el controlador, no se corre el riesgo de poseer agujeros de seguridad en las configuraciones de los switchs.

Todas estas características que poseen las SDN y no las redes tradicionales nos llevan a pensar que en un futuro, no muy lejano, esta arquitectura será ampliamente aceptada por todos los sectores del mercado de telecomunicaciones, favoreciendo tanto a los clientes que buscan velocidad, seguridad, y robustez en sus conexiones, como a las empresas que buscan brindar un servicio de alta calidad y confiabilidad con el menor costo de operación y de inversión posible.

III. Arquitectura lógica de las SDN

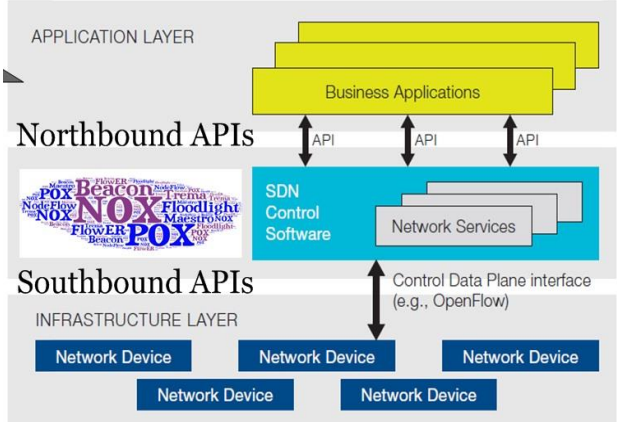

Figura 2: Arquitectura lógica SDN [12]

Examinando las redes definidas por software en profundidad, podemos separarlas en tres capas bien diferenciadas. La capa inferior o capa de hardware, compuesta por el conjunto de switchs, routers, hosts, medios de transmisión, etc, que pueda contener el sistema. La capa del medio recibe el nombre de capa de control, compuesta por el controlador SDN, que se encarga de las decisiones a tomar sobre los paquetes en la red y como deben responder los dispositivos de la capa de hardware. En esta capa se encuentra el sistema operativo de red (NOS) y posee una visión global de la red. Se comunica con la capa inferior por medio de las SouthBound APIs, que permite el diálogo entre el controlador y los dispositivos de red. Entre estas APIs se encuentra OPENFLOW.

La capa superior o capa de aplicaciones está compuesta por las aplicaciones de usuario y se comunica con el controlador por medio de las NorthBound APIs. Esta parte es muy importante y es la que saca ventaja sobre las redes tradicionales. Estas APIs incorporan los patrones de uso de la red, esto quiere decir que, la red actuará de diferente manera de acuerdo a la aplicación que se necesite. Entonces la capa de aplicación comunica los patrones de uso de la red de una aplicación por medio de las NorthBound APIs al controlador, este toma la decisión más inteligente y se comunica con la capa inferior por medio de Openflow para indicar que hacer con los paquetes [8].

IV. Openflow

Es una tecnología de switching y se define como "un protocolo emergente y abierto de comunicaciones que permite a un servidor de software determinar el camino de reenvió de paquetes que debería seguir una red de switches"[8]. Su propósito es ser el comunicador entre el controlador y los distintos dispositivos de la red, logrando que la red se vea como un todo y sea el controlador el que tome las decisiones sobre los paquetes que viajan en ella, dejándole a los dispositivos de encaminamiento la función de reenvío solamente. Los switch convencionales reciben los paquetes, buscan coincidencias en sus tablas de flujos, y realizan la acción preestablecida. La diferencia con los switch OpenFlow radica en estas tablas, los switch convencionales poseen una tabla de flujo creada por el proveedor de dicho instrumento, lo que nos ata a los protocolos y decisiones que se hayan preestablecido sin poder realizar cambios. Por el contrario, con OpenFlow, los usuarios pueden configurar las tablas de flujo a su antojo y decidir cómo serán tratados los paquetes según las necesidades de cada red y usuario. Si algún paquete no encuentra coincidencia en su tabla de flujo, se enviará al controlador que lo procesará y devolverá al switch, agregando una nueva entrada de flujo a su tabla, creando una red dinámica, ágil, centralizada y automatizada [9].

