UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS

ELECTRÓNICA E INDUSTRIAL

Carrera de Ingeniería en Electrónica y Comunicaciones

TEMA:

"RED DEFINIDA POR SOFTWARE (SDN) EN BASE A UNA INFRAESTRUCTURA DE SOFTWARE DE LIBRE DISTRIBUCIÓN"

Proyecto de Trabajo de Graduación. Modalidad: TEMI. Trabajo Estructurado de Manera Independiente, presentado previo la obtención del título de Ingeniero en Electrónica y Comunicaciones.

SUBLÍNEA DE INVESTIGACIÓN: Teoría, diseño e interconexión de redes

AUTOR: Alex Vladimir Núñez Ramires

PROFESOR REVISOR: Ing. Mg. Santiago Manzano

Ambato - Ecuador

Abril 2015

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de investigación sobre el tema: "RED DEFINIDA POR SOFTWARE (SDN) EN BASE A UNA INFRAESTRUCTURA DE SOFTWARE DE LIBRE DISTRIBUCIÓN" del señor Alex Vladimir Núñez Ramires, estudiante de la Carrera de Ingeniería Electrónica y Comunicaciones , de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe de investigación reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el Art. 16 del Capítulo II, del Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad Técnica de Ambato.

Ambato, Abril 2015

EL TUTOR

_____

Ing. Mg. Santiago Manzano

AUTORÍA

El presente trabajo de investigación titulado "RED DEFINIDA POR SOFTWARE (SDN) EN BASE A UNA INFRAESTRUCTURA DE SOFTWARE DE LIBRE DISTRIBUCIÓN" es absolutamente original, auténtico y personal en tal virtud, el contenido, efectos legales y académicas que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato Abril, 2015

_____

Alex Vladimir Núñez Ramires

CC: 1803942000

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes aprobó el Informe Final del trabajo de graduación titulado "RED DEFINIDA POR SOFTWARE (SDN) EN BASE A UNA INFRAESTRUCTURA DE SOFTWARE DE LIBRE DISTRIBUCIÓN" presentado por el señor Alex Vladimir Núñez Ramires de acuerdo al Art. 17 del Reglamento de Graduación para obtener el título Terminal de tercer nivel de la Universidad Técnica de Ambato.

Ing. Vicente Morales L., Mg.

_____

PRESIDENTE DEL TRIBUNAL

Ing. Mg Juan Pablo Pallo	Ing. Mg. Patricio Córdova
_____	_____
DOCENTE CALIFICADOR	DOCENTE CALIFICADOR

DEDICATORIA

"El amigo siempre es amigo, y en los tiempos difíciles es más que un hermano" Proverbios 17:17

A mi familia por estar siempre conmigo y brindarme su amor y apoyo incondicional. En especial a mi tía Zoila que más que una madre ha sabido, educarme, guiarme y ayudarme a cumplir mis metas con amor y sabiduría.

A mis padres Víctor y Margoth por el gran sacrificio que hicieron al separase de mi para brindarme un mejor porvenir.

A mi hermana Liliana que con sus palabras y su forma de ser me ha reconfortado y alegrado mi vida.

Alex

AGRADECIMIENTO

A Dios por cuidar y guiar mi camino en cada momento y rodearme de personas maravillosas.

A mi tía Zoila por quererme y guiarme como a un hijo.

A mis padres por su amor y apoyo incondicional.

A mi hermana Liliana por estar conmigo en los momentos difíciles e inspirarme con su valentía.

A mi novia Viviana por toda la ayuda, cariño y amor que me ha brindado.

A la Facultad de Ingeniería en Sistemas, Electrónica e Industrial por instruirme y brindarme los conocimientos necesarios para cumplir mis metas.

Al Ing. Santiago Manzano por brindarme su guía en el transcurso de este proyecto.

PAGINAS PRELIMINARES

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La presente investigación tiene como objetivo estudiar las redes definidas por software mediante el análisis de los diferentes aspectos que componen esta nueva arquitectura, llegando a diseñar e implementar un prototipo de red tanto con dispositivos diseñados para trabajar con el protocolo OpenFlow como con aquellos que pueden ser habilitados para este propósito.

El estudio de esta nueva tecnología se debe a la gran demanda de servicios de red con mejor calidad, siendo esta la causa de que la mayoría de las empresas de redes estén reestructurando sus arquitecturas con el objetivo de dar cabida a una nueva propuesta que permita solucionar los problemas generados sin afectar totalmente sus infraestructuras.

Gracias a que se trata de una tecnología de código abierto se espera un gran desarrollo de la misma, con el propósito de encontrar soluciones que permitan minimizar costos de implementación y dejar de lado la dependencia de servicios de un solo fabricante de dispositivos de red, solucionando así problemas de incompatibilidad.

ABSTRACT

The aim of this research is study the software defined networks through the analysis of different aspects that made up this new architecture, leading to design and implement a network prototype with both devices designed to work with the OpenFlow protocol and with those that can be authorized for this purpose.

The study of this new technology is due to the high demand for network services with better quality, this is the reason why most companies are restructuring their network architectures in order to accommodate a new approach to solving the problems generated without totally affect their infrastructure.

Because it is an open source technology we expect a great development thereof, in order to find solutions to minimize deployment costs and neglecting the services of only one manufacturer of network devices and troubleshooting incompatibility issues.

GLOSARIO DE TÉRMINOS

SND.- Nueva arquitectura de red denominada “Redes definidas por software”.

NAC.- (Network Access Control) aplicación diseñada para controlar el acceso a la red.

Open VSwitch.- Switch Virtual multicapa que se encuentraba bajo la licencia de Apache.

NOX.- Primer controlador OpenFlow desarrollado.

API.- Interfces de programación de aplicaciones

Plano de control.- Dictamina la manera de tranferir los datos.

Plano de datos.- Se encarga de hacer efectiva la tranferencia de los datos.

ACL.- Listas de control de acceso.

OpenFlow.- Protocolo que hace posible el acceso a las tablas de flujo de los dispositivos de red.

Controlador.- Aplicación de software que se encarga de gestionar la inteligencia de la red.

Mininet.- Simluador de redes SDN.

DPCTL.- Utilidad por línea de comandos útil para interacturar con los switches OpenFlow.

Beacon.- Controlador OpenFlow desarrollado por la universidad de Stanford.

Floodlight.- Controlador OpenFlow con base de programación en java.

Packet-in.- Mensaje Openflow enviado por el switch hacia el controlador cuando un paquete no puede ser procesado.

FlowMod.- Objeto de los controladores que permite interactuar con las entrads de flujo.

INTRODUCCIÓN

En el presente proyecto se realiza un estudio sobre la nueva tecnología de red emergente denominada Redes Definidas por Software (SDN), la cual propone nuevos beneficios y soluciones a muchos de los problemas presentes en la arquitectura de red actual como la demanda de ancho de banda, control de tráfico, servicios de red personalizados etc., Dotando a las redes de un mejor control y administración. El proyecto de investigación se encuentra estructurado de la siguiente manera:

En el capítulo I, detalla la problemática a tratar junto con la justificación de la importancia de las redes definidas por software.

En el capítulo II, desarrolla el marco teórico en el cual se describen los principales componentes de la arquitectura de Redes Definidas por Software, además se analizan las limitaciones que impulsaron al desarrollo de esta nueva arquitectura, los tipos de dispositivos que soportan esta nueva tecnología y las herramientas para el diseño y simulación de éste tipo de redes.

En el capítulo III, describe el proceso mediante el cual se recolecto la información necesaria para el desarrollo de la propuesta, descrita en el capítulo IV, comenzando por escoger el prototipo de red que se desarrollará, así como el análisis y elección del software controlador, la determinación de los equipos a utilizarse para la implementación del prototipo físico, la simulación y pruebas del prototipo de red.

Los resultados del proyecto se presentan en el capítulo V, a través de las conclusiones y recomendaciones.

CAPÍTULO I

EL PROBLEMA

1.1 Tema

Red definida por software (SDN) en base a una infraestructura de software de libre distribución

1.2 Planteamiento del Problema

Desde el surgimiento de la tecnología ha existido la necesidad de intercambiar información, y fue gracias a la llegada de la era de la informática y las comunicaciones que se desarrollaron distintos métodos para lograrlo, pasando desde sistemas rudimentarios hasta llegar a aquellos de gran complejidad y alcance. Debido a esto surgen las redes de computadoras en el año de 1969 con la creación de Arpanet [1].

Las redes han trabajado de manera eficaz hasta la actualidad, pero debido a que fueron construidas hace varios años ya no son capaces de satisfacer los requerimientos que se presentan hoy en día, como: la necesidad de implementar servicios de red personalizados, cambios en los patrones de tráfico de forma dinámica, flexibilidad, escalabilidad, acceso rápido y ágil a los servicios de la nube, etc. Esto se debe en gran parte a que el modelo de red actual es muy rígido y no permite realizar cambios de una manera sencilla [2].

Otro problema que se presenta en las redes actuales es que los protocolos que se utilizan están diseñados para solucionar problemas específicos, por lo cual se manejan de manera aislada y no permiten separar las aplicaciones de ciertos factores que se deben tomar en cuenta en él envió de paquetes, lo cual provoca una demora en el tráfico al atravesar dispositivos, actualizar las listas de control de acceso, realizar el enrutamiento, entre otros, sumándole complejidad a la red.

Es por ello que el desarrollo de nuevas alternativas para gestionar las redes es de vital importancia no solo en el Ecuador sino para el resto del mundo, puesto que, a medida que se incrementa el tráfico con la adición de nuevos dispositivos y requerimientos a la red será cada vez más difícil su gestión, ya que en el modelo de red actual no se da prioridad al tráfico entre servidores causando retrasos al momento de transmitir la información [3].

En las redes actuales tanto el plano de control como el plano de datos están situados en los dispositivos de la red, mientras que en las redes definidas por software (SDN) el plano de datos está separado del plano de control, por lo que la toma de decisiones ya no se realiza por el dispositivo sino por el administrador de la red.

En lo referente a la programación de la red existe una gran diferencia entre las redes actuales y las redes definidas por software (SDN), puesto que en las redes actuales no se puede programar la red mediante aplicaciones, y cada elemento de la misma debe ser configurado por separado, mientras que las redes definidas por software (SDN) la red se puede programar mediante aplicaciones y el controlador puede exponer interfaces de aplicación para la manipulación de la red.

Otro aspecto importante es que la inteligencia y el control en las redes definidas por software (SDN) se encuentran centralizados en el controlador SDN mientras que en las redes actuales se encuentran distribuidos por todos los elementos de la red [4].

1.3 Delimitación

Delimitación de Contenidos

ÁREA ACADÉMICA: Programación y Redes

LÍNEA DE INVESTIGACIÓN: Programación y Redes.

SUBLÍNEA DE INVESTIGACIÓN: Teoría, diseño e interconexión de redes.

Delimitación Espacial

El presente proyecto se realizó en la ciudad de Ambato

Delimitación Temporal

El proyecto se desarrolló en los seis meses posteriores a la aprobación del proyecto por parte del Honorable Consejo Directivo de la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato.

1.4 Justificación

El desarrollo de este proyecto es importante debido a que permitirá mejorar el rendimiento de las redes actuales, además se analizará conceptos nuevos, los cuales aportarán al entendimiento de un nuevo tipo de arquitectura. Con el estudio de las redes definidas por software se podrá solucionar las limitaciones que hoy en día presentan las redes de datos, como: la necesidad de configurar los dispositivos de la red de manera individual, la dependencia de los fabricantes, soporte para nuevos servicios y capacidades, etc. [2].

Beneficiándose principalmente empresas que manejan grandes cantidades de información tales como: proveedores de servicios de telecomunicaciones, entidades bancarias, empresas de desarrollo y venta de aplicaciones para dispositivos móviles etc., debido al gran potencial que poseen las redes definidas por software en cuanto se refiere al manejo del tráfico.

El desarrollo de este proyecto será de gran utilidad, puesto que las redes definidas por software (SDN) presentan grandes ventajas, y una de las más importantes es la seguridad, ya que en una red normal se debe configurar de forma manual cada dispositivo de la misma, siendo esta una tarea tediosa en la cual se podrían cometer fallas por las cuales las redes podrían ser atacadas, mientras que en las redes definidas por software (SDN) al poseer un controlador encargado de administrar toda la red la seguridad será mejor, ya que se centralizaría en un solo punto.

Realizar un proyecto de esta magnitud será de gran aporte y brindará una visión diferente al momento de construir redes, ya que con las redes definidas por software (SDN) ya no será el hardware sino el software el encargado de controlar la red, logrando de esta manera que la evolución de las redes se realice a la misma velocidad que se desarrolla el software, permitiendo la reutilización de código que fue creado para solucionar una alternativa en particular, en problemas similares. Abaratando costos de implementación y sobre todo sin verse obligado a utilizar equipos de un solo fabricante o marca.

1.5 Objetivos

General

Analizar las redes definidas por software (SDN) en base a una infraestructura de software de libre distribución.

Objetivos Específicos

- Estudiar las redes definidas por software (SDN).
- Analizar una alternativa de software de libre distribución para el prototipo de red definida por software (SDN).
- Implementar un prototipo de red definida por software (SDN).

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes Investigativos

Debido a que se trata de una nueva tecnología existen pocos proyectos realizados. En el Ecuador se encontraron los siguientes proyectos:

En el año 2013 en la Escuela Politécnica Nacional Juan Carlos Chico Jiménez realizo el proyecto de titulación denominado “Implementación de un prototipo de una red definida por software (SDN) empleando una solución basada en hardware”, el cual consiste en el diseño e implementación de una red SDN. Para la realización del proyecto se utilizan diferentes herramientas como es el caso del software Mininet, el mismo que incluye el protocolo de comunicaciones OpenFlow necesario para la simulación de este tipo de redes, y posteriormente se trabaja con un controlador diferente al que viene instalado por defecto en el software de simulación. Mediante el cual se añade más funcionalidades y gestiona de manera personalizada el tráfico y los requerimientos de la red.

En la etapa de diseño y simulación se evalúan los resultados obtenidos con el objetivo de realizar la implementación del prototipo físico. Determinando así la topología que se utilizará, los componentes que compondrán la red y el software de programación necesario para realizar los distintos cambios en el controlador en dependencia de los requerimientos de la red, con lo cual se corroboró las ventajas

que presentan las redes definidas por software ya que permiten modelar la red una manera ágil y personalizada además que proveen una visión global de la misma [5].

Otro proyecto similar se realizó en el año 2014, en la Escuela Politécnica Nacional por Diana Gabriela Morillo Fueltala denominado "Implementación de un prototipo de una red definida por software (SDN) empleando una solución basada en Software". En el cual se implementa una red SDN utilizando una infraestructura basada en software, debido a esto se reemplazan los equipos físicos por máquinas virtuales. Para el desarrollo del prototipo de red se utilizaron máquinas virtuales para emular a los host de la topología y Open VSwitch para emular un switch OpenFlow, además se desarrolló una aplicación NAC (Network Access Control) mediante la cual se permite o deniega el acceso a la red, siendo Floodlight y Trema los controladores utilizados para el desarrollo para el desarrollo de las aplicaciones SDN.

Al finalizar el proyecto se concluye que: las redes definidas por software son ideales para el control de tráfico en las redes ya que hacen posible la implantación de un sin número de aplicaciones. Se corroboró además que la implementación de equipos virtuales para el desarrollo de la infraestructura SDN es una solución mucho más económica y brinda las mismas características y potencia que una infraestructura física [6].

En el exterior existen algunos proyectos realizados acerca de redes definidas por software. A continuación de describen algunos de estos proyectos:

En el 2012 Sergio Rodríguez Santamaría de la universidad Cantabria en España realizó el proyecto denominado "Mecanismos de control de las comunicaciones en la internet del futuro a través de OpenFlow". En este proyecto se detallan las características del protocolo OpenFlow que hace posible la consecución de las redes definidas por software, también se estudian herramientas como el software de simulación Mininet, el cual permite crear redes virtuales escalables sobre un mismo equipo de trabajo, con el objetivo de facilitar la realización de pruebas a los usuarios, sin necesidad de requerir de un entorno de trabajo más complejo.

Además se menciona la importancia que tiene el estudio y familiarización del controlador NOX, ya que permite implementar funcionalidades que puedan ser aplicables a un switch con tecnología OpenFlow. Concluyendo que OpenFlow será uno de los principales protocolos de routing en las nuevas redes, especialmente en entornos de datos y cloud computing [7].

De igual manera Ana Milena Rojas Calero en el año 2012 en la universidad ICESI de Colombia realizó el proyecto denominado "Propuesta para la implementación de un laboratorio de acceso remoto usando redes definidas en software". En el cual se plantea la implementación de un laboratorio de acceso remoto utilizando redes definidas por software, este proyecto se desarrolló con el objetivo de proveer laboratorios remotos a los investigadores ya que son utilizados en diferentes ramas de la investigación y la ingeniería.

La implementación de este laboratorio se llevó a cabo con una topología prototipo creada en lenguaje de programación Phyton, y su simulación se la realizó con el software Mininet, ya que proveía una plataforma escalable para la simulación de redes SDN. Realizando además la configuración de las interfaces de los dispositivos conectados a la red y la gestión de las Tablas ARP. Llegando a la conclusión de que las redes definidas por software son ideales para los investigadores ya que permiten poner en práctica los experimentos realizados y ejecutarlos en tiempo real, además de que abre nuevos campos de aplicación como es el caso de la docencia y la investigación [3].

2.2 Fundamentación Teórica

2.2.1 Redes Definidas por Software

- **Introducción**

La aparición de nuevos servicios y aplicaciones en el internet, así como el incremento de dispositivos móviles, los servicios de la nube, y la gran cantidad de información que hoy en día manejan los servidores ha provocado que la infraestructura de red actual sea insuficiente para satisfacer todas estas demandas [8] [9].

La continua evolución de estos servicios ha traído consigo grandes retos tanto a empresas como a desarrolladores de hardware y software de redes, los cuales buscan explotar al máximo la infraestructura de red existente para permitir satisfacer todas estas necesidades [10].

Los Carriers se enfrentan a retos similares tales como la demanda de movilidad y la explosión de ancho de banda, debido a que los patrones de trafico están cambiando rápidamente, en los últimos años este ha tendido a generarse entre los servidores creando conflictos, ya que tradicionalmente las redes fueron diseñadas para dar prioridad al trafico cliente servidor y no entre servidores [8] [9] [10].

La utilización de todos estos servicios provoca un incremento en el tamaño de la red haciéndola cada vez más compleja, ya que si antes una red tenía 100 nodos ahora podría tener 1000 lo cual implica 1000 sistemas de configuración, esto ha hecho que los fabricantes busquen nuevas maneras para que la programación no se realice en cada nodo sino de una manera global [9].

La arquitectura de red actual es ineficiente para satisfacer todas estas demandas, por lo cual se ha pensado en una manera de mejorar su eficiencia sin la necesidad de cambiar toda la infraestructura existente y permitiendo reducir tantos y complejos protocolos que además de no soportar un alto nivel de escalabilidad son incapaces de manejar la gran cantidad de datos que se manejan hoy en día, es por ello que surgen las redes definidas por software como una solución a todos estos problemas [8].

- **Definición**

Las redes definidas por software se definen como una arquitectura de red dinámica, manejable, adaptable, de costo eficiente. Lo cual la hace ideal para las altas demandas de ancho de banda y la naturaleza dinámica de las aplicaciones actuales. Esta arquitectura desacopla el control de la red y la funcionalidad de reenvió de información permitiendo que el control de la red pueda ser completamente programable logrando que la infraestructura de red subyacente sea abstraída por las aplicaciones y servicios de red [11].

La migración de la lógica de control que solía estar estrechamente ligada en los dispositivos de red tales como switches y routers Ethernet permite que la inteligencia de la red se desprenda del hardware y se delegue a una aplicación de software denominada controlador. Permitiendo así conseguir redes más automatizables y flexibles [9].

Las redes definidas por software eliminan la rigidez presente en las redes tradicionales, su estructura permite que el comportamiento de la red sea más flexible y adaptable a las necesidades de cada organización, campus, o grupo de usuarios. Además su diseño centralizado permite obtener información importante de la red y adaptar sus políticas de forma dinámica [8].

La eficiencia de las SDN radica en programar el plano de control a través de un conjunto de aplicaciones, es decir, la posibilidad de crear aplicaciones a través de una API (Interfaces de programación de aplicaciones). Estas aplicaciones simplifican la implementación de servicios de red comunes (por ejemplo, el enrutamiento, multidifusión (multicast), seguridad , control de acceso , gestión de ancho de banda, ingeniería de tráfico , calidad de servicio (QoS) entre otros [12] [13].

El Objetivo de las redes SDN es conseguir redes más sencillas, programables y flexibles, así como como crear redes más escalables y automatizables, tener un control centralizado, aumento en la seguridad y fiabilidad de la misma en la figura 2.1 se muestra la arquitectura de una red SDN [10].

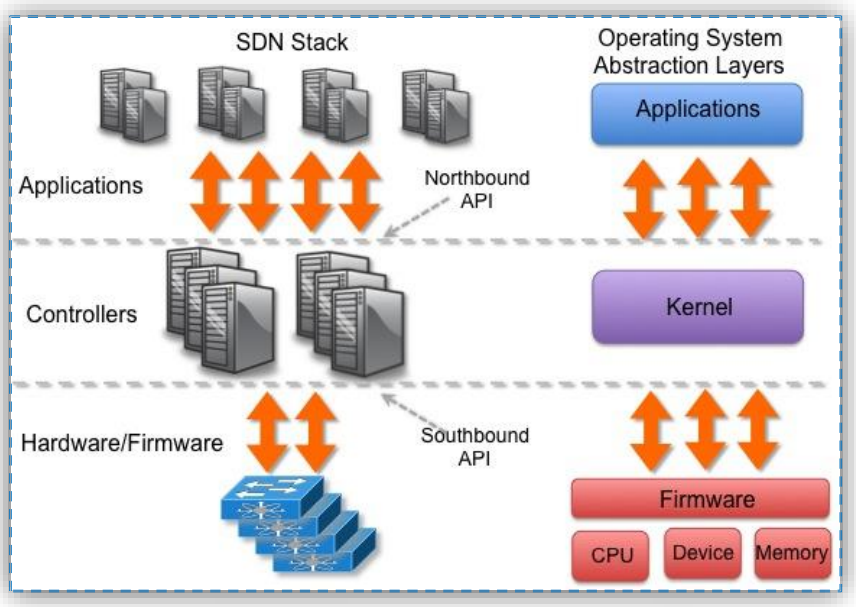


Fig 2. 1 Arquitectura de las redes SDN

Fuente: http://networkstatic.net/wp-content/uploads/2012/06/SDN-Abstraction.jpg

- **Necesidad de una nueva arquitectura de red**

Existen varias tendencias que impulsan a la industria del networking a reexaminar su arquitectura, tales como:

- **Complejidad de la arquitectura de red actual:** La tecnología de red hasta la fecha ha consistido en gran medida en un conjunto de protocolos discretos diseñados para conectar hosts de forma fiable a través de distancias arbitrarias, velocidades de enlace, y topologías.

  Para satisfacer las necesidades técnicas en las últimas décadas, la industria ha desarrollado protocolos de red para ofrecer un mayor rendimiento, fiabilidad, conectividad, y la más estricta seguridad. Estos protocolos tienden a ser definidos en aislamiento, sin embargo, con cada solución de un problema específico y sin el beneficio de ninguna abstracción fundamental ha dado lugar a una de las principales limitaciones de las redes de hoy en día:

Por ejemplo, para agregar o remover cualquier dispositivo, se debe programar múltiples switches, routers, firewalls, portales Web de autenticación, etc. y actualización de ACL (Listas de control de acceso), VLAN, calidad de servicios (QoS), y otros mecanismos basados en protocolos que utilizan las herramientas de gestión a nivel de dispositivo. Además, la topología de la red, modelo del switch de fabricante, y la versión del software. Todo debe ser tomando en cuenta. Debido a esta complejidad, las redes actuales son relativamente estáticas.

La mayoría de las empresas de hoy operan una red convergente IP para voz, datos y tráfico de vídeo. Mientras que las redes existentes pueden proporcionar niveles de calidad de servicio diferenciados para diferentes aplicaciones, la provisión de esos recursos es altamente manual. Los equipos de cada proveedor se deben configurar por separado y ajustar parámetros tales como ancho de banda y calidad de servicio uno por uno. Debido a su naturaleza estática, la red no puede adaptarse dinámicamente a los cambios de tráfico, aplicaciones y las actuales demandas de los usuarios.

- **Políticas inconsistentes:** Para implementar una política de toda la red, se tendría que configurar miles de dispositivos y mecanismos. Por ejemplo, cada vez que una nueva máquina virtual es creada, puede tardar horas, en algunos casos días, para que las ACLs sean reconfiguradas en toda la red. La complejidad de las redes de hoy en día hace que sea muy difícil aplicar un consistente conjunto de acceso, seguridad, calidad de servicio, y otras políticas que cada vez más usuarios móviles demandan, lo que deja a las redes vulnerables a las violaciones de la seguridad, el incumplimiento de las regulaciones, y otras consecuencias negativas.

- **Incapacidad para escalar*:*** Debido a que las demandas en los data centers crecen rápidamente, la red también crece. Sin embargo, se vuelve muy compleja con la adición de cientos o miles de dispositivos nuevos que deben ser configurados y gestionados. Por lo cual las empresas se han basado en patrones de tráfico previsibles; sin embargo, en los centros de datos

virtualizados de hoy, los patrones de tráfico son increíblemente dinámicos y por lo tanto impredecibles.

Mega-operadores, tales como Google, Yahoo! y Facebook, se enfrentan a más desafíos desalentadores de escalabilidad. Estos proveedores de servicios emplean algoritmos de procesamiento en paralelo a gran escalar y conjuntos de datos asociados a través de toda la infraestructura de red. A medida que el alcance de las aplicaciones de usuario final se incrementa (por ejemplo, el rastreo y la indexación de toda la web en todo el mundo para devolver al instante resultados de búsqueda para los usuarios), el número de elementos de computación explota y el número de información que se intercambian entre los nodos de cómputo pueden alcanzar peta bytes. Estas empresas necesitan las llamadas redes hiperescala que pueden proporcionar alto rendimiento, conectividad de bajo costo entre millones los servidores físicos. Dicho ajuste no se puede hacer con configuración manual.

Para seguir siendo competitivos, los carriers deben ofrecer cada vez mejores servicios diferenciados a los clientes, ya que la red debe atender grupos de usuarios con diferentes aplicaciones y diferentes necesidades de rendimiento. Operaciones claves que parecen relativamente sencillas, como dirigir el tráfico de un cliente para proporcionar flujos control de rendimiento personalizado o entrega bajo demanda, son muy complejas para poner en práctica con las redes existentes, especialmente a escala de carriers. Debido a que requieren de dispositivos especializados en el borde de la red, lo que aumenta el capital y los gastos de operaciones, así como el tiempo de lanzamiento al mercado para introducir nuevos servicios.

- **Dependencia del proveedor:** Los carriers y las empresas buscan implementar nueva capacidades y servicios de respuesta rápida a las cambiantes necesidades y demandas de los usuarios. Sin embargo, su capacidad de respuesta se ve obstaculizada por los proveedores de los productos y equipos, que pueden demorar hasta tres años o más para lanzar nuevos dispositivos con las características que se requieren. La falta de estándares e interfaces abiertas

limita la capacidad de los operadores para adaptar la red a sus entornos individuales.

Esta falta de correspondencia entre las necesidades del mercado y las capacidades de la red ha llevado a la industria a un punto de inflexión. En respuesta, la industria ha creado la Red (SDN), arquitectura definida por software y el desarrollo de estándares asociados.

- **El cambio de los patrones de tráfico:** Dentro de la empresa de los data centers, los patrones de tráfico han cambiado significativamente. En contraste con las aplicaciones cliente-servidor donde se produce la mayor parte de la comunicación entre un cliente y un servidor, las aplicaciones de hoy en día acceden a diferentes bases de datos y servidores, creando un flujo de tráfico "Este-Oeste" de máquina a máquina, antes de regresar datos al dispositivo de usuario final en el patrón de tráfico clásico "norte-sur". En la figura 2.2 se muestra este proceso:

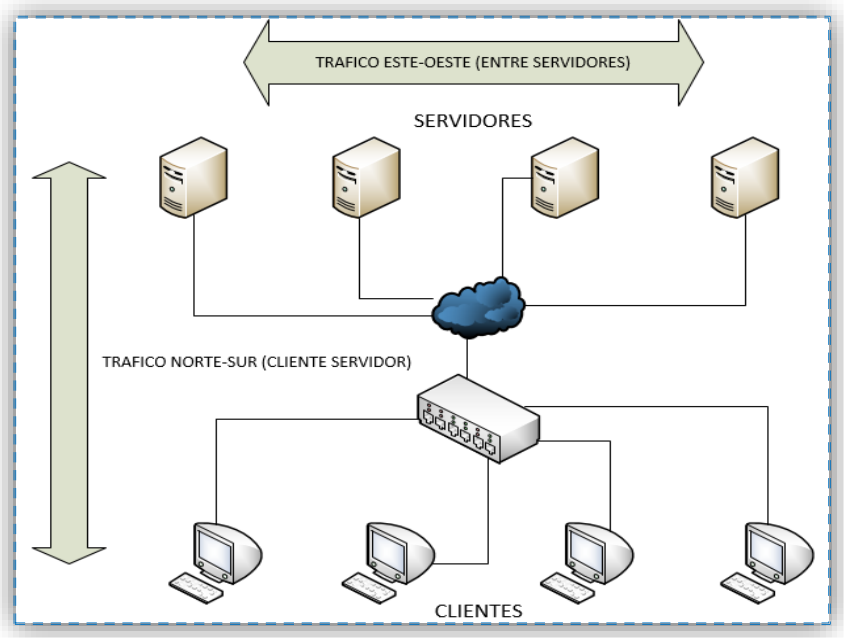


Fig 2. 2 Tendencias de tráfico

Al mismo tiempo, los usuarios están cambiando los patrones de tráfico de red a medida que impulsan el acceso a contenidos y aplicaciones corporativas desde cualquier tipo de dispositivo, conectándose desde cualquier lugar, en cualquier momento, lo que resulta en el tráfico adicional a través de la red WAN.

- **El consumo de las tecnologías de información (IT):** Los usuarios están empleando cada vez más dispositivos personales móviles, como smartphones, tablets y ordenadores portátiles para el acceso a la red. Las tecnologías de información se encuentran bajo presión para dar cabida a todos estos dispositivos personales de una manera adecuada y al mismo tiempo proteger los datos corporativos y la propiedad intelectual.

- **El aumento de los servicios en la nube:** Las empresas han adoptado con entusiasmo los servicios de la nube, tanto públicos como privados, lo que resulta en un crecimiento sin precedentes de estos servicios. Siendo así que las unidades de negocio empresariales ahora quieren mayor agilidad a la hora acceder a aplicaciones, infraestructura y otros recursos de computación.

- **"Big data" mayor demanda de ancho de banda:** Manejar la gran cantidad de datos de hoy en día requiere un procesamiento paralelo masivo en miles de servidores, todos de los cuales necesitan conexiones directas entre sí. El surgimiento de esta gran cantidad de información está impulsando una demanda constante de la capacidad de red adicional en los data centers. Los operadores de redes de data centers de hiperescala se enfrentan a la desalentadora tarea de ampliar la red a un tamaño inimaginable [10].

♦ **Comparación entre la arquitectura de red actual y la arquitectura SDN**

En el networking tradicional él envió de paquetes y las decisiones de enrutamiento de alto nivel suceden dentro del mismo dispositivo, mientras que en las redes definas por software estas dos funcionalidades se separan permitiendo tener un control centralizado, en la figura 2.3 se muestra la diferencia de estas dos arquitecturas [14].

En una red convencional cuando un paquete llega a un switch, las reglas integradas en el firmware propietario del switch le dicen al switch a donde transferir el paquete, el switch envía cada paquete al mismo destino por la misma trayectoria y trata a todos los paquetes de la misma manera. Mientras que en las redes definidas por software es posible tener acceso a las tablas de flujo del switch donde se encuentras contenidas estas reglas con la finalidad de modificar, eliminar o añadir nuevas reglas de flujo, lo cual permite controlar el tráfico de la red de manera personalizada [9] [12] [13].

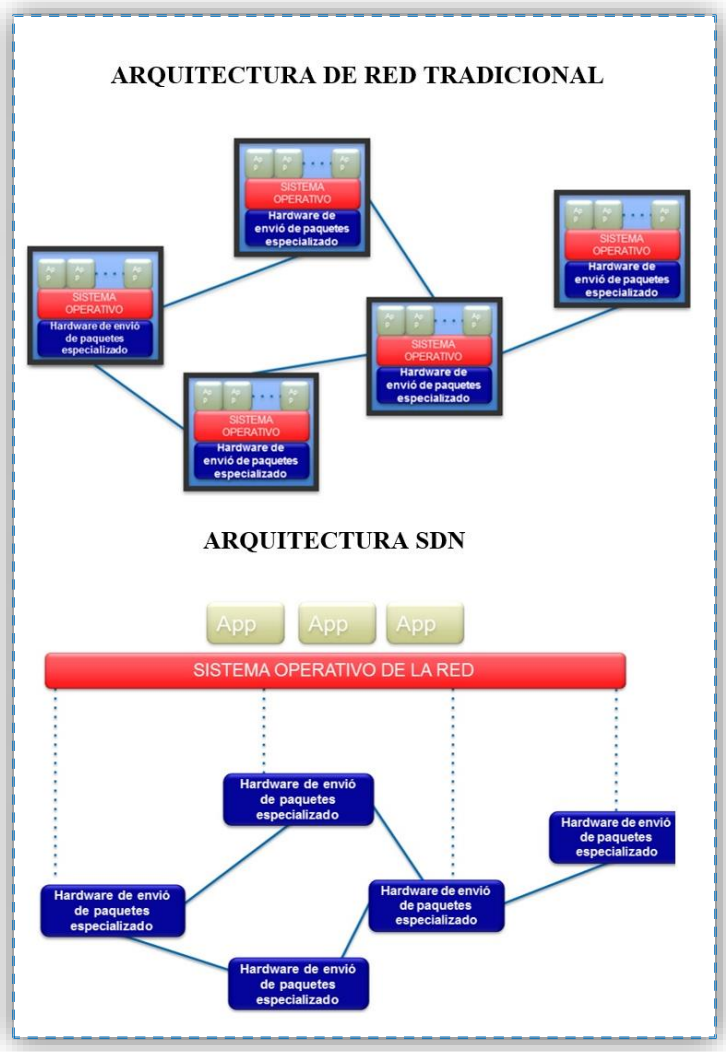


Fig 2. 3 Comparación entre la arquitectura de red tradicional y la arquitectura SDN.

Fuente: http://es.slideshare.net/openflow/openflow-tutorial

En una red definida por software, un administrador de red puede darle forma al tráfico desde una consola de control centralizada sin tener que tocar conmutadores individuales. El administrador puede cambiar cualquier regla de los conmutadores de red cuando sea necesario dando o quitando prioridad, o hasta bloqueando tipos específicos de paquetes con un nivel de control muy detallado algo que no es posible en el entorno de red actual en la figura 2.4 se puede visualizar este proceso [9].

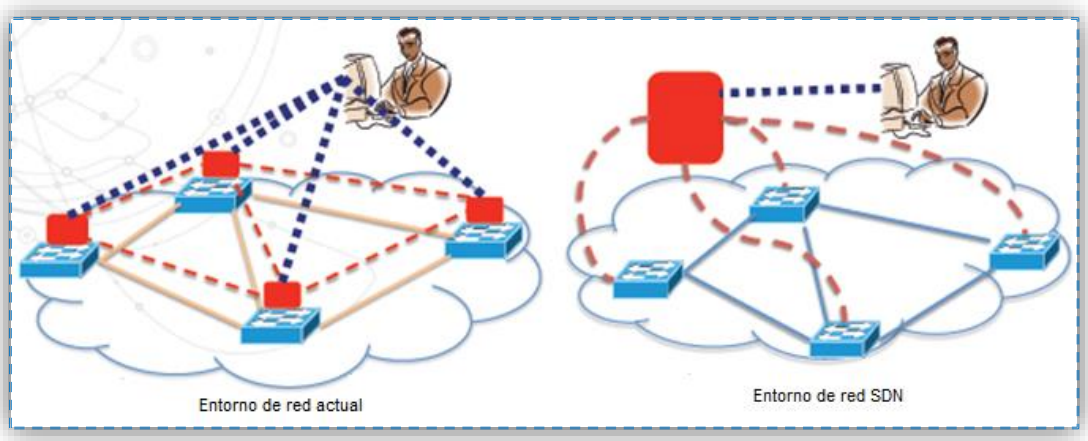


Fig 2. 4 Gestión individual vs gestión centralizada

Fuente:http://openlab.web.cern.ch/publications/presentations/software-defined-networking-technology-details-and-openlab-research

En la tabla 2.1 se presentan las diferencias existentes entre estas dos arquitecturas.