Este protocolo, como ya dijimos, define la comunicación entre el switch Openflow y el controlador. El switch establece una comunicación con el controlador, usando un determinado puerto, normalmente el 6634, e iniciará una comunicación TCP estándar. Cuando se establece la comunicaciones, cada lado envía un mensaje de HELLO junto con la versión de OpenFlow más alta que puede soportar, y así se negocia la versión mínima entre la que se envió y la que se recibió en los dispositivos. Si se soporta la versión negociada, la conexión será iniciada, sino se enviará un mensaje de HELLO-FAILED y la conexión será finalizada [7].

Podemos encontrar tres mensajes distintos en el protocolo OpenFlow [8]:

- **Mensaje del controlador al Switch:** Es iniciado por el controlador y su objetivo es conocer y/o actuar sobre el estado actual del switch. Dentro de este mensaje podemos encontrar cuatro tipos distintos:
    - **Modificar estado:** Añade, modifica o elimina entradas en tablas de flujo y fija características en los puertos.
    - **Lectura de estado:** Sirve para consultar al switch sobre las estadísticas del tráfico (se usa la parte de las estadísticas de las entradas de flujo).
    - **Enviar paquete:** Indica que envié paquetes por un puerto específico luego de agregar una nueva entrada de flujo a la tabla del switch.
    - **Notificaciones:** El controlador lo utiliza para asegurarse de que los objetivos de un mensaje se han cumplido o para recibir notificaciones por operaciones finalizadas.
- **Mensajes asíncronos:** Son mensajes enviados por los switch hacia el controlador. Existe cuatro tipos de mensajes distintos:
    - **Entrada de paquete:** Se envía al controlador cuando algún paquete no encuentra coincidencia con alguna entrada de la tabla de flujo del switch o la acción asociada con esa entrada es reenviar el paquete al controlador.
    - **Modificación de flujo:** Se envía cuando se ha agregado, eliminado o cambiado con éxito una entrada en la tabla de flujo del switch.
    - **Estado de puertos:** Se envía cuando se modifica el estado de algún puerto del switch.
    - **Error:** El switch informa al controlador si existen problemas con estos mensajes.
- **Mensajes síncronos:** Son enviados en cualquier dirección sin solicitud previa. Existen tres tipos distintos:
    - **Hello:** Mensaje intercambiado durante la duración de la negociación de conexión.
    - **Echo:** Es utilizado por cualquiera de los dos dispositivos para verificar el ancho de banda, la latencia, o, simplemente, la conexión entre los dos dispositivos. Una petición Echo request es respondida por un mensaje Echo reply desde el destino al origen.
    - **Mensaje de proveedor:** Esta planeado para futuras versiones de OpenFlow y consta de mensajes para ofrecer funcionalidades adicionales a los switch.

Está claro que el protocolo OpenFlow está totalmente pensado para la implementación de las SDN y se presenta en la actualidad como el primer estándar desarrollado para las redes definidas por software. A pesar de sus características, la Open Networking Fundation, organización desarrolladora y promotora de Software Defined Networks, sigue trabajando para poder agregarle más beneficios para mejorar la interacción entre el controlador y los Switch OpenFlow.

V. CONTROLADOR SDN

Como ya se vió más arriba, el controlador es el núcleo de estas redes, es la parte más importante, el encargado de la "inteligencia" de la red. Él es quien toma las decisiones como la distribución de recursos, decisiones sobre paquetes que no coinciden con las tablas de flujo de los switchs, creación, modificación, o eliminación de entradas de flujo, ejecuta las instrucciones que le proporcionan las distintas aplicaciones, básicamente, centraliza el funcionamiento de la red que es la característica básica de las SDN. Según el tipo de red en el que se encuentre, se lo puede clasificar en distintos modos de funcionamiento [8]:

- **Emplazamiento:** Se separa en dos configuraciones, una es centralizada, donde un solo controlador maneja a toda la red y otra en donde hay un controlador para un conjunto de dispositivos de una red.
- **Por flujo:** Donde cada entrada define un flujo en particular o donde una entrada define un conjunto de flujos.
- **Comportamiento:** Puede ser reactivo, donde el controlador va a agregar una nueva entrada de flujo cuando llegue un paquete al switch que no tenga coincidencia con ninguna entrada, o proactivo, donde el controlador completa la tabla de flujo antes de que lleguen flujos nuevos. Cuando es reactivo, cada paquete nuevo que llegue al switch va a tener un tiempo de retardo que se corresponde a su procesamiento en el controlador y a la implementación de una nueva entrada de flujo, y si se pierde la conexión entre ambos se corta, el switch no podrá conmutar paquetes de flujos nuevos, limitando su conectividad. En el modo proactivo, no hay retardo y no se pierde conectividad si se corta la comunicación entre el controlador y el switch.