Tabla 2. 1 Comparación entre la arquitectura de red tradicional y la arquitectura SDN.

ARQUITECTURA DE RED ACTUAL	ARQUITECTURA DE RED SDN
• Envió de paquetes basados en coincidencias de la tabla • Las tablas de flujo están cerradas en los dispositivos	• Tablas de enrutamiento abiertas • Formato y acciones de las tablas claramente especificadas • APIs bien definidas

• Protocolos totalmente distribuidos • Interfaces propietarias • Configuración individual de los dispositivos • Automatización posible pero tediosa	• Control lógicamente centralizado en el Software controlador • APIs abiertas para el acceso y manipulación del plano de Datos ○ Ej: OPENFLOW • Control central • Una solo interfaz (API) para todos los dispositivos [15] [16].

♦ **Beneficios de las redes definidas por software**

Los beneficios que las empresas y los usuarios pueden lograr a través de una arquitectura SDN se presentan a continuación:

- **Control centralizado de entornos de múltiples proveedores:** El software de control SDN puede controlar cualquier dispositivo de red habilitado para OpenFlow de cualquier proveedor, incluyendo switches, routers y switches virtuales. En lugar de tener que gestionar grupos de dispositivos de un solo vendedor.

- **Reducción de la complejidad a través de la automatización:** La redes SDN ofrecen un marco de automatización y gestión de red flexible, lo que hace posible el desarrollo de herramientas que automatizan muchas tareas de administración que se realizan de forma manual hoy en día.

- **Mayor tasa de innovación:** la adopción SDN acelera la innovación empresarial permitiendo a los operadores de red, literalmente programar y reprogramar la red en tiempo real para satisfacer las necesidades específicas de negocio y de los usuarios a medida que estas surgen.

- **Aumento de la fiabilidad y seguridad de la red:** SDN hace posible definir sentencias de configuración y de política de alto nivel, que luego son traducidas a la infraestructura a través de OpenFlow. Una Arquitectura SDN basada en OpenFlow elimina la necesidad de configurar individualmente los dispositivos de red cada vez que se realice un cambio en un punto final, servicio, o aplicación, lo que reduce la probabilidad de fallas en la red debido a configuraciones erróneas o políticas inconsistentes.

  Dado que los controladores SDN proporcionan una completa visibilidad y control sobre la red, pueden asegurarse de que el control de acceso, la ingeniería de tráfico, calidad de servicio, seguridad y otras políticas son aplicadas consistentemente a través de las infraestructuras de redes cableadas e inalámbricas, incluyendo las sucursales, los campus y data centers.

- **Mejor experiencia de usuario:** Al centralizar el control de la red y hacer que el estado de la información esté disponible para las aplicaciones de alto nivel, una infraestructura SDN puede adaptarse mejor a las necesidades dinámicas del usuario [10].

2.2.2 Protocolo OpenFlow

El protocolo OpenFlow fue originalmente propuesto como una alternativa para el desarrollo de protocolos experimentales en el campus de una universidad, en donde es posible probar nuevos algoritmos sin interrumpir o interferir con la normal operación del tráfico y surgió a raíz del proyecto de investigación "OpenFlow: Enabling Innovation in Campus Networks" en la universidad de Stanford en el 2008 [8].

OpenFlow se define como un protocolo emergente y abierto de comunicaciones y es la primera interfaz de comunicaciones estándar definida entre los planos de control y datos de una arquitectura de SDN. OpenFlow permite el acceso directo y manipulación del plano de reenvío de los dispositivos de red tales como switches y routers, tanto físicos como virtuales. OpenFlow hace posible que se pueda mover el control de la red fuera de los dispositivos para centralizarlo lógicamente en el

software de control y establecer las primitivas básicas para programar el plano de reenvió [10] [17].

El protocolo OpenFlow se implementa entre la infraestructura de los dispositivos de red y el software de control de SDN. OpenFlow utiliza el concepto de flujos para identificar el tráfico de red basado en reglas de coincidencia predefinidas que pueden ser programadas de forma estática o dinámica por el un controlador SDN lo cual permite definir cómo el tráfico debe fluir a través de los dispositivos de red.

En la actualidad la estandarización de OpenFlow está a cargo de la Open Networking Foundation y lo hace a través de grupos de trabajo técnicos encargados de la configuración, pruebas de interoperabilidad, y otras actividades del protocolo, ayudando a garantizar la interoperabilidad entre los dispositivos de red y software de control de diferentes proveedores [10].

- **Funcionamiento de OpenFlow**

OpenFlow aprovecha el hecho que la mayorías de los switches Ethernet contienen tablas de flujo (Flow-Tables), aunque cada una de estas son propias de los fabricantes se han identificado varias características en común las cuales son utilizadas por OpenFlow para programar dichas tablas, en la figura 2.5 se muestra un ejemplo de una tabla de flujo.

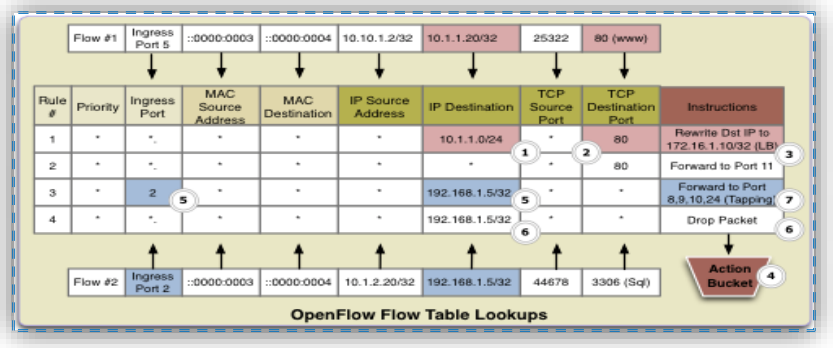


Fig 2. 5 Tabla de flujo OpenFlow

Fuente:http://etherealmind.com/sdn-use-case-firewall-migration-in-the-enterprise/

Cuando el primer paquete de un flujo llega al switch, este verifica las tablas de flujo con el objetivo de encontrar coincidencias, si ninguna coincidencia es encontrada el paquete es enviado al controlador y este es el encargado de insertar una entrada de flujo en la tabla del switch, luego de insertar esta entrada los paquetes que coincidan con la nueva entrada añadida se envían directamente a su destino sin la necesidad de enviarlo al controlador [18].

2.2.3 Switch OpenFlow

Un switch OpenFlow está compuesto por dos elementos esenciales que son: el canal seguro y la tabla de flujos como se muestra en la figura 2.6 [19].

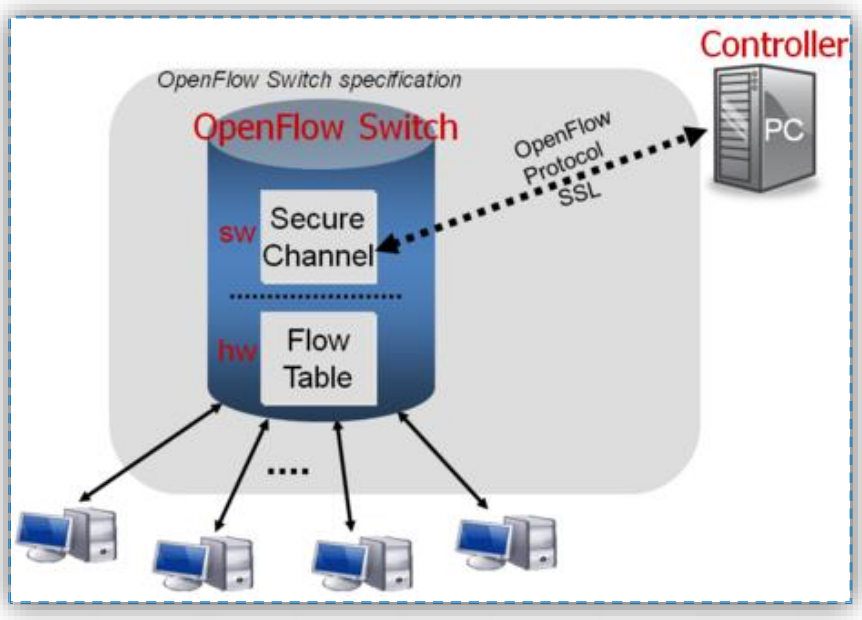


Fig 2. 6 Estructura de un switch OpenFlow

Fuente: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

- **Tabla de flujos**

La tabla de flujo contiene un conjunto de entradas de flujos, las cuales a su vez contienen constan de tres partes importantes que son: los campos de cabecera (cabeceras del paquete para poder comparar los paquetes entrantes), los contadores de actividad, y un conjunto de cero o más acciones para aplicar a paquetes coincidentes.

Todos los paquetes procesados por el switch se comparan con las entradas de flujo contenidas en la tabla, si se encuentra una entrada coincidente, alguna acción para esa entrada es realizada (por ejemplo, la acción podría ser la de enviar un paquete fuera de un puerto especifico). Si no se encuentra ninguna coincidencia, el paquete se envía al controlador a través del canal seguro.

A continuación se describen los componentes de una entrada de flujos y el proceso por el cual los paquetes entrantes son comparados con dichas entradas. En la figura 2.7 se muestra los componentes de una entrada de flujo [19] [20].

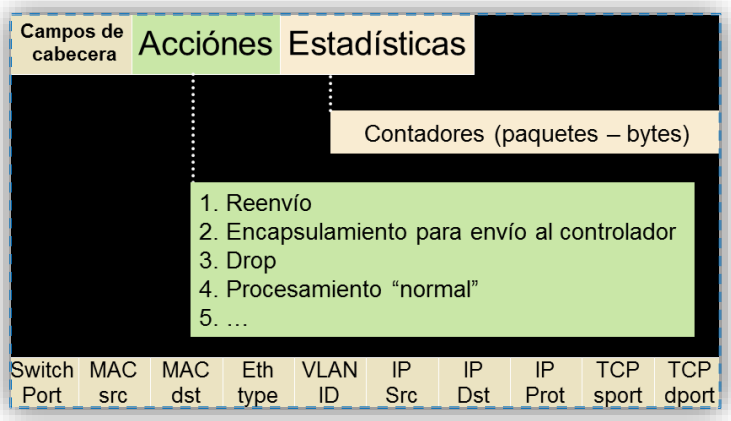
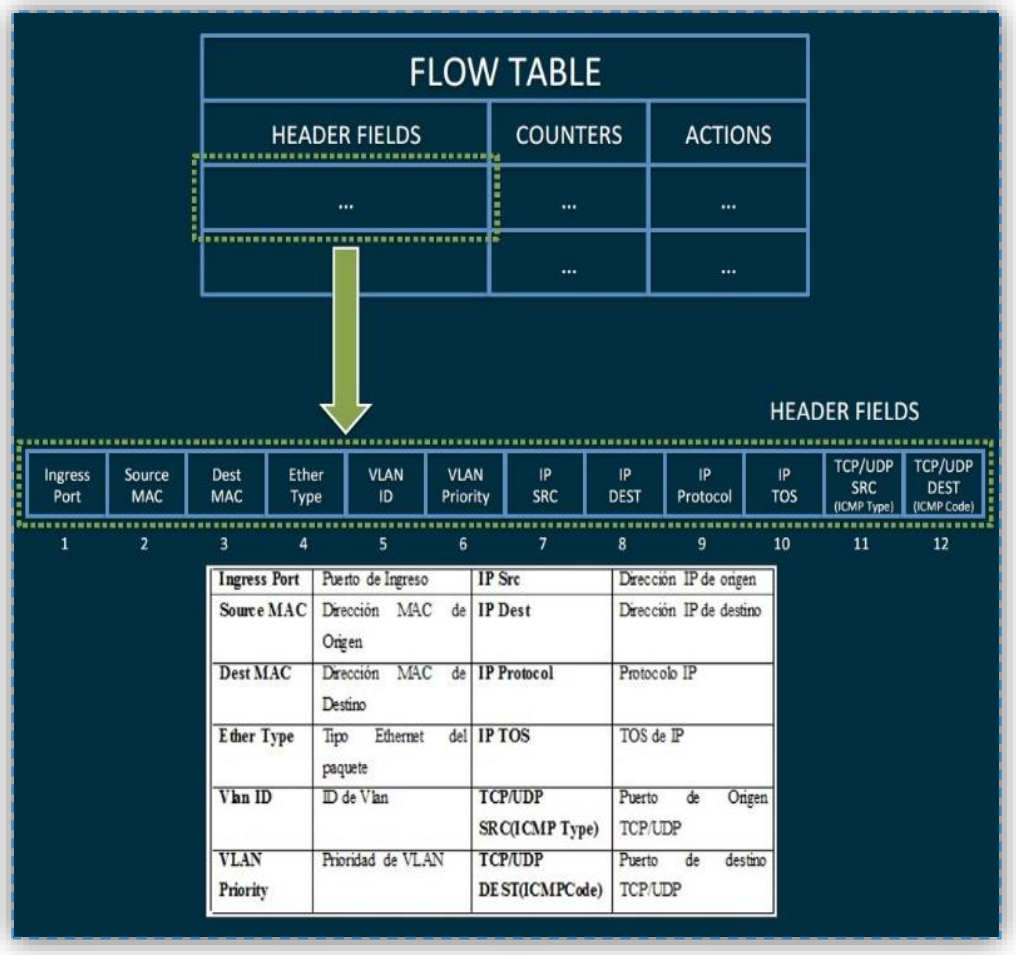


Fig 2. 7 Estructura de una tabla de flujos

Fuente:http://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB0QFjAA&url=http%3A%2F%2Fticec.cedia.org.ec%2Fdmdocuments%2FDocumentacion%2FDia2%2Fponencia8.pptx&ei=qC76U8jIH4zlsAS5h4CYAQ&usg=AFQjCNHa4Vq5ew9bDAxWIpZAEQ0tWmp4sQ

- Campos de cabecera

  Los campos de cabecera contienen información que se encuentra en la cabecera del paquete y es utilizada para comparar los paquetes entrantes. En la figura 2.8 se muestra estos campos [12] [20].



Ingress Port	Puerto de Ingreso	**IP Src**	Dirección IP de origen
Source MAC	Dirección MAC de Origen	**IP Dest**	Dirección IP de destino
Dest MAC	Dirección MAC de Destino	**IP Protocol**	Protocolo IP
Ether Type	Tipo Ethernet del paquete	**IP TOS**	TOS de IP
Vlan ID	ID de Vlan	**TCP/UDP SRC(ICMP Type)**	Puerto de Origen TCP/UDP
VLAN Priority	Prioridad de VLAN	**TCP/UDP DEST(ICMPCode)**	Puerto de destino TCP/UDP

Fig 2. 8 Campos de cabecera

Fuente:http://www.cisco.com/web/CZ/ciscoconnect/2014/assets/tech_sdn2_sp_api_openflow_ungerman.pdf

En la figura 2.9 se muestra como el switch analiza los campos de cabecera para determinar las coincidencias.

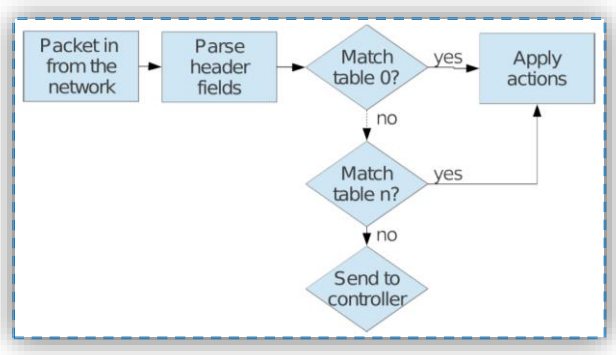


Fig 2. 9 Descripción del proceso de análisis de paquetes coincidentes

Fuente:https:http://www.csg.ethz.ch/education/lectures/ATCN/hs2012/schedule/hwpres/Gaemperli_slides.pdf

- Contadores

Los contadores son utilizados para recoger estadísticas del flujo particular, así como el número de paquetes recibidos, número de bytes, y el tiempo de vida del flujo. En la figura 2.10 se muestran los contadores y su valores [12].

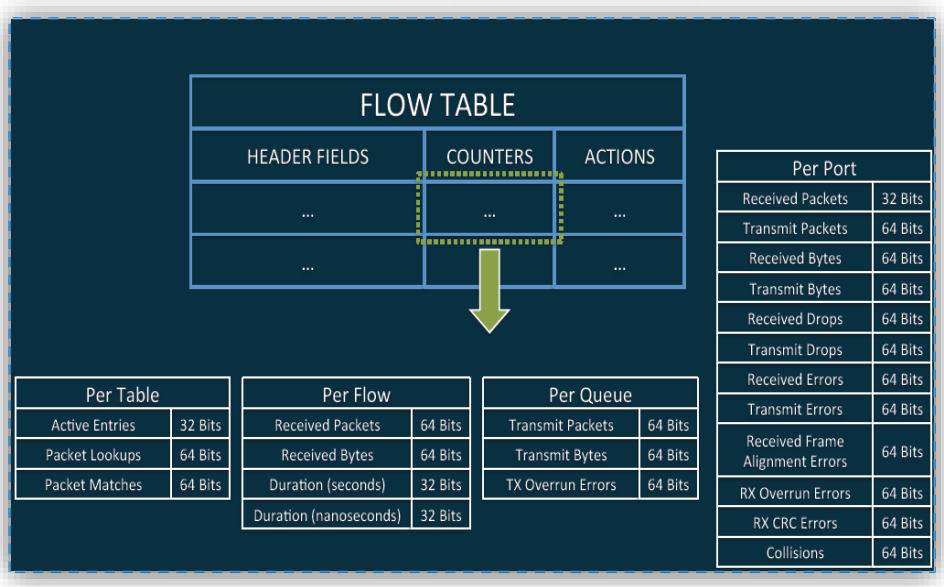


Fig 2. 10 Descripción de los contadores

Fuente:http://www.cisco.com/web/CZ/ciscoconnect/2014/assets/tech_sdn2_sp_api_openflow_ungerman.pdf

- Acciones

Cada entrada de flujo se asocia con cero o más acciones que dictan cómo el switch trata a los paquetes coincidentes. Si no hay acciones de envió presentes, el paquete se descarta. Las listas de acciones para las entradas de flujo insertadas deben ser procesadas en el orden especificado. Sin embargo, no hay un orden de paquetes de salida garantizado dentro de un puerto.