Un controlador puede ser, desde un simple software en una computadora, encargado de gestionar las tablas de flujos de los dispositivos de una red, quedando en mano de un solo administrador encargado de la gestión, hasta múltiples administradores de red que pueden gestionarla por medio de distintas cuentas y con la posibilidad de poder configurar distintos conjuntos de flujos de la red.

Dentro de los principales controladores OpenFlow que hay hoy en día están [8]:

- **NOX/POX:** Un controlador open-source que brinda una plataforma para escribir software de control de la red en C++ o Phyton.
- **Floodlight:** Controlador Open-Source basado en Java.
- **Beacon:** Controlador basado en JAVA.
- **Trema:** Completa plataforma OpenFlow para Ruby/C.
- **Maestro:** Plataforma de control escalable escrita en JAVA.
- **SNAC:** Controlador que usa una interfaz web para controlar la red.
- **IRIS:** Controlador basado en JAVA.

VI. Switch OpenFLow

Estos dispositivos cuentan con tres partes:

- Una o más tablas de flujo que indican como debe ser procesado cada paquete que entre en el switch.
- Un canal especial o seguro que comunique el switch con el controlador. Esta comunicación se realiza por medio del protocolo OpenFlow.
- Un controlador que sea quien gestione las tablas de flujo del switch de acuerdo a las necesidades del usuario.

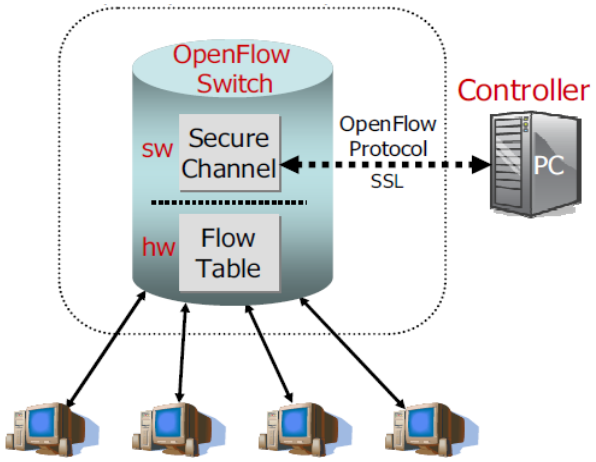

Figura 3: Arquitectura Switch OpenFlow[11]

Cada switch puede poseer una o más tablas de flujo utilizadas para decidir como redireccionar los flujos entrantes. Para eso, cada una de las entradas en las tablas de flujo poseen tres campos [9]:

- **Cabecera:** Se encarga de definir de que flujo se trata.
- **Acciones:** Que se realizarán cuando el paquete entrante encuentre coincidencia con alguna entrada de flujo de la tabla.
- **Estadísticas:** Que monitorizan el paso de flujos, número de paquetes, número de bytes, etc.

Dentro de las acciones que se pueden realizar, podemos encontrar tres:

- Reenviar paquete a través de un puerto.
- Encapsular el paquete y enviarlo al controlador para decidir qué hacer con él. Esto normalmente sucede cuando llega un paquete de un flujo nuevo.
- Eliminar paquete.

Con todo esto, gracias al protocolo OpenFlow, el switch solo debe recibir el paquete, buscar alguna coincidencia en la tabla de flujo y realizar la acción indicada, si no hay coincidencia se envía el paquete al controlador para que este decida si agregar una nueva entrada a las tablas de flujo o eliminar el paquete. Tomando en cuenta esto, se puede observar que en el único lugar donde hay retardo es cuando llega un nuevo flujo que no encuentra coincidencia en la tabla y debe ser examinado por el controlador, luego de que se agregue una nueva entrada de flujo para ese paquete, los próximos que llegue del mismo tipo no deberán ser enviados al controlador puesto que ya se agregó una entrada para decidir qué hacer con ellos. Esto también nos agrega seguridad, ya que podemos configurar el controlador para que elimine cualquier paquete que se considere peligroso para la red.

VII. Mininet

Con el fin de promocionar el uso de SDN, aprender sus características, desarrollar nuevos protocolos, y favorecer su enseñanza a las nuevas generaciones, en la Universidad de Stanford se desarrolló Mininet, un emulador de red virtual, con un conjunto de switch, routers, links, controladores, host, etc. Corriendo en un simple núcleo Linux, brindando la posibilidad de acceder a cada dispositivo individualmente, cambiando sus características a nuestro gusto. También nos da la posibilidad de modificar los enlaces que unen estos dispositivos, cambiando su retardo, su ancho de banda, etc. Para poder tener una experiencia más real aun. Entre sus puntos fuertes se encuentran [2]:

- Es muy veloz, dado que crear una red virtual toma tan solo unos pocos segundos.
- Posibilidad de crear diferentes tipos de redes, desde dos host con un solo controlador, hasta centros de datos, o redes para una universidad o ciudad.
- Se puede combinar con cualquier programa que corra en Linux para sacarle más provecho, por ejemplo, podemos monitorizar los paquetes OpenFlow con Wireshark.

- Permite modificar a gusto las tablas de flujo de los switchs de la red virtual, adaptándola a nuestras necesidades. Característica básica de las SDN.
- Posibilidad de correr Mininet en una computadora de escritorio, Notebook, máquina virtual, etc.
- Su uso es muy fácil y se pueden crear distintos tipos de topologías mediante scrips en Phyton.
- Es un proyecto Open-Source, por lo cual, cualquiera puede acceder a su código y modificarlo, arreglar errores, agregar características, etc.

A pesar de sus buenas características, no es perfecto y tiene sus limitaciones:

- Como cualquier virtualización, comparte recursos con la máquina que la aloja.
- Los host creados no pueden acceder a internet ya que no se puede hacer NAT al exterior.

Es una herramienta potente para la enseñanza, investigación y desarrollo de las SDN.

Su uso es bastante básico, con solo configurarla y poner a correr la máquina virtual, se pueden crear redes virtuales. Desde una red básica con el comando **sudo mn** o **sudo mn --topo minimal**, que consta de un controlador, un switch y dos hosts [3].

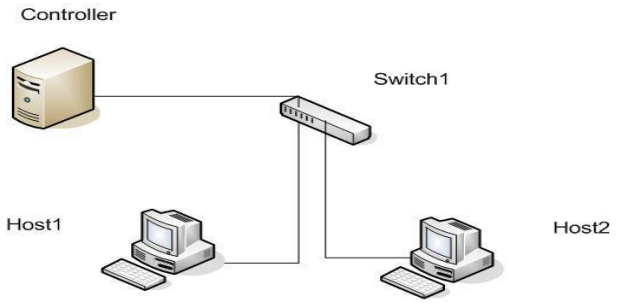

Figura 4: Topología mínima [8]

Hasta una red de las dimensiones que queramos con los parámetros de host, switch, controlador y links que queramos.

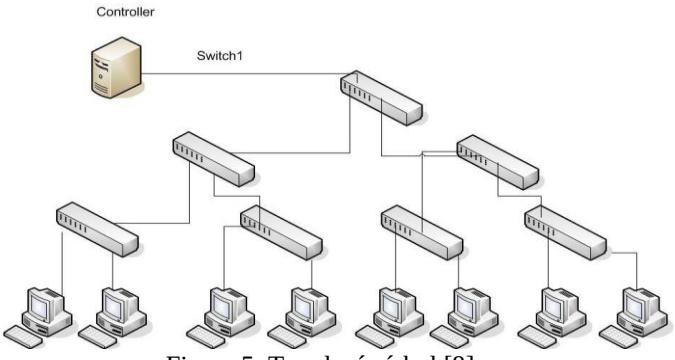

Figura 5: Topología árbol [8]

Para demostrar las cualidades básicas de esta arquitectura de red, se utilizó Mininet como software de simulación de redes y el controlador Floodlight[13], que es un controlador Open Source basado en java.