Un switch puede rechazar una entrada de flujo si no se puede procesar la lista de acciones en el orden especificado, en cuyo caso se debería devolver inmediatamente un error de flujo no soportado.

A continuación se detalla la lista de acciones soportadas por el protocolo OpenFlow:

1. **Acción requerida:** Los switches OpenFlow deben soportar el reenvío el paquete a puertos físicos y a los siguientes puertos virtuales:

   ALL: Envía el paquete a todas las interfaces, sin incluir la interface de entrada.

   CONTROLADOR: Encapsular y enviar el paquete al controlador.

   LOCAL: Envíe el paquete a la pila de red local de los switch.

   TABLE: Realizar acciones en la tabla de flujo. Solamente para mensajes de tipo packet-out.

   IN_PORT: Enviar el paquete al puerto de entrada.

2. **Acción Opcional:** El switch puede soportar opcionalmente los siguientes puertos virtuales:

   NORMAL: procesar el paquete utilizando la ruta de reenvío tradicionales soportada por el switch (es decir, el procesamiento tradicional de la capa 2, VLAN, y La capa 3) .El switch se puede

comprobar el campo VLAN para determinar si debe o no reenviar el paquete a lo largo de la ruta de procesamiento normal. Si el switch no puede enviar entradas por el VLAN OpenFlow especificada regresa a la ruta de procesamiento normal, esto debe indicar que no admite esta acción.

FLOOD: Inundación del paquete a todas las interfaces del switch excepto la interfaz de entrada.

3. **Acción Opcional:** *Enqueue*. La acción enqueue envía un paquete a través de un cola conectado a un puerto. El comportamiento de reenvío está dictaminado por la configuración de la cola y se utiliza para proporcionar soporte básico de calidad de servicio (QoS).

4. **Acción requerida:** *Drop*. Una entrada de flujo con ninguna acción especificada indica que todos los paquetes que concuerden deben descartarse.

5. **Acción Opcional:** *Modifique-Field*. Este tipo acción permite los campos de cabecera de un flujo determinado [12] [19].

- **Canal seguro**

El canal seguro es la interfaz encargada de conectar cada switch OpenFlow a un controlador. A través de esta interfaz, el controlador configura y gestiona al switch, recibiendo eventos desde el switch y enviando paquetes al mismo [12] [19].

2.2.4 Tipos de Switches OpenFlow

Existen dos tipos de switches OpenFlow aquellos que soportan únicamente el protocolo OpenFlow denominados OpenFlow only y los switch híbridos.

- **Switches OpenFlow-only**

  Este tipo de switches soportan únicamente el protocolo OpenFlow y no tienen características de control en su interior es decir confían plenamente en un controlador para la toma de decisiones. Un switch que soporta OpenFlow, contiene múltiples tablas de flujos, y cada flujo contiene múltiples entradas de flujo. El procesado define como los paquetes interactuarán con estas tablas de flujos. Requiere que tenga al menos una tabla de flujo y opcionalmente más. Cuantas menos entradas más simplificado será el procesado.

- **Switches OpenFlow-híbridos**

  Este tipo de Switches que soportan tanto la operación OpenFlow como el Switching Ethernet convencional. Las operaciones habituales pueden ser: Switching Ethernet de Capa 2, aislamiento de tráfico con VLAN, nivel 3 o de routing (IPV4, IPV6), listas de acceso o QoS. Estos switches deberán proveer un mecanismo de clasificación fuera de OpenFlow que enrute el tráfico o bien ser procesado por OpenFlow. Por ejemplo, un switch deberá usar una etiqueta VLAN o puerto de entrada para decidir si se procesa el paquete de una manera u otra [12].

2.2.5 Mensajes OpenFlow

La comunicación entre el controlador y el switch ocurre mediante la utilización del protocolo OpenFlow, durante este proceso una serie de mensajes son intercambiados.

OpenFlow Soporta tres tipos de mensajes que se describen a continuación:

- **Mensajes del controlador al switch**

  Estos mensajes son iniciados por el controlador y se usan para gestionar o inspeccionar el estado del switch. Este tipo de mensajes pueden o no requerir una respuesta de parte del switch y poseen varios subtipos tales como:

- Features

  Una vez establecida la sesión TLS (Seguridad en la Capa de Transporte), el controlador envía un mensaje feature request al switch. El switch debe enviar un mensaje features reply el cual especifica las características y capacidades soportadas por el switch.

- Configuration

  El controlador es capaz de establecer una consulta de configuración de paramentos en el switch. El switch únicamente responde a una consulta proveniente del controlador.

- Modify-State

  Este tipo de mensaje es enviado por el controlador para gestionar el estado de los switches. Su propósito principal es añadir modificar o borrar entadas de flujo en la tabla y establecer las propiedades del switch.

- Read-State

  Este tipo de mensajes son utilizados por el controlador para recolectar estadísticas desde las tablas de flujo del switch, así como de puertos y de entradas de flujo individuales.

- Send-Packet

  Este tipo de mensaje es usado por el controlador para enviar paquetes a un especificado puerto del switch.

- Barrier

  Este tipo de mensajes son utilizados por el controlador para asegurar que se cumplan las dependencias de los mensajes o recibir notificaciones de operaciones completadas.

- **Mensajes simétricos**

  Los mensajes simétricos son iniciados por el switch o el controlador y enviados sin necesidad de una solicitud. Hay tres tipos de mensajes simétricos en OpenFlow:

  - Hello

    Los mensajes hello son intercambiados entre el switch y el controlador al establecer la conexión.

  - Echo

    Los mensajes Echo pueden ser enviados por el switch o el controlador y se deberá devolver un echo reply. Este mensaje puede ser usado para indicar la latencia, el ancho de banda y el tiempo que vida de la conexión entre el switch y el controlador.

  - Vendor

    Este tipo de mensajes proveen una manera estándar para que los switches OpenFlow ofrezcan funcionalidades adicionales dentro del espacio de tipo de mensaje OpenFlow para futuras revisiones de OpenFlow.

- **Mensajes Asíncronos**

  Este tipo de mensajes son iniciados por el switch sin que el controlador los solicite y se utilizan para avisar al controlador que un paquete ha arribado. Existen cuatro mensajes asíncronos.

  - Packet-in

    Se envía un mensaje Packet-in al controlador cuando un paquete no tiene una entrada de flujo o si el paquete concuerda con una entrada cuya acción sea la de enviarlo al controlador.

- Flow-Removed

  Un mensaje flow removed se envía al controlador para informar que el flujo ha sido eliminado porque se ha vencido el tiempo de inactividad.

- Port-status

  Este tipo de mensajes informan al controlador de un cambio en el estado de un puerto o de un error que ha ocurrido en un switch, como cuando un paquete lega con una instrucción de reenvió no especificada o cuando los links se caen [12] [19].

2.2.6 Controlador SDN

El controlador es el elemento principal de una red SDN y se considera como el sistema operativo de la misma, el controlador centraliza toda la comunicación que pasa por los dispositivos y provee una visión general de la red.

Las aplicaciones que se ejecutan en el controlador determinan la manera en que los flujos se comportaran en la red, el controlador se comunica con los dispositivos a través de OpenFlow y cada elemento de la red se comunica con el controlador pidiendo instrucciones cada vez que no sepa cómo actuar frente a un determinado flujo [21] [22].

Existen múltiples controladores SDN, algunos de estos desarrollados por comunidades y otros por empresas. La diferencia principal de estos controladores es el lenguaje de programación que utilizan, pero en si realizan las misma funciones. Los cuales se describen en la tabla 2.2.

Tabla 2. 2 Controladores SDN

Controlador	Descripción	Entorno de desarrollo
NOX	❖ Fue el primer controlador desarrollado ❖ Este software es fabricado por Niciara Networks y fue desarrollado a la par del protocolo OpenFlow ❖ Está diseñado para ser instalado y configurado sobre sistemas operativos Linux, sobre todo Ubuntu en sus versiones 11.0 y 12.04	C++ Python
POX	❖ Controlador SDN que soporta OpenFlow. ❖ Tiene APIs, mapas topológicos y soporte para virtualización. ❖ Puede correr en plataformas como: Linux, Windows, Mac OS.	Python
MUL	Proyectado para redes de misión crítica con alto desempeño y confiabilidad.	C
JAXON	Controlador en Java	Java
TREMA	Desarrollado en Ruby & C	C Ruby
BEACON	❖ Controlador en Java que soporta operaciones Basadas en eventos e threads ❖ Estable	Java

	❖ Multiplataforma ❖ Código Abierto ❖ Dinámico ❖ Desarrollo rápido	
FLOODLIGHT	❖ Fue una actualización de Beacon, originalmente desarrollado por David Erickson en Stanford ❖ Fácil de configurar con dependencias mínimas ❖ Soporta una amplia gama de conmutadores tanto virtuales como físicos	Java
MAESTRO	Es un sistema operativo para controlar aplicaciones	Java
NDDI - OESS	OESS es una aplicación para configurar y controlar switches Openflow usando una interface sencilla y amigable	
RYU	Es un sistema operativo de red (NOS) que soporta Openflow	Python
NODEFLOW	Controlador escrito totalmente en JavaScript para Node.JS	JavaScript
OVS-CONTROLLER	❖ Controlador mínimo y sencillo para trabajar con Openvswitch. ❖ Viene empaquetado [12] [21].	C

2.2.7 Mininet

Mininet es un simulador de red el cual es capaz de crear redes de hosts, switches controladores y links virtuales. Mininet ejecuta una colección de equipos finales dentro de un solo núcleo de Linux y utiliza la virtualización ligera para hacer que un solo sistema parezca una red completa.

En Mininet los host, switches, links y controladores se comportan como dispositivos reales, la ventaja de mininet es que estos dispositivos son recreados usando software en lugar de hardware [23] [24] [25].

- **Ventajas de Mininet**

Mininet se presentan un sin número de ventajas que se describen a continuación:

- Rápido

  Permite crear una red en tan solo unos pocos segundos.

- Creación de topologías personalizadas

  Permite crear topologías diferentes a las existentes en mininet a través de código basado en Python.

- Capacidad de personalizar el renvió de paquetes.

  Los switches en mininet se pueden programar mediante el protocolo OpenFlow. Las redes creadas en mininet pueden ser fácilmente transferidas a switches OpenFlow reales.

- Capacidad de compartir y replicar resultados

  Se puede crear y ejecutar experimentos en mininet mediante la creación de scripts en Python.

- Mininet es un proyecto de código abierto

  Lo cual permite examinar su código y modificarlo [23] [25].

♦ **Limitaciones de mininet**

A pesar de que mininet es una herramienta muy completa pero posee ciertas limitaciones las cuales se describen a continuación.

- Impone límites de recursos debido a que corre en un solo sistema, es decir si un servidor tiene 3Ghz de CPU y puede cambiar alrededor de 3 Gbps de tráfico simulado, tendrán que ser equilibrado y compartido entre los host virtuales y switches.

- Mininet utiliza un solo kernel de Linux para todos los hosts, esto significa que no se puede ejecutar software que depende de BSD, Windows u otro sistema operativo aunque se puede correr en una máquina virtual.

- No se puede realizar cambios en el controlador que tiene por defecto Mininet, si se necesita enrutamiento de conmutación, se tendrá que desarrollar en un controlador que posea las características que se requiera [23] [25].

♦ **Comandos de mininet**

Exit

Sale del modo mininet

Xterm [nodo]

Abre un terminal del nodo especificado

Link [nodo1][nodo2][up o down]

Crea o elimina un link entre dos hosts

Pingall

Prueba conectividad entre todos los nodos

Help

Muestra la lista de comandos disponibles

Sudo wireshark &

Inicia el analizador de trafico Wireshark

Nodes

Muestra los nodos de la red

Net

Muestra información de los links

Dump

Muestra información acerca de los nodos

Sudo mn –c

Limpia residuos de las redes creadas [26] [27].

- **Topologías estándar de mininet**

  - Topología mínima

    Esta tecnología consta de un controlador, un switch OpenFlow y dos host conectados al switch. La cual se puede crear mediante el siguiente comando

    $ sudo mn --topo minimal

    En la figura 2.11 Se muestra esta topología [28]

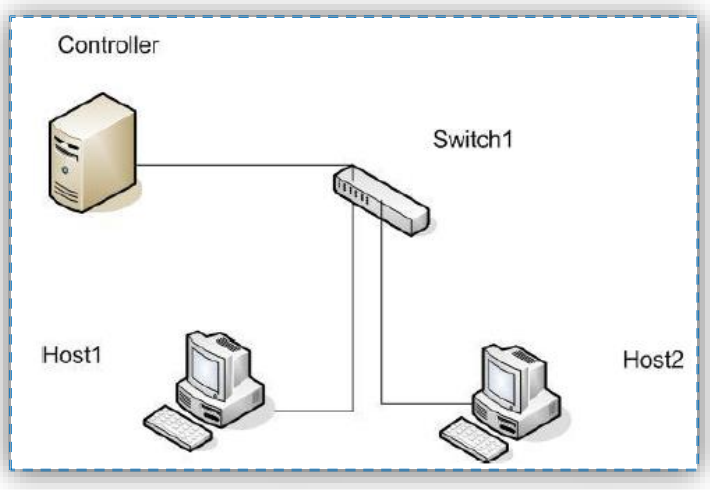


Fig 2. 11 Topología mínima

Fuente: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf

- Topología single

  Esta topología se crea mediante el siguiente comando.

  $ sudo mn --topo single, n

  Dónde:

  [n] es el número de hosts conectados a un solo switch

  Esta topología se muestra en la figura 2.12 [28]

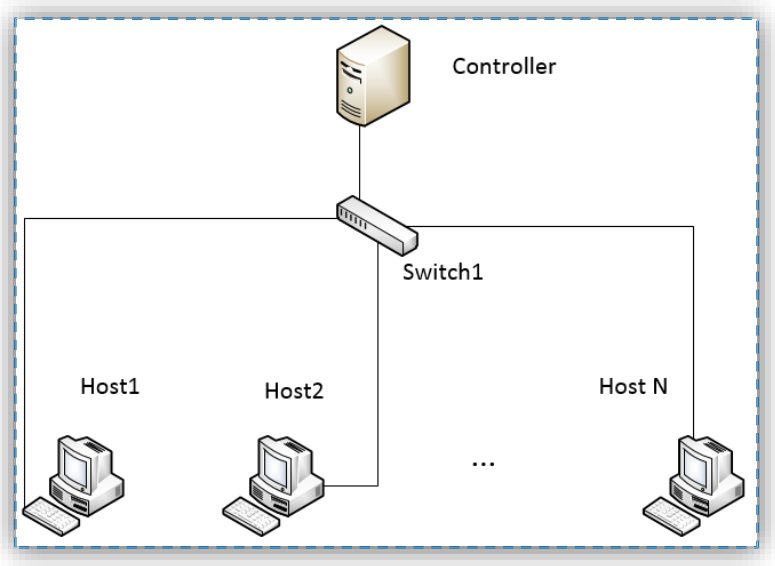


Fig 2. 12 Topología Single

Fuente: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf

- Topología linear

  Esta topología se crea mediante el comando.

  $ sudo mn – topo linear,n

  Donde [n] es el número de switches conectados entre sí, y cada switch posee un solo host conectado a él.

  En la figura 2.13 se muestra esta topología [28]

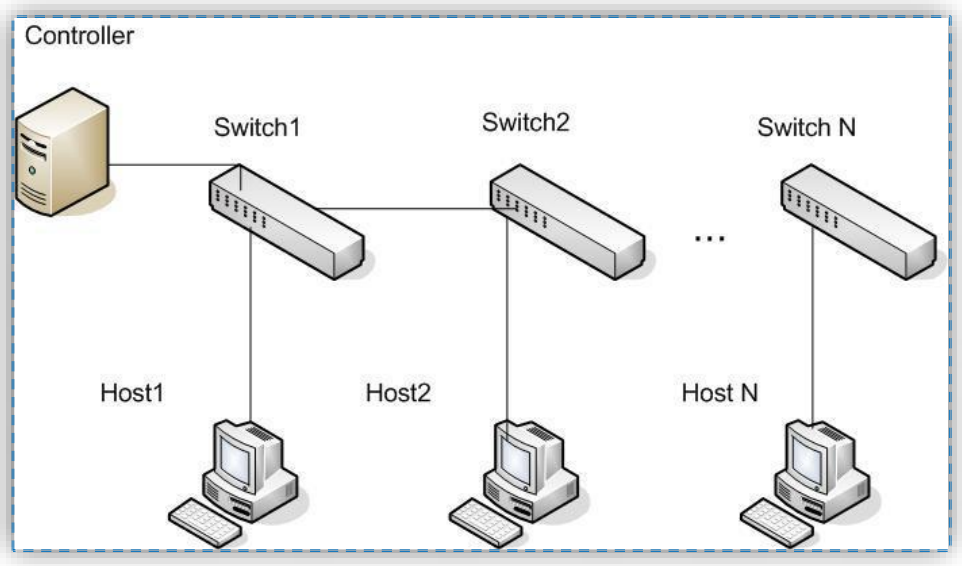


Fig 2. 13 Topología linear

Fuente: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf

- Topología árbol

  Este tipo de topología se puede implementar de mediante el siguiente comando.

  $ sudo mn –topo tree, depht=N,fanout =M

  Donde:

  N = es el número de niveles de switches interconectados

  M = es el número de host conectados a casa switch

  Un ejemplo de este tipo de topología se puede observar en la figura 2.14 [28]

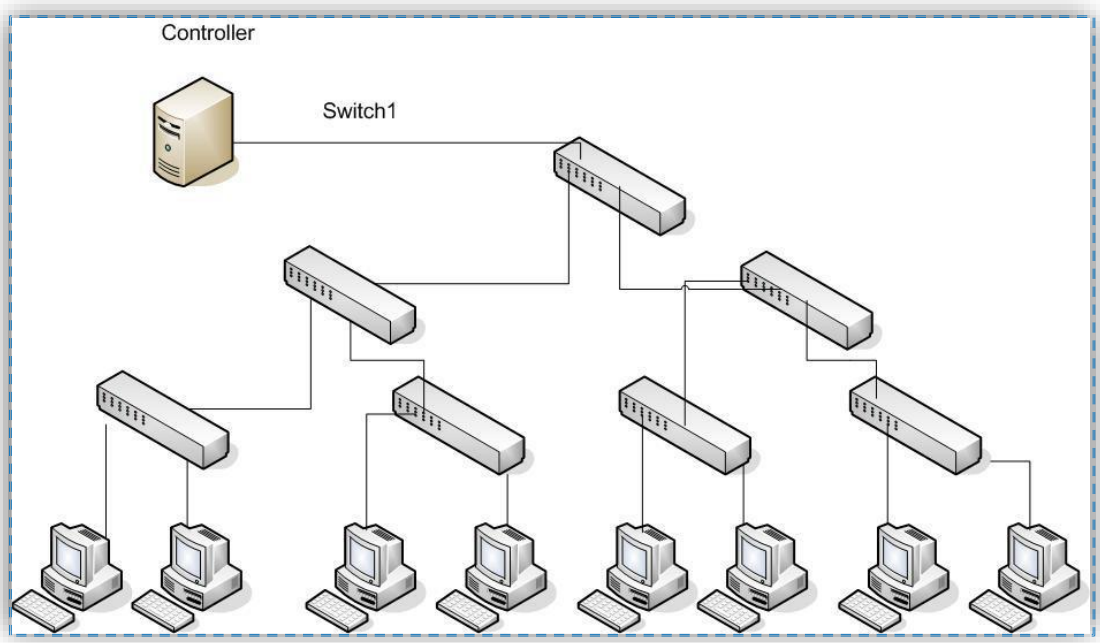


Fig 2. 14 Topología tipo árbol

Fuente: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf

2.2.8 DPCTL

Dpctl es una utilidad por línea de comandos que ofrece Mininet para permitir en intercambio de mensajes OpenFlow básicos con los switches.

La principal característica de dpctl es que permite comunicarse directamente con el switch sin la necesidad de un controlador SDN, mediante este comando se puede realizar varias tareas tales como:

- Ver las estadísticas de los flujos y puertos del switch
- Visualizar las entradas de flujo contenidas en el switch
- Anadir y Eliminar entradas de flujo

A continuación se describe los comandos más utilizados que ofrece DPCTL.

show *switch*

Imprime en consola información del plano de datos del switch incluyendo información sobre sus tablas de flujo y puertos.

status *switch* [*key*]

Imprime en la consola de una serie de pares clave-valor que informan el estado del switch. Si la ***key*** es especificada, sólo los pares clave-valor cuyos nombres clave empiezan con la ***key*** se imprimen. Si la ***key*** es omitida, todos los pares clave-valor se imprimen.

show-protostat *switch*

Imprime la información estadística del protocolo OpenFlow del switch

dump-tables *switch*

Imprime en consola estadísticas de cada flujo de las tablas usadas por el datapath del switch

dump-ports *switch* [*número de puerto*]

Imprime en consola estadísticas de cada interfaz supervisada por el ***switch***. Si se especifica el número de puerto, se imprimen estadísticas sólo para la interfaz correspondiente al número de puerto.

mod-port *switch netdev acción*

Modifica las características de una interfaz monitoreada por el switch, ***netdev*** hace referencia al número de puerto OpenFlow asignado o al nombre del dispositivo, por ejemplo, **eth0**. La ***acción*** puede ser uno cualquiera de las siguientes:

Up

Activa la interfaz. Esto es equivalente a “ifconfig up” en un sistema Linux.

Down

Desactiva la interfaz. Esto es equivalente a “ifconfig down” en un sistema Linux.

Flood

Cuando una acción ***flood*** es especificada, el tráfico se envía a todas las interfaces del switch excepto por la cual ingresaron los paquetes.

Noflood

Cuando una acción de inundación es especificada, esta interfaz no podrá enviar tráfico fuera de ella. Esto es principalmente útil para evitar bucles cuando el protocolo spanning tree no está en uso.

dump-flows *switch* [*flujos*]

Imprime en la consola todas las entradas de flujo contenidas en la tabla de flujos del switch. Si los ***flujos*** se omiten, se recuperan todos los flujos, excepto los de emergencia

add-flows *switch flujo*

Añade una nueva entrada de flujo en la tabla de flujos del switch.

mod-flows *switch flujo*

Modifica las acciones de las entradas de flujo que concuerden con los parámetros definidos en la opción ***flujo***.

del-flows *switch*

Elimina las entradas de flujo contenidas dentro de la tabla de flujo del switch.

Los siguientes comandos supervisan y controlan la configuración de la cola de salida de un conmutador OpenFlow, si el conmutador es compatible con este tipo de operaciones. Después de que una cola es creada mediante la operación add o modify, la acción OpenFlow ***enqueue*** puede especificar a los paquetes directo a una cola en particular.

Las colas se asocian con puertos específicos (por lo que el mismo id de la cola se pueden utilizar en diferentes puertos y esto referencia a diferentes colas).

add-queue *puerto del switch q-id* [*el ancho de banda*]

Conecta el Switch y añade una cola de salida identificado como ***q-id*** para el ***puerto***. Si se especifica, ***el ancho de banda*** indica el ancho de banda mínimo de garantía para la cola y se especifica en décimas de por ciento. Esta es la única característica de la cola que se puede configurar.

mod-queue *puerto del switch q-id ancho de banda*

Conecta el switch y modificar la configuración de ancho de banda para una cola de salida identificado como ***q-id*** para el puerto. La cola no tiene por qué haber sido creado con el comando **add-cola** previamente. El parámetro de ***ancho de banda*** indica la garantía mínima de ancho de banda para la cola y se especifica en décimas de por ciento.

del-queue *puerto del switch q-id*

Elimina una cola de salida identificado como *q-id* para el puerto que había sido creado por **add-queue** o **mod-queue**.

dump-queue *switch* [*puerto*[*q-id*]]

Muestra la configuración actual de la cola. Un puerto puede ser especificado. Si es así, una ***queue-id*** también puede ser especificado.

Campos y sintaxis de los flujos

in_port = *Puerto de ingreso del flujo*

Coincide con el número de puerto físico. Los puertos del conmutador están numerados como muestra el comando **dpctl show**.

dl_vlan = *vlanID*

Coincide con la etiqueta de edificador de la VLAN. Se especifica el valor **0xffff** como *VLAN* para que coincida con los paquetes que no están etiquetados con una LAN virtual; de lo contrario, se especifica un número entre 0 y 4095, como el ID de VLAN de 12 bits para que coincida.

dl_src = *dirección mac de origen*

Este parámetro se compara con la dirección ***mac*** de origen, que debe especificarse como 6 pares de dígitos hexadecimales delimitados por dos puntos, por ejemplo, **00: 0A: E4: 25: 6B: B0**.

dl_dst = *dirección mac de origen*

Coincide con dirección de destino ***mac***.

dl_type = *ethertype*

Este parámetro define el tipo de protocolo Ethernet ***ethertype***, que debe especificarse como un número entero entre 0 y 65535, ya sea en decimal o como un número hexadecimal con el prefijo **0x**, por ejemplo **0x0806** para que coincida con los paquetes ARP.

nw_src = *dirección ip* [*/ máscara de red*] *de origen*

Compara la dirección IPv4 de origen, la cual debe especificarse como una dirección IP o nombre de host, por ejemplo, **192.168.1.1** o **www.example.com**. La máscara de red es opcional

y permite comparar solo un prefijo de dirección IPv4. Se puede especificar como (por ejemplo **192.168.1.0/255.255.255.0**) o como (por ejemplo **192.168.1.0/24**).

nw_dst = *dirección ip* [/ *máscara de red*]

Compara la dirección IPv4 de destino.

nw_proto = *proto*

Compara con el tipo de protocolo IP ***proto***, que debe especificarse como un número decimal entre 0 y 255, por ejemplo, 6 para que coincida con los paquetes TCP.

nw_tos = *TOS / DSCP*

Compara el **TOS / DSCP** (sólo 6 bits, no modifica los 2-bits reservados para uso futuro) campo de encabezado **IPv4 *TOS / DSCP***, que debe especificarse como un número decimal entre 0 y 255, ambos inclusive.

tp_src = *puerto*

Compara con los puertos de origen UDP o TCP, que debe especificarse como un número decimal entre 0 y 65535, por ejemplo, 80 para que coincida con los paquetes procedentes de un servidor HTTP.

tp_dst = *puerto*

Compara con los puertos de destino UDP o TCP.

ICMP_TYPE = *tipo*

Compara los mensajes ICMP con el parámetro ***tipo***, que debe especificarse como un número decimal entre 0 y 255.

ICMP_CODE = *código*

Compara los mensajes ICMP con el parámetro ***código***.

Las siguientes anotaciones también están disponibles:

ip

Igual que **dl_type = 0x0800**.

icmp

Igual que **dl_type = 0x0800, nw_proto = 1**.

tcp

Igual que **dl_type = 0x0800, nw_proto = 6**.

UDP

Igual que **dl_type = 0x0800, nw_proto = 17**.

Arp

Igual que **dl_type = 0x0806**.

ACCIONES

Los comandos **add-flow** y **add-flows** requieren de campos adicionales que son las acciones que se pretende dar a determinada entrada de flujo.

acciones **=** *objetivo* [, *objetivo*...]

Especifica una lista separada por comas de las acciones a tomar en un paquete cuando la entrada de flujo coincide. El ***objetivo*** puede ser un número de puerto decimal designar el cual designa el número de puerto físico del switch, o una de las siguientes palabras clave:

output: *puerto*

Envía al paquete hacia el puerto especificado por el parámetro ***puerto***.

enqueue: *puerto* : *q-id*

Coloca al paquete en una cola especificada por el ***q-id*** o id de la cola en el puerto especificado por el parámetro ***puerto***.

normal

Temas del paquete a la normalidad el procesamiento de L2 / L3 del dispositivo. (Esta acción no se lleva a cabo por todos los switches OpenFlow.)

flood

Envía paquetes a todos los puertos físicos del switch excepto en el que fue recibido el paquete y los puertos en los que las inundaciones están deshabilitadas.

all

Envía paquetes a todos los puertos físicos del switch excepto al puerto en el que fue recibido el paquete.

controlador: *max_len*

Envía el paquete al controlador OpenFlow como un mensaje "packet-in". Si ***max_len*** es un número, entonces se especifica el número máximo de bytes que deben ser enviadas. Si ***max_len*** es **ALL** o se omite, se envía todo el paquete.

mod_vlan_vid: *vlan_vid*

Modifica la ID de VLAN del paquete. La etiqueta de VLAN se agrega o se modifica según sea necesario para que coincida con el valor especificado. Si se añade la etiqueta VLAN, se utiliza una prioridad de cero.

mod_vlan_pcp: *vlan_pcp*

Modifica la prioridad VLAN en un paquete. La etiqueta de VLAN se agrega o se modifica según sea necesario para que coincida con el valor especificado. Los

valores válidos están comprendidos entre 0 (el más bajo) y 7 (el más alto). Si se añade la etiqueta VLAN, se utiliza una vid de cero.

mod_dl_dst: *dst_mac*

Modifica la dirección MAC de destino de un paquete, por ejemplo, las acciones mod_dl_dst: 12: 34: 56: 78: 9a: bc

mod_dl_src: *src_mac*

Modifica la dirección MAC de origen en un paquete, por ejemplo, las acciones mod_dl_src: 12: 34: 56: 78: 9a: bc

mod_nw_tos: *TOS / DSCP*

Modifica el **TOS / DSCP** (sólo 6 bits, no modifica los 2-bits reservados para uso futuro) campo de la cabecera IPv4 en un paquete.

Los comandos **add-flow**, **add-flows** y **del-flows**, soportan un campo opcional adicional:

priority = *valor*

Establece la prioridad del flujo que se añade o se elimina al ***valor***, que debe ser un número entre 0 y 65535. Si no se especifica este campo, el valor predeterminado es 32768.

Los comandos **add-flow** y **add-flows** comandos admiten campos opcionales adicionales:

idle_timeout = *segundos*

Hace que el flujo expire después de un cierto número de segundos de inactividad. Si se establece este valor como 0 impide que el flujo expire por inactividad. El valor predeterminado es de 60 segundos.

hard_timeout = *segundos*

Hace que el flujo expire después de un cierto número de segundos, independientemente de la actividad. Un valor de 0 (por defecto) provoca que el flujo tenga un plazo de caducidad prolongado [29] [30].

CAPÍTULO III

METODOLOGÍA

3.1 Modalidad de la Investigación

Se realizó una investigación bibliográfica, documental mediante libros, revistas científicas, publicaciones, papers y publicaciones electrónicas lo cual nos permitió profundizar en el tema y adquirir el conocimiento necesario que ayude en el desarrollo del proyecto.

3.2 Población y Muestra

Para el desarrollo del proyecto no se contó con población y muestra debido a que es un proyecto de investigación y la información se recolectó de fuentes bibliográficas.

Recolección de la Información

Para la realización del proyecto se recolectó información a través de libros, internet, papers, revistas científicas, la guía del profesor tutor, etc.

3.3 Procesamiento y Análisis de datos

Para la realización del procesamiento y análisis de los datos se llevaron a cabo una serie de procesos descritos a continuación.

- Recolección de datos mediantes libros y documentación electrónica
- Revisión de la información con el objetivo de encontrar la información necesaria para la realización del proyecto
- Lectura de artículos relacionados con el tema de investigación
- Interpretación de resultados
- Planteamiento de la propuesta de solución

3.4 Desarrollo del Proyecto

Para el desarrollo del proyecto se llevaron a cabo los siguientes procesos

- Determinación de las limitaciones que presentan las redes actuales.
- Análisis de la información recolectada para el desarrollo de redes (SDN).
- Análisis de las ventajas que presentan las redes definidas por software.
- Análisis de los controladores de libre distribución para la creación de redes definidas por software.
- Configuración de los controladores necesarios para el desarrollo de una red (SDN).
- Simulación del prototipo de la red (SDN) en el software de simulación Mininet.
- Creación de módulos para los controladores de la red.
- Implementación de los controladores y módulos en la simulación del prototipo de la red.
- Realización de pruebas del prototipo virtual.
- Evaluación del prototipo de red virtual.
- Implementación del prototipo de la red utilizando conmutadores físicos.
- Realización de prueba y evaluación del prototipo de red físico.

CAPÍTULO IV

DESARROLLO DE LA PROPUESTA

En la presente investigación se plantea una arquitectura personalizada la cual consta de dos switches híbridos y un switch habilitado, con el objetivo de comprobar el correcto funcionamiento de la red tanto con dispositivos diseñados para OpenFlow como con dispositivos habilitados de bajo costo. Lo cual demuestra que esta nueva arquitectura es posible implementarla sin la necesidad de reemplazar totalmente la infraestructura de red existente. En la figura 4.1 se muestra la topología de red a realizarse.

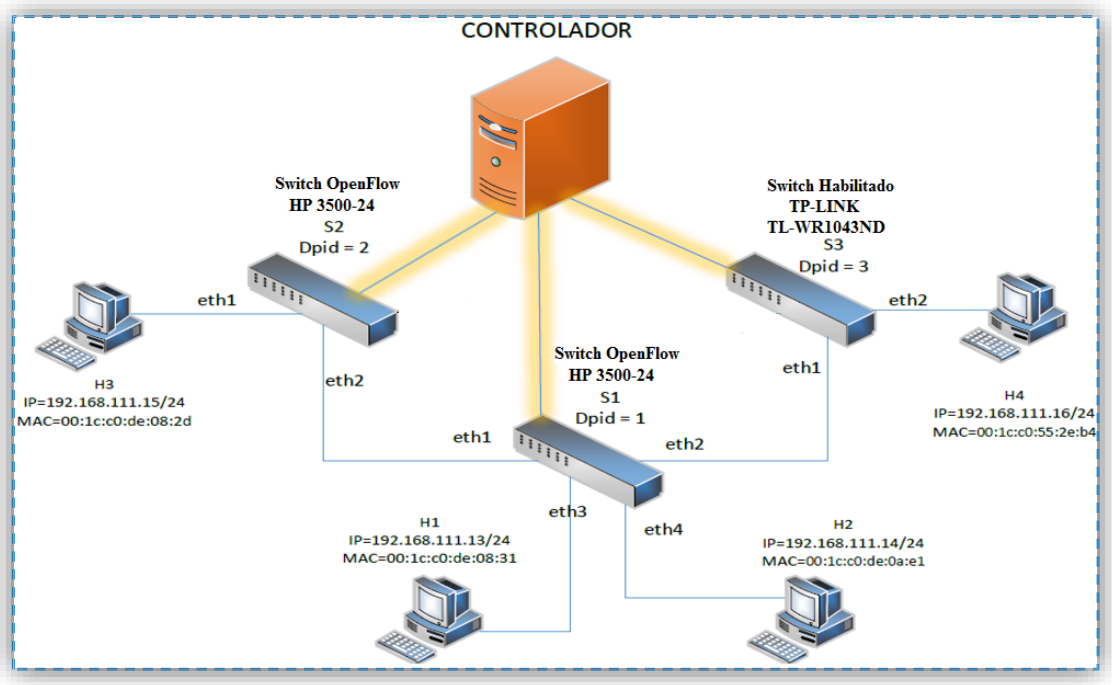


Fig. 4. 1 Topología personalizada del prototipo de red SDN.

4.1 Análisis de los controladores

En el capítulo II se analizó las redes definidas por software SDN, determinando limitaciones en la infraestructura de red actual, debido a los nuevos requerimientos que han surgido, y la gran demanda de información que se maneja en la actualidad. Se ha planteado la idea de migrar hacia una nueva infraestructura de red con SDN.