Lo Primero es crear la red virtual con Mininet, en este caso se utilizara la topología tipo árbol (tree) con tres niveles.

Esto se logra escribiendo en el termina el comando:

- sudo mn --topo=tree,3 --link tc,bw=100 --controller=remote,port=6653

Con esto se crea una red tipo árbol, con tres niveles, los enlaces tienen una velocidad de 100Mbits, y con un controlador remoto al cual se accede por el puerto 6653.

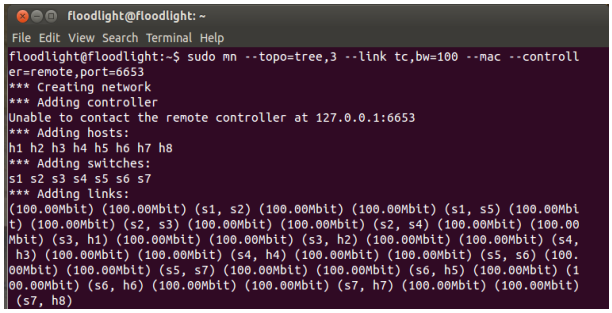

Figura 6: Creacion de la red.

Como no tiene un controlador que le indique que hacer con los paquetes que circulan por la red, si yo intento realizar un ping desde el host número 1 al host número 8, no se podrá realizar. Esto se hace con el comando:

- h1 ping –c4 h8

Con esto envió cuatro paquetes ICMP de 64 bytes del host 1 al host 8 esperando respuesta a cada uno de ellos, pero como no hay controlador estos paquetes nunca llegan a destino.

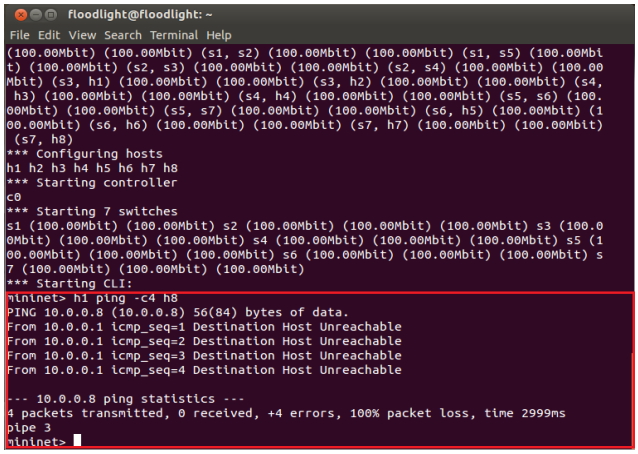

Figura 7: Ping rechazado.

Ahora conectamos el controlador floodlight que armara una tabla de flujo en cada uno de los switchs indicando que tienen que hacer con los paquetes que reciben. Esto se logra ingresando en otro terminal el comando:

- java –jar taget/floodlight.jar

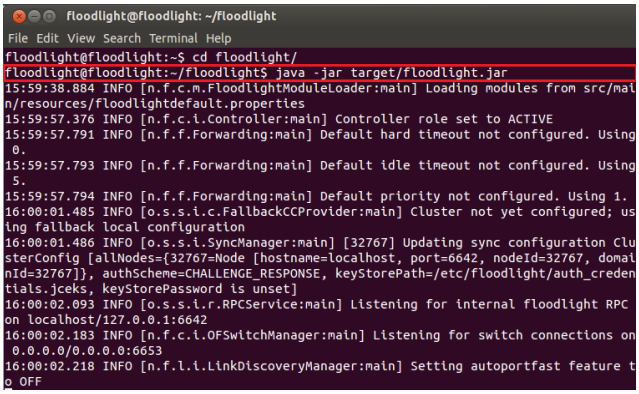

Figura 8: Ejecucion del controlador.

Posee una interfaz gráfica muy útil para ver la topología de la red e información sobre los hosts y los switchs.

Ahora si probamos realizar nuevamente los cuatro ping del host 1 al host 8 veremos que la operación se realiza con éxito y además podemos apreciar el retardo que experimenta el primer paquete con respecto a los siguientes.