Estas redes permiten definir la ruta que seguirá un determinado flujo de datos a través de la red. A diferencia de la arquitectura de red convencional en donde el dispositivo es el encargado de tomar esta decisión, logrando con esto tener un control casi total y personalizado [31].

Para el desarrollo de una infraestructura SDN se necesita de varias partes tales como:

El controlador SDN.- Siendo este la parte fundamental de una red SDN

El protocolo OpenFlow.- Permite acceder a las tablas de flujo contenidas en los switches con el objetivo de manipular sus entradas de flujo.

Un canal seguro.- Hace posible conexión entre el controlador y el protocolo OpenFlow.

Switch OpenFlow o Habilitados para OpenFlow.- Establecen la comunicación entre los dispositivos finales.

Software de simulación Mininet.- Establece topologías red definidas por software para su simulación.

La parte más importante de una infraestructura SDN es el controlador. En la tabla 4.1 se ha tomado como referencia 5 de los controladores más utilizados para la realización de este tipo de redes, ya que existen un sin número de controladores SDN solamente se han escogido los más relevantes y los que poseen un mayor número de aplicaciones para el desarrollo de módulos SDN.

Tabla 4. 1 Cuadro comparativo de los controladores SDN

Características	**CONTROLADORES**				
	NOX	**POX**	**BEACON**	**FLOODLIGHT**	**RYU**
Lenguaje de desarrollo	C++	Python	Java	Java	Python
Lenguajes soportados por el controlador	C, C++, Python	Python	Java	Java, Python	Python
Facilidad de instalación	Regular	Fácil	Fácil	Fácil	Difícil
Simplicidad	Regular	Regular	Fácil	Fácil	Difícil
Soporte	Malo	Bueno	Regular	Bueno	Malo

Módulos para el desarrollo de aplicaciones	Malo	Bueno	Regular	Bueno	Regular
Tiene una comunidad activa	No	Si	Si	Si	Si
Documentación	Mala	Mala	Buena	Buena	Buena
Provee REST API	No	No	No	Si	Si (Básica)
Tiene Interfaz Gráfica	Python+QT4	Python+QT4, Web	Web	Java, Web	Web
Soporta bucles en la topología	No	No	No	Si	No
Provee REST API	No	No	No	Si	Si (Básica) [6] [32]

Al realizar el análisis de la tabla 4.1 se determinó la utilización de los controladores Beacon y Floodlight por ser los que presentan las mejores características en cuanto al lenguaje de programación, soporte, documentación, y simplicidad. Además de proveer una gran cantidad de herramientas necesarias para el desarrollo del proyecto.

4.2 Análisis de los switches OpenFlow

Debido a que existen una gran variedad de switches OpenFlow manufacturados, disponibles comercialmente. En la tabla 4.2 se realiza una comparativa de estos con el objetivo de determinar la mejor alternativa para el prototipo red [21].

Tabla 4. 2 Lista de switches OpenFlow Comerciales

Fabricantes	Modelo del switch	Versión de OpenFlow	Precios referenciales
Brocade	NetIron CES 2000 Series, CER 2000,	1.0	Brocade NetIron CES-2024C Compact Carrier Eethernet Switch $8,913.86 Brocade Communications - BR-CER-2024F-4X-RT-DC - Brocade NetIron CER 2024F-4X Router - 4 Ports - 28 Slots - Rack-mountable $24,250.54
Hewlett Packard	3500-24/3500yl, 5400zl,6200zl, 6600, 8200zl	1.0	Procurve Switch 3500YL 24G-PWR Edge $1,750.00
BM	RackSwitch G8264, G8264T	1.0	IBM RackSwitch G8264F 3 Layer Switch - 52 Slot (7309-64F) $23,620.94
Juniper	EX9200 Programmable switch	1.0	
NEC	PF5240, PF5820	1.0	
Pica8	P-3290, P-3295, P -3780, P3920	1.2	
Pronto	3290 and 3780	1.0	
Broadcom	BCM56846	1.0	

Extreme Networks	BlackDiamond 8K, Summit X440, X460, X480	1.0	Extreme Networks - SFP (mini-GBIC) transceiver module - LC multi-mode - for BlackDiamond 8800, Summit X450-24, X450a-24, X450a-48, X450e-24, X450e-48 Principio del formulario Final del formulario $340.73
Netgear	GSM7352Sv2	1.0	Netgear ProSafe GSM7352Sv2 Gigabit L3 Managed Stackable Switch. 48PORT PROSAFE 10/100/1000 GIGABIT L3 MANAGED. $1,000.00
Arista	7150, 7500, 7050 series	1.0	

Fuente:http://eventos.redclara.net/indico/getFile.py/access?contribId=0&resId=4&materialId=slides&confId=197

Al realizar el análisis de la tabla 4.2 se determinó la utilización de los switches Hewlett Packard 3500. Debido a que es un dispositivo de excelentes características, precio y se encuentra en el inventario del departamento de investigación de la Universidad Técnica de Ambato.

4.3 Análisis de los dispositivos de bajo costo habilitados para OpenFlow

Existen dispositivos de bajo costo los cuales pueden ser habilitados para que funcionen con el protocolo OpenFlow. En la tabla 4.3 se muestra una lista de dispositivos compatibles para habilitación del protocolo OpenFlow [33].

Tabla 4. 3 Listado de switches OpenFlow de bajo costo

Dispositivo	Descripción	Precio referencial
Router TP-Link TL-WR1043ND versión 1.11	Router que consta de un procesador potente (procesador Atheros AR9132@400Mhz) compatible con la versión Attitude Adjustment 12.09 de Open Wrt.	TP-LINK TL-WR1043ND V2 Wireless N300 Gigabit Router, 300Mbps, USB port for Storage, 3 Detachable Antennas, Speed Boost up to 450Mbps, WPS Button $50,00
Linksys WRT54GL – OpenWRT de Cisco	• Un puerto Internet RJ-45 10/100 Mbps • Cuatro puertos Ethernet RJ-45 10/100 Mbps • Todos los puertos soportan intercambio de pines por software MDI/MDI-X • CPU Broadcom BCM5352 @200MHz • Memoria RAM de 16 Mb • Memoria Flash de 4 Mb	Linksys WRT54GL Wi-Fi Wireless-G Broadband Router $50,00

	• Soporta actualización de firmware OpenWRT • WLAN: soporta protocolos IEEE 802.11b y g	
TP-LINK TL-WR1043ND (v1.7)	**Chipset** Atheros	$70,00
TP-LINK TL-WR1043ND (v1.8)	**Chipset** Atheros	$80,00
Broadcom Genérico	**Chipset** Broadcom BCM47xx	

Fuente: http://archive.openflow.org/wk/index.php/OpenFlow_1.0_for_OpenWRT

Tras realizar el análisis de la tabla 4.3 se determinó la utilización de un router TP-LINK TL-WR1043ND versión 2.1 ya que existe un gran número de infraestructuras SDN realizadas con este tipo de dispositivos. Se plantea además utilizar la versión 2.1 de este router a manera de investigación ya que no se ha habilitado OpenFlow en un router de esta versión.

4.4 Configuraciones

4.4.1 Controlador

En este ítem se describe el proceso de instalación y configuración de los controladores SDN para el desarrollo de módulos SDN.

Cabe recalcar que desde este punto en adelante todos estos procesos se realizaron en Ubuntu 14.04 ya que uno de los requisitos para poder utilizar los controladores

es usar un sistema operativo Ubuntu que puede ser desde la versión 10.04 en adelante.

- **Instalación y configuración de Beacon**

Prerrequisitos

Java 6 JDK y JRE

Eclipse **RCP** and RAP Developers v3.7.0

- Proceso de instalación

1. Instalar el comando git para descargar los archivos tanto de Beacon como de Floodlight. Para instalar esta herramienta se utilizó el siguiente comando:

   Sudo apt-get install git

2. Crear una carpeta denominada git necesaria para descargar los archivos de Beacon dentro de la misma, utilizando los comandos:

   $ mkdir git

   $ cd git

   $ git clone git://gitosis.stanford.edu /openflowj.git

   $ git clone git://gitosis.stanford.edu/beacon.git

- Configuración del controlador en eclipse.

Para la configuración de controlador Beacon en eclipse se deben realizar los siguientes pasos:

1. Crear un nuevo espacio de trabajo. Tal como se muestra en la figura 4.2.

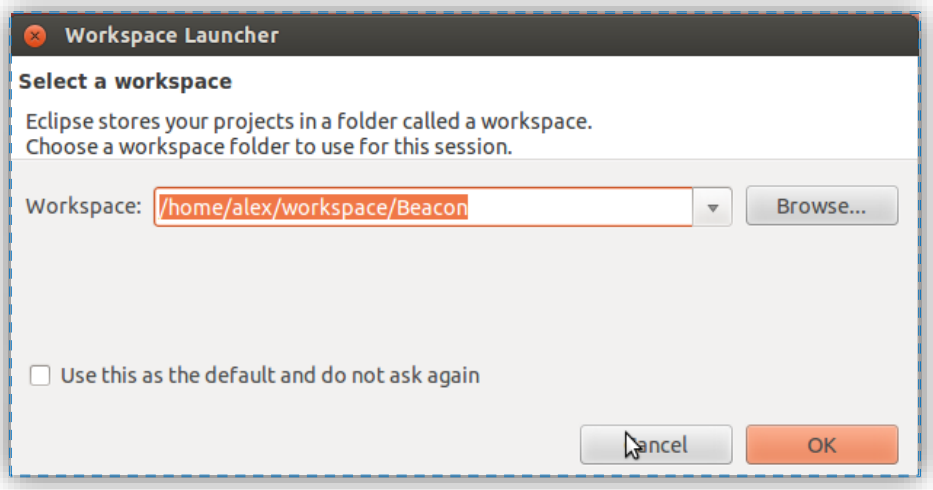


Fig. 4. 2 Creación del espacio de trabajo en eclipse

2. Establecer el compilador al nivel 1.6, para lo cual dirigirse a: ***Window/ Preferences/Java/ Compiler***. Tal como se observa en la figura 4.3.

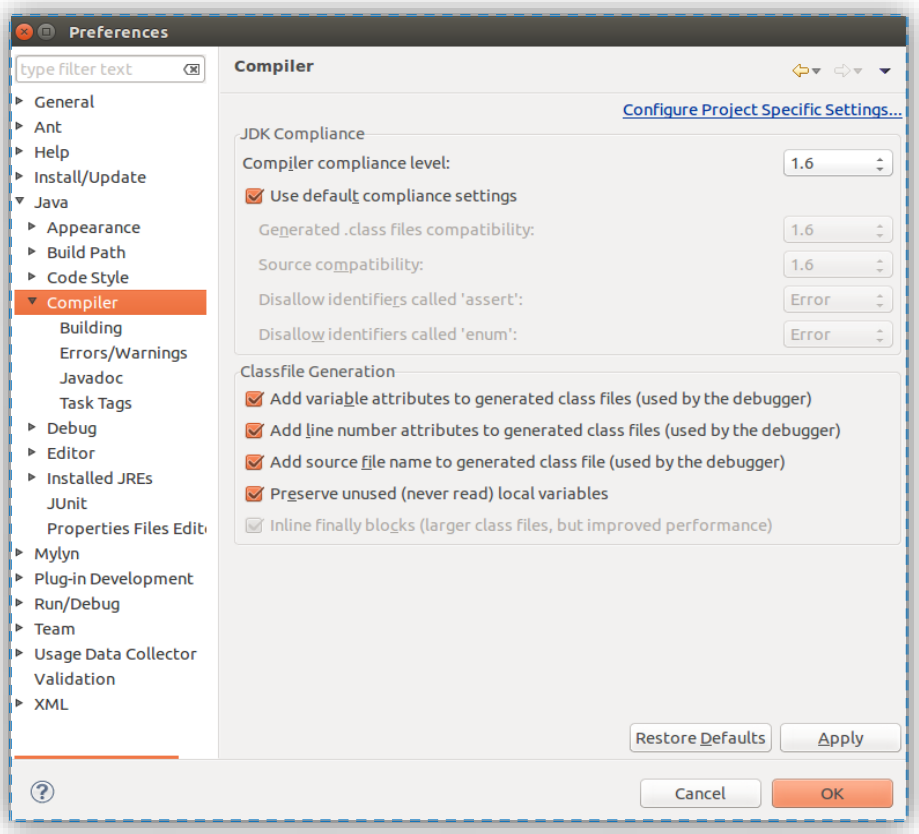


Fig. 4. 3 Establecimiento del compilador al nivel 1.6

- Importación de Beacon a eclipse.

1. Importar los archivos de Beacon a eclipse y configurarlo. Para lo cual dirigirse a: ***File/Import/General/ Existing Projects into Workspace***. Como se muestra en la figura 4.4.

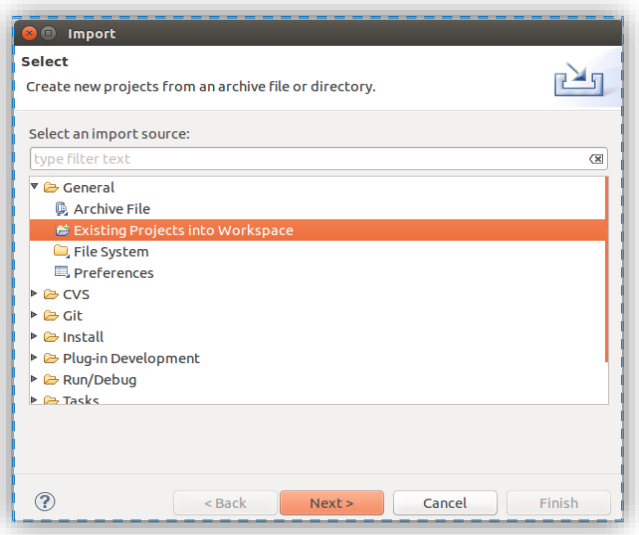


Fig. 4. 4 Importación de Beacon en eclipse

La figura 4.5 indica la selección e importación de la carpeta openflowj. La misma que contiene las librerías de funcionamiento del protocolo OpenFlow.

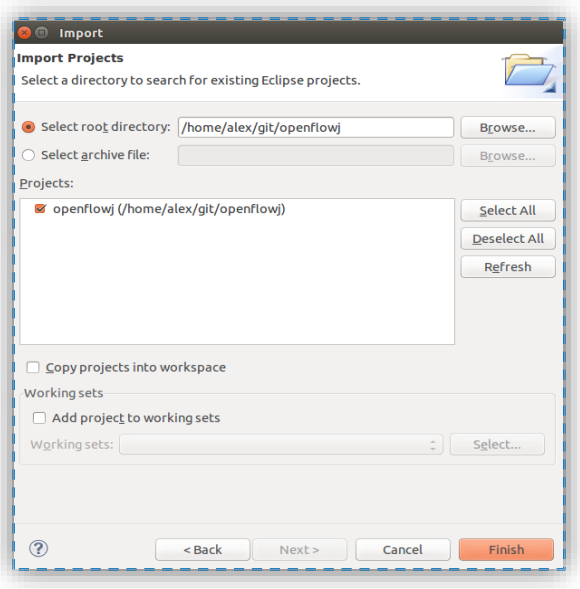


Fig. 4. 5 Importación de las librerías del protocolo OpenFlow

2. Importar la carpeta Beacon de la misma forma que se realizó en el paso anterior. En la figura 4.6 se muestra este proceso.

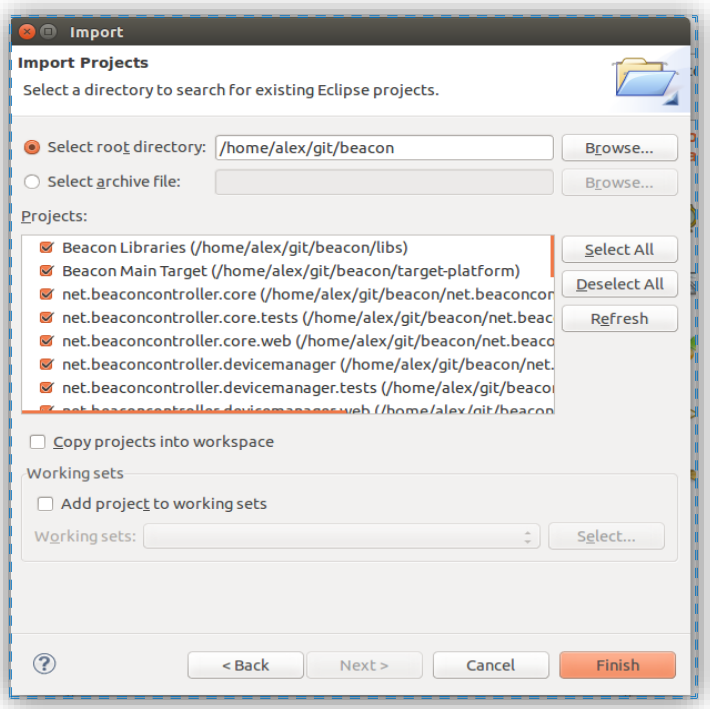


Fig. 4. 6 Importación del protocolo Beacon

3. Abrir el proyecto Beacon Main Target, dando doble click en archivo main.target.

4. Esperar que el proceso ***Resolving Target Definition*** se complete. (Ver figura 4.7).

5. Actualizar los plugins de Beacon presionando el link ***Set as Target Platform***.

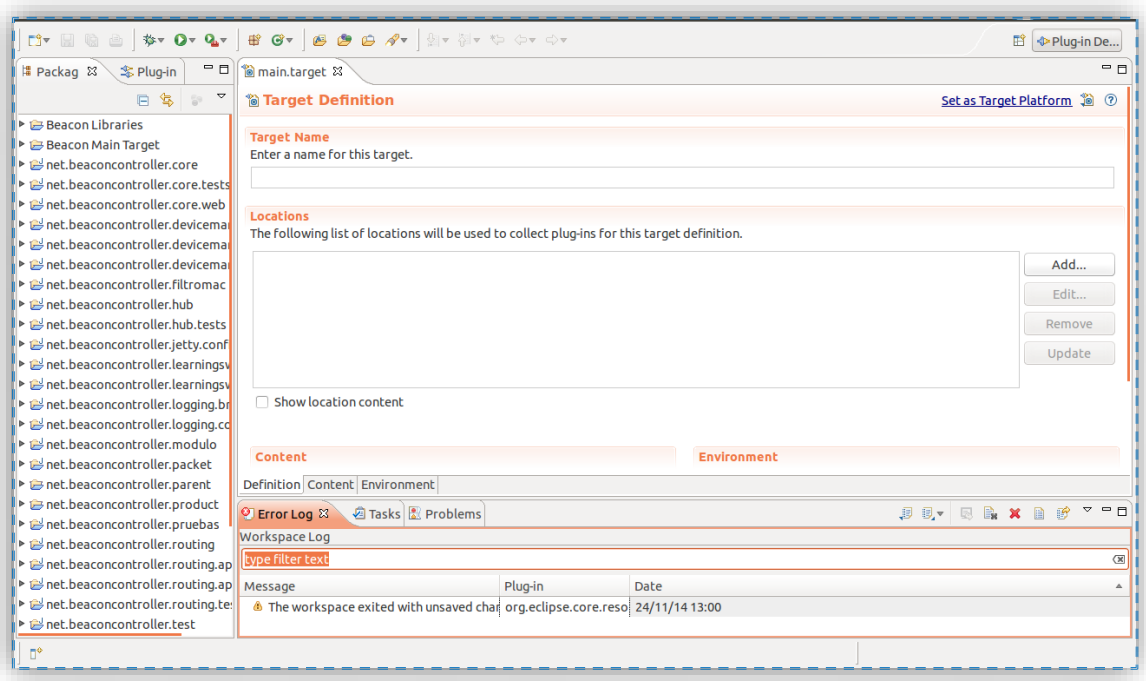


Fig. 4. 7 Resolución del proceso Main Target Definition

6. Importar el archivo ***beacon_style_settings.xml***, para ello dirigirse a: ***Windows/Preferences*** y seleccionar ***Java/Code Style/Formatter***. Tal como se muestra en la figura 4.8

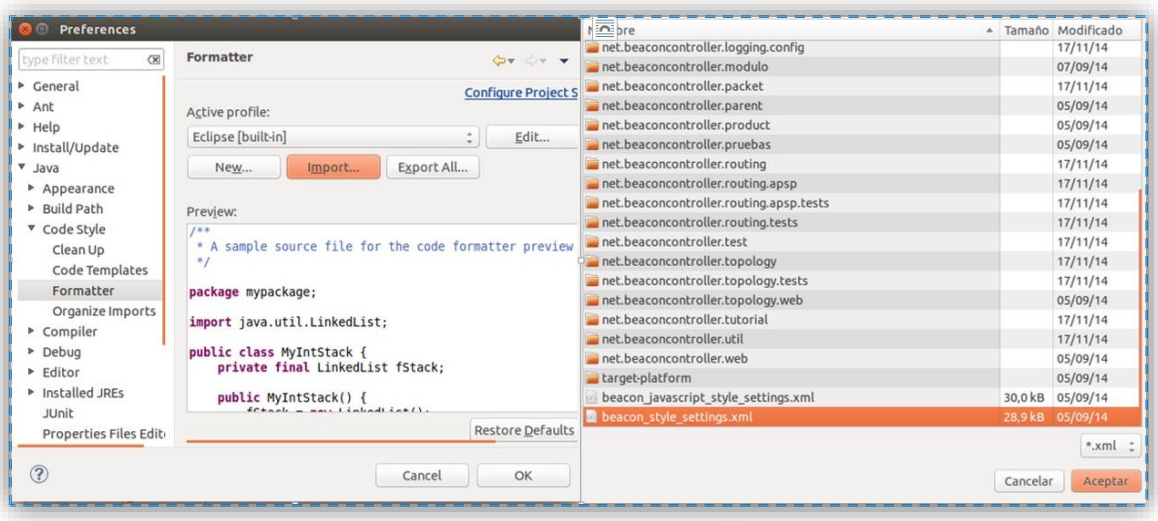


Fig. 4. 8 Importación del archivo beacon_style_settings.xml

- Ejecución de Beacon

  Para ejecutar Beacon por primera vez es necesario seguir pasos

  1. Dirigirse a Debug Configurations
  2. Desplegar el menú OSGi Frameworks y seleccionar Beacon
  3. Dar un click en debug. Tal como se muestra en la figura 4.9 [34].

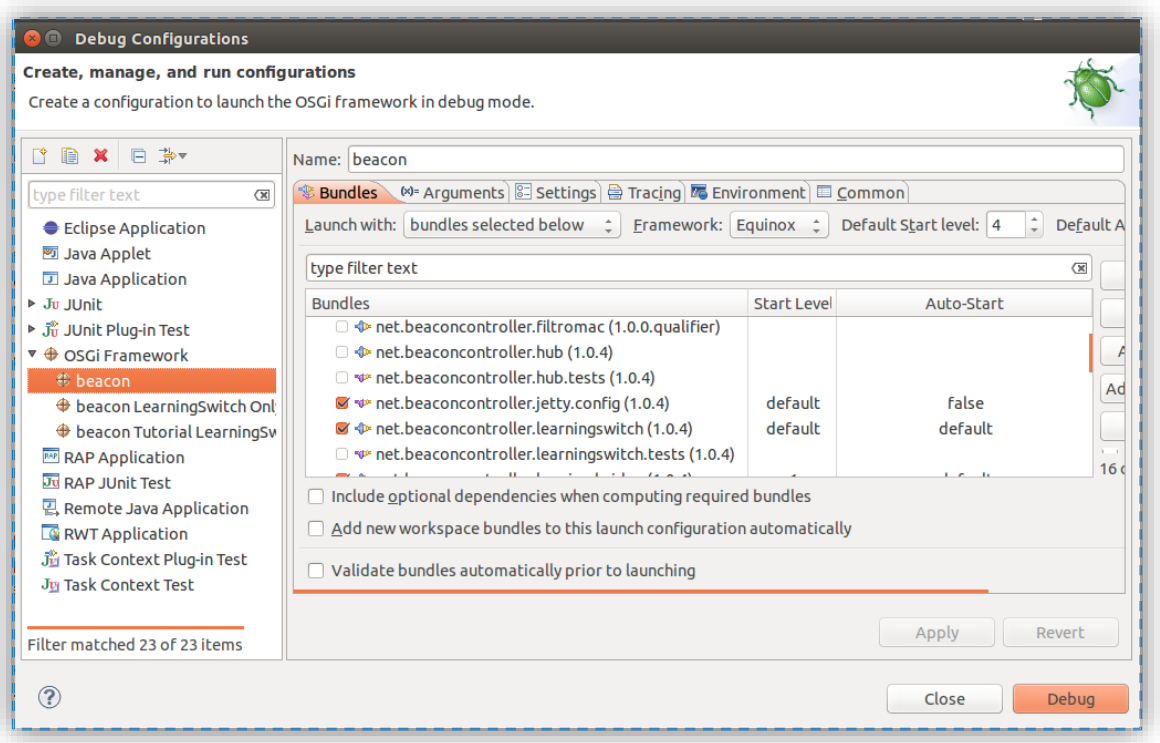


Fig. 4. 9 Establecimiento de la configuración inicial de Beacon

- **Instalación y configuración de Floodlight**

Prerrequisitos

Ubuntu 10.04 o superior

Instalar JAVA, Ant también se puede instalar eclipse aunque es opcional debido a se puede utilizar eclipse portátil.

Instalar el comando git necesario para descargar los archivos necesarios para la configuración de los controladores

El proceso de instalación y configuración de Floodlight es muy similar al de Beacon, por lo cual se describirá solamente las partes más importantes.

- Descargar los archivos del controlador Floodlight

  Para descargar los archivos del controlador Floodlight se utilizaron los siguientes comandos:

  $ git clone git://github.com/floodlight/floodlight.git

  $ cd floodlight

  $ ant eclipse

- Crear el FloodlightLaunch para la ejecución de Floodlight

  Para ejecutar Floodlight es necesario seguir los siguientes pasos:

  1. Click Run/Run Configurations

  2. Click Derecho en Java Application/New

  3. Establecer el nombre del nuevo archivo como: **FloodlightLaunch**

  4. Establecer el nombre del proyecto como: **floodlight**

  5. Establecer el Main como: net.floodlightcontroller.core.Main

  6. Click Apply [35], [36]

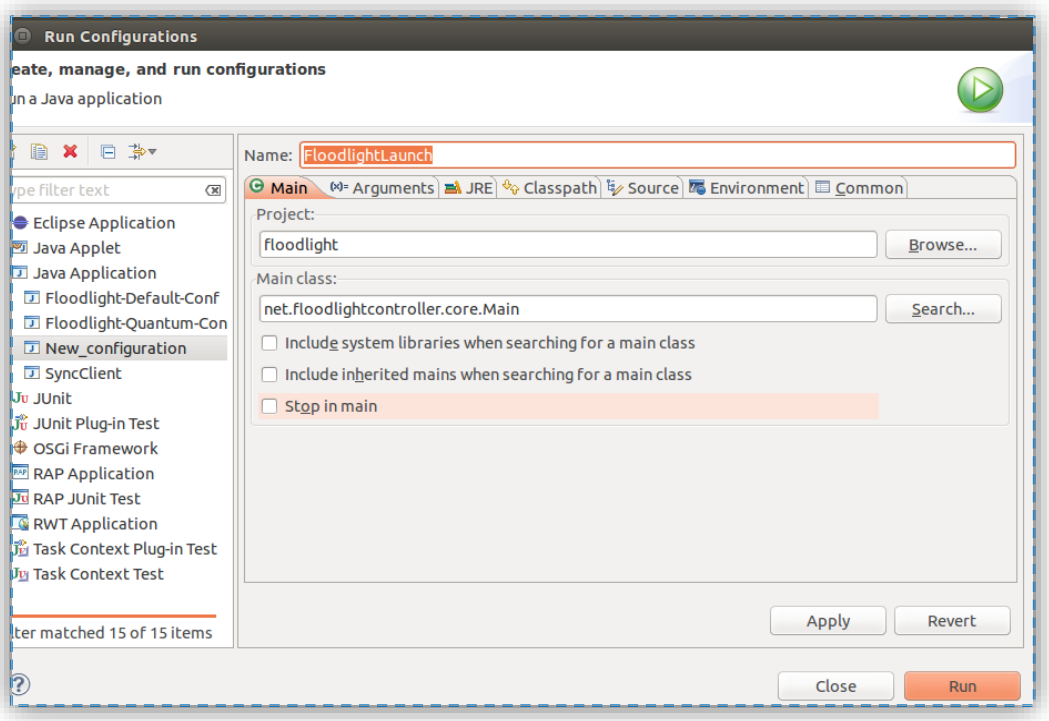


Fig. 4. 10 Creación del FloodlightLaunch para la ejecución de Floodlight

4.4.2 Configuración de los switches HP 3500-24

En este ítem se describe el proceso de configuración de los switches HP 3500-24 el cual es necesario para el correcto funcionamiento del protocolo OpenFlow para lo cual se siguió los siguientes pasos:

1) Creación de una vlan exclusivamente para el controlador.

```
configure                  //Ingresamos a modo configuración
vlan 10                    //Creamos una vlan
name SW1                   //Asignamos un nombre a la vlan
untagged 13-16             //Agregamos los puertos desigandos para el controlador
```

ip address 192.168.1.200/24 //Asignamos una ip que sea de la misma subred que la del controlador a la vlan

exit

2) Creación de una vlan designada para los host de la red

vlan 20 //Creamos la vlan

name openflow //Asignamos un nombre

unttaged 1-12 //Agregamos los puertos desde el 1 hasta el 12

exit

3) Configuración de la comunicación del switch HP con el controlador

openflow //Ingresamos al OpenFlow

controller-id 1 ip 192.168.1.100 port 6633 controller-interface vlan 10 //Agregamos el controlador

instance sdn //Creamos una instancia

member vlan 20 //Asignamos la vlan 20 a la instancia OpenFlow

controller-id 1 //Agregamos el controlador a la vlan

enable //Activamos la instancia

exit //Salimos de la instancia

enable //Activamos el OpenFlow

exit

4.4.3 Configuración del Switch OpenFlow Habilitado.

- **Actualización del firmware del router TP-LINK Modelo TL-WR1043ND Versión 2.1**

Este proceso se lleva a cabo con la finalidad de dotar al router de características más avanzadas como QoS o VPN. En el siguiente proyecto se actualizó el firmware del router al sistema Operativo OpenWRT, con el objetivo de habilitar el protocolo OpenFlow al router de bajo costo.

En la dirección electrónica http://wiki.openwrt.org/toh/start se indica la lista de dispositivos soportados por OpenWRT. En la figura 4.11 se muestra la imagen de este router.

Fig. 4. 11 Imagen del router TP-LINK TL-WR1043ND versión 2.1

Fuente: http://wiki.openwrt.org/toh/tp-link/tl-wr1043nd

- Creación de la imagen OpenWRT

Para crear la imagen del sistema operativo OpenWRT se debe realizar los siguientes pasos:

1. Instalar los paquetes necesarios para construir la imagen mediante los siguientes comandos:

sudo apt-get update

sudo apt-get install autoconf binutils bison build-essential ccache flex gawk gettext git libncurses5-dev libssl-dev ncurses-term quilt sharutils subversion texinfo xsltproc zlib1g-dev

2. Crear una carpeta denominada OpenWRT para almacenar los archivos de la imagen.

mkdir ~/openwrt

cd ~/openwrt

svn co svn://svn.openwrt.org/openwrt/trunk/

3. Actualizar e instalar los feeds

cd ~/openwrt/trunk

./scripts/feeds update -a

./scripts/feeds install –a

4. Establecer la arquitectura y modelo de router para la creación de la imagen OpenWRT mediante los siguientes comandos:

cd ~/openwrt/trunk

make menuconfig

Establecer el **'Target System'** a **Atheros AR71xxx/Ar9xxx** como se muestra en la figura 4.12.

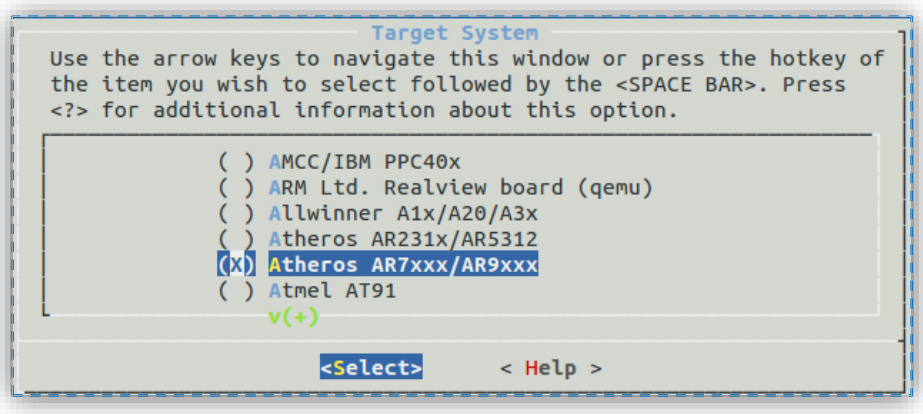


Fig. 4. 12 Selección de la arquitectura del router

Establecer el **'Target Profile'** a **TP-LINK TL-WR740N/ND OR TP-LINK TL-WR1043ND**, como se muestra en la figura 4.13.

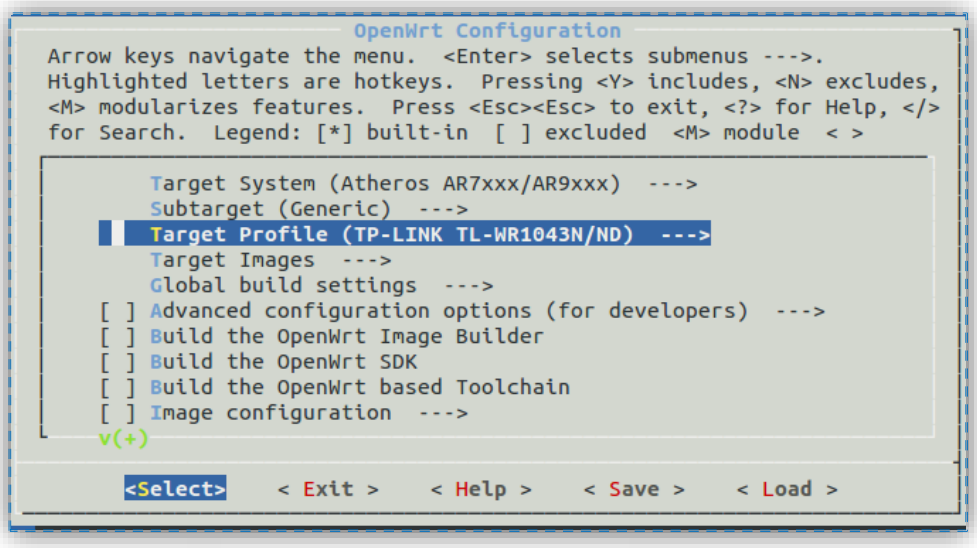


Fig. 4. 13 Selección del Target Profile

5. Creación de la imagen mediante el comando **Make.**

Es necesario aclarar que la primera vez que se realiza este proceso el tiempo de creación tarda varias horas para posteriores modificaciones este proceso es mucho más rápido.

- Sustitución del firmware por OpenWRT

    1. Conectar la PC al router mediante un cable de red a cualquiera de los puertos LAN.

    2. Configurar la red para que el router entregue una IP por DHCP.

    3. Acceder al panel de administración del router vía web, a través de la dirección: **192.168.1.1**, con el usuario **admin** y contraseña **admin**.

    4. Dirigirse al menú System tolos, Firmware Upgrade.

    5. Seleccionar la imagen creada anteriormente la cual se encuentra en el directorio: **~/openwrt/trunk/bin/ar71xx/** bajo el siguiente nombre:

       openwrt-ar71xx-generic-tl-wr1043nd-v2-squashfs-factory.bin

       Es necesario aclarar que la primera vez que se actualiza el firmware del router la imagen que se debe actualizar es: **squashfs-factory.bin** mientras que para posteriores actualizaciones debe ser la imagen: **squashfs-sysupgrade.bin**. En la figura 4.14 se muestra el proceso de actualización del firmware.