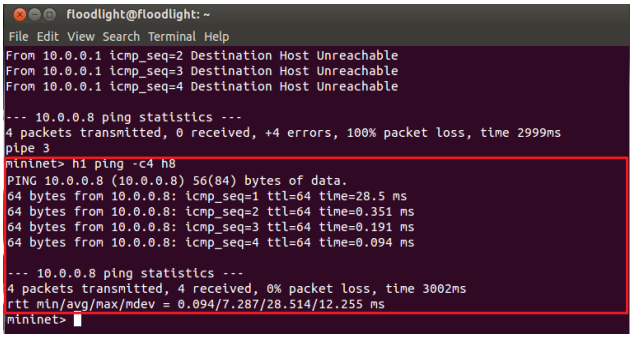

Figura 9: Ping con el controlador.

También, como se dijo antes, se puede utilizar el software Wireshark para ver la comunicaciones entre el controlador y los switchs de la red.

Se pueden ver los paquetes Hello, negociación de versión, configuraciones, etc.

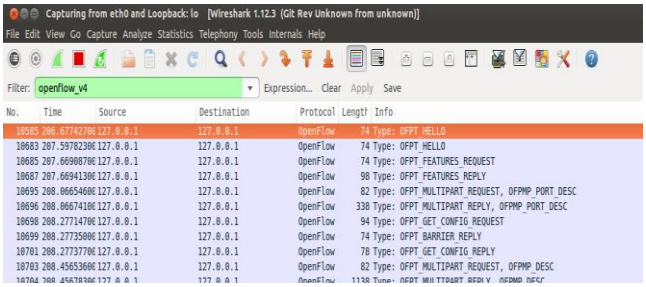

Figura 10: Captura de paquetes con Wireshark.

VIII. Conclusión

Este nuevo paradigma de las redes llamado SDN trajo consigo un nuevo futuro positivo para las telecomunicaciones. Con todas las ventajas antes mencionadas como programación facilitada, agilidad, gestión centralizada, bajos costos, escalabilidad, etc. Provee soluciones a problemas que son muy comunes en la actualidad y para los cuales no se encontraban respuestas.

Pero donde se encuentra su mayor utilidad y su principal característica, es en el hecho de poder crear protocolos personalizados y redes individuales para cada necesidad, con mayor seguridad, mejor calidad de servicio y mayor convergencia de utilidades, pudiendo asignar anchos de banda específicos para cada recurso que se utilice en la misma red de forma dinámica.

También hemos visto el entorno Mininet, basado en un kernel de Linux que es muy didáctico y de fácil entendimiento. Es una excelente opción para la enseñanza de este nuevo protocolo en carreras relacionadas a este tema. Con una base simple de programación y de las funciones de un switch tradicional, se puede lograr que el alumno entienda el funcionamiento de este protocolo, y junto con el programa Wireshark se puede estudiar todos sus mensajes en detalle, obteniendo una base sólida para el futuro.

REFERENCIAS

[1] Open Networking Fundation. Disponible en: https://www.opennetworking.org

[2] Mininet. Disponible en : http://www.mininet.org

[3] Guthub mininet. Disponible en : https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

[4] James F. Kurose & Keith W. Ross, Redes de computadoras, un enfoque descendente, 5° ed., Pearson, 2010.

[5] Joseph D. Sloan, Network Troubleshooting Tools, 1° ed., O'Reilly, 2001.

[6] William Stallings, Comunicaciones y redes de computadores, 7° ed., Person-Prentice Hall, 2004.

[7] Thomas D. Nadeau & Ken Gray, SDN Software Defined Network, 1° ed., O'Reilly, 2013.

[8] Óscar Roncero Hervás, Tesis "Software Defined Network", Universidad Politécnica de Catalunia, España. Disponible en: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf

[9] Wikipedia, OpenFlow. Disponible en: http://es.wikipedia.org/wiki/Openflow

[10] Wikipedia, Redes definidas por software. Disponible en: http://es.wikipedia.org/wiki/Redes_definidas_por_software

[11] Departamento de ciencias de la computacion de Stanford. Disponible en: http://yuba.stanford.edu/cs244wiki/index.php/Overview.

[12] Principia Technologica. Disponible en: http://principiatechnologica.com/2014/03/25/investigacion-de-redes-sdn-parte-iii-la-arquitectura-sdn/

[13]Floodlight, Disponible en: http://www.projectfloodlight.org/floodlight/