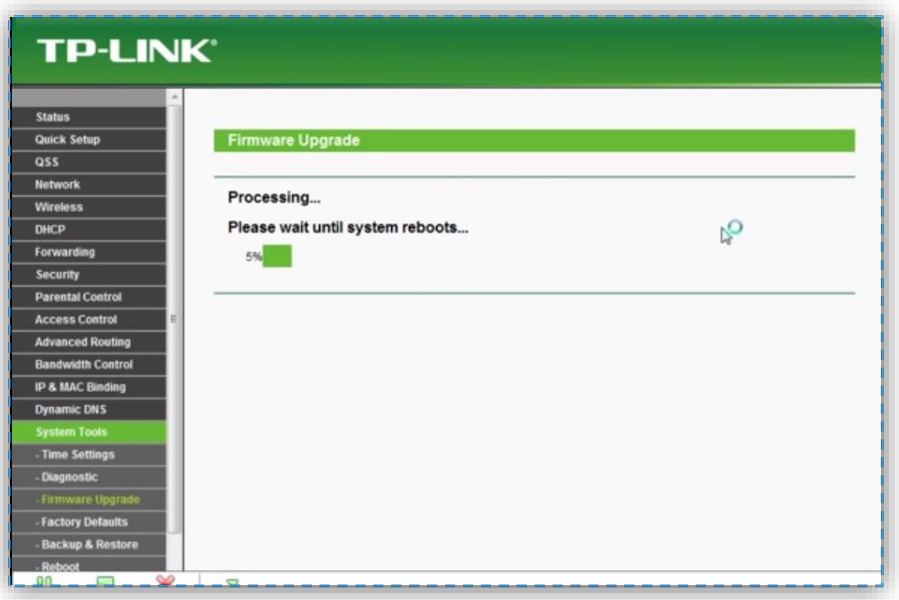


Fig. 4. 14 Actualización del firmware

- Instalación de la interfaz web del sistema OpenWRT denominada LuCI

  Debido a que este firmware está en la versión trunk no viene con la interfaz visual de serie, por lo cual debe ser instalada mediante los siguientes pasos:

  1. Acceder al router vía telnet a la dirección 192.168.1.1 lo cual permitirá acceder a la interfaz del sistema operativo OpenWRT que se muestra en la figura 4.15.

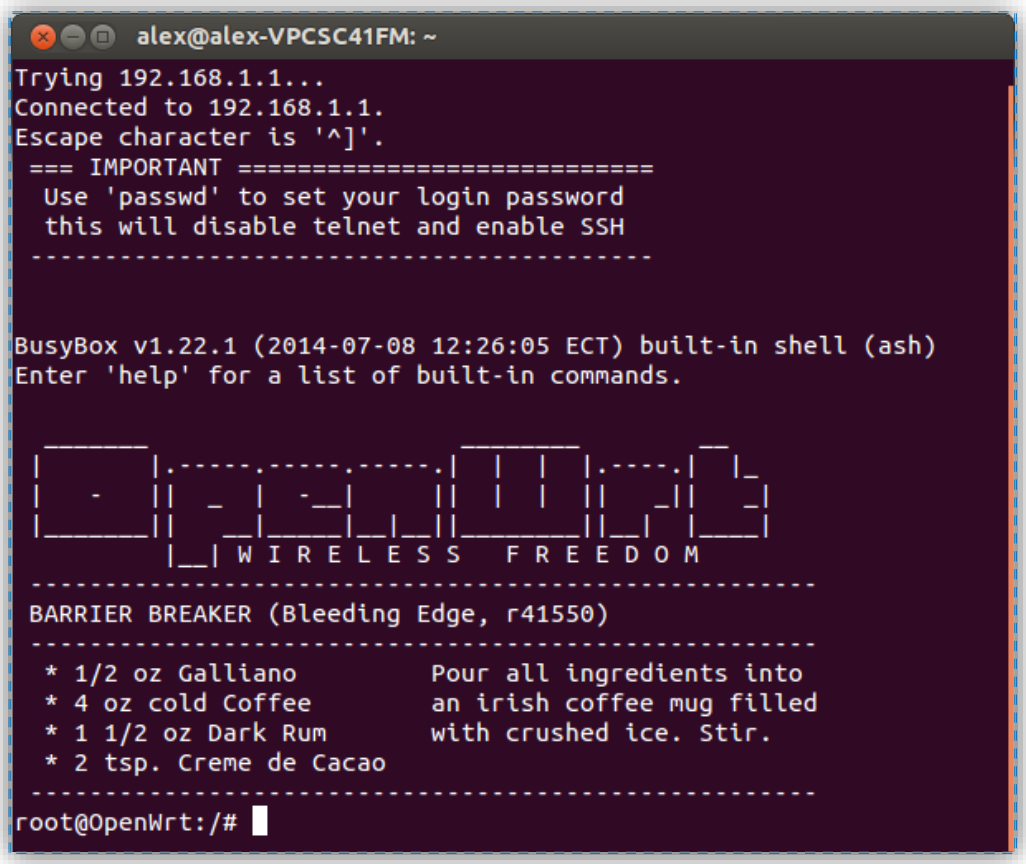


Fig. 4. 15 Interfaz del sistema OpenWRT

2. Establecer conexión a internet al router.

3. Actualización de repositorios mediante el comando: **opkg update.**

4. Instalar la interfaz LuCI mediante el comando: **opkg install luci.**

5. Habilitar el servidor web mediante el comando: **/etc/init.d/uhttpd enable.**

6. Iniciar el servicio mediante el comando: **/etc/init.d/uhttpd start** [37]**.**

- Adición del protocolo OpenFlow a la imagen del router

  Terminado el proceso de creación de la imagen OpenWRT se debe incluir el protocolo OpenFlow dentro de esta imagen para lo cual se realizaron los siguientes pasos:

1. Crear un directorio y descargar OpenFlow y enlazarlo a OpenWRT para la compilación.

2. Cabe mencionar que la versión de OpenFlow que se descargó es la 1.0 y se realizó este proceso mediante los siguientes comandos:

   cd ~/openwrt

   git clone git://gitosis.stanford.edu/openflow-openwrt

   cd openflow-openwrt

   git checkout -b openflow-1.0/tplink origin/openflow-1.0/tplink

   cd ~/openwrt/trunk/package/

   ln -s ~/openwrt/openflow-openwrt/openflow-1.0/

   cd ~/openwrt/trunk/

   ln -s ~/openwrt/openflow-openwrt/openflow-1.0/files

3. Ingresar a la interfaz para la creación de la imagen mediante los siguientes comandos:

   cd ~/openwrt/trunk

   make menuconfig

4. Seleccionar el paquete **OpenFlow** dentro del paquete '**Network**'. (Ver imagen 4.16).

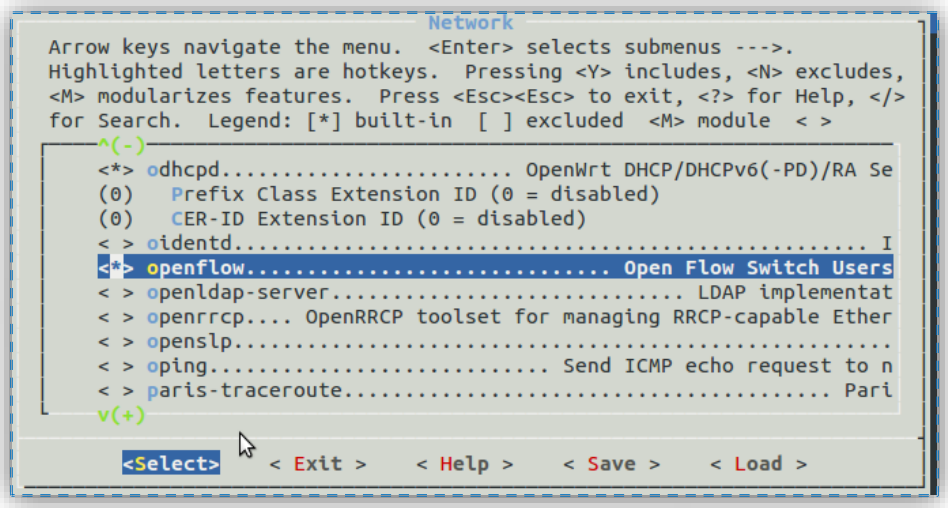


Fig. 4. 16 Adición del protocolo OpenFlow a la imagen del router

5. Seleccionar el paquete '**kmod-tun**' dentro de '**Kernel Modules/ network support**' como se muestra en la figura 4.17.

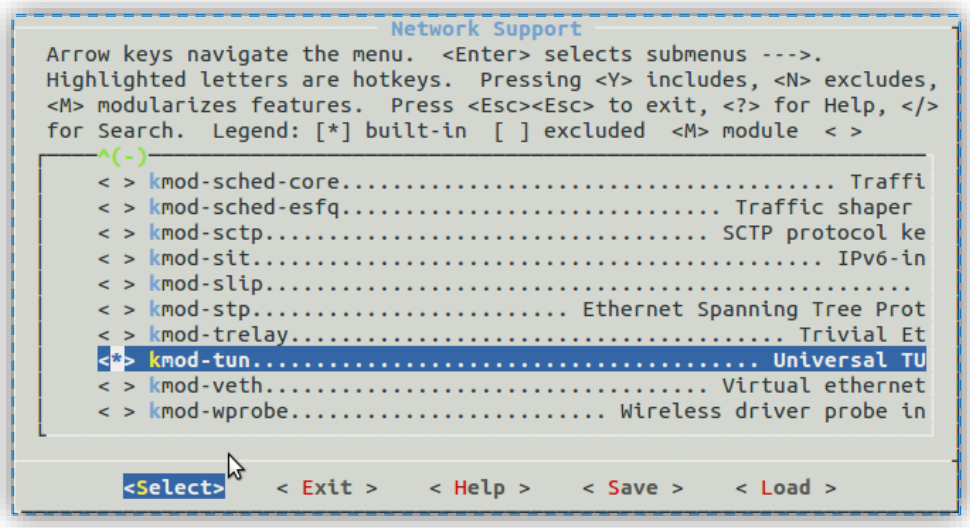


Fig. 4. 17 Selección del paquete kmod-tun

6. Seleccionar el paquete 'kmod-sched-core' y 'kmod-sched' dentro del paquete 'Kernel Modules / network support'

   El paquete '**kmod-sched-core**' es un requisito para instalar el paquete **TC**, necesario para el funcionamiento del protocolo OpenFlow.

7. Ingresar al menú **kernel_menuconfig** mediante el siguiente comando:

   make kernel_menuconfig

8. Seleccionar el paquete 'Hierarchical Token Bucket (HTB)' dentro de 'Networking support/Networking options' en la sección 'Queueing/Scheduling'.

9. Compilar la nueva imagen con el protocolo OpenFlow mediante los siguientes comandos:

   cd ~/openwrt/trunk

   make V=s (V=s es opcional y será útil para depurar en caso de cualquier error).

- Actualizar la imagen en el router con el protocolo OpenFlow

  1. Localizar la imagen openwrt-ar71xx-generic-tl-wr1043nd-v2-squashfs-sysupgrade.bin dentro del directorio ~/openwrt/trunk/bin/ar71xx/.

  2. Ingresar a la interfaz web del router. (ver figura 4.18)

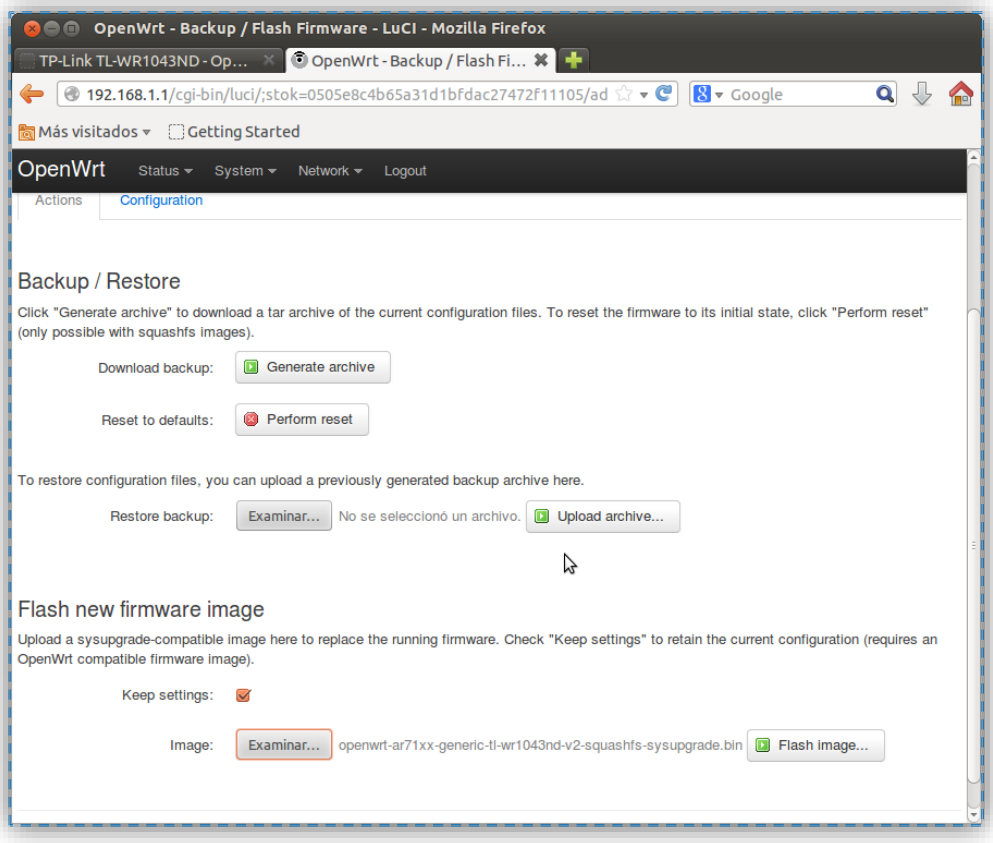


Fig. 4. 18 Actualización de la imagen con el protocolo OpenFlow

- Configuración del protocolo OpenFlow

1. Ingresar al router vía telnet.
2. Instalar el paquete **tc** mediante los siguientes comandos:

   opkg update

   opkg install tc

3. Crear un link para el paquete tc.

   cd /sbin

   ln -s /usr/sbin/tc

En la figura 4.19 se muestra la creación de este link

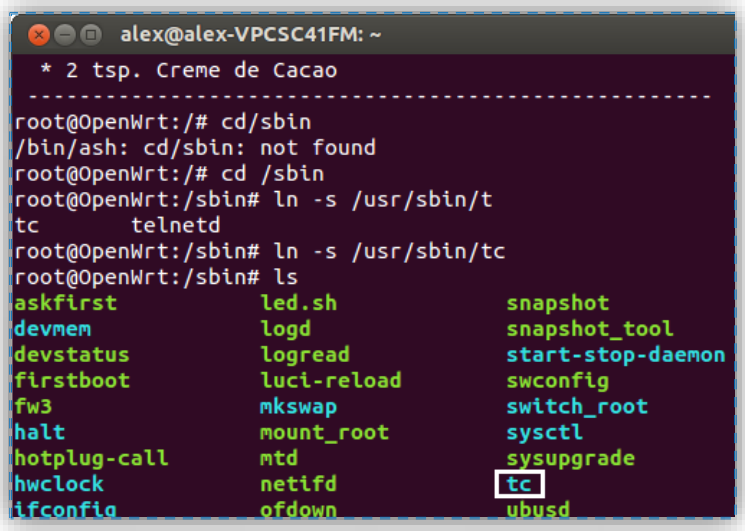


Fig. 4. 19 Creación del link del paquete tc

4. Crear el link **functions.sh** mediante los siguientes comandos:

   cd /etc

   ln -s /lib/functions.sh

5. Verificar que los procesos **ofprotocol** y **ofdatapath** se ejecuten correctamente mediante el siguiente comando: [38].

   To check: ps | grep of

   (Ver figura 4.20)

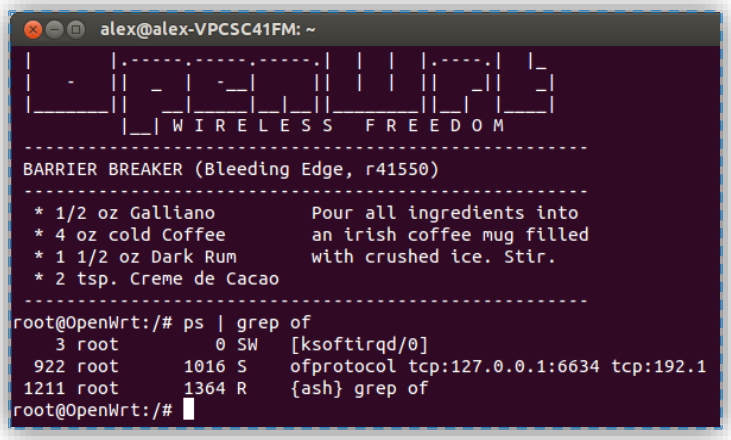


Fig. 4. 20 Verificación del protocolo OpenFlow

- Configuración del archivo de red

Luego de haber instalado estos paquetes es necesario configurar los archivos del protocolo OpenFlow y Network que se encuentran dentro del directorio ***/etc/config/*** como se muestra en la figura 4.21.

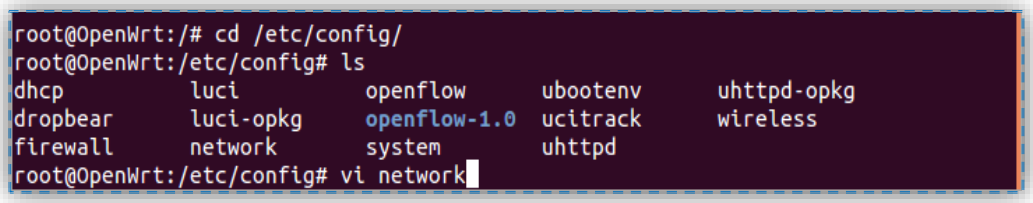


Fig. 4. 21 Archivos del protocolo OpenFlow y Network

La configuración inicial del archivo de red se muestra a continuación en la figura 4.22.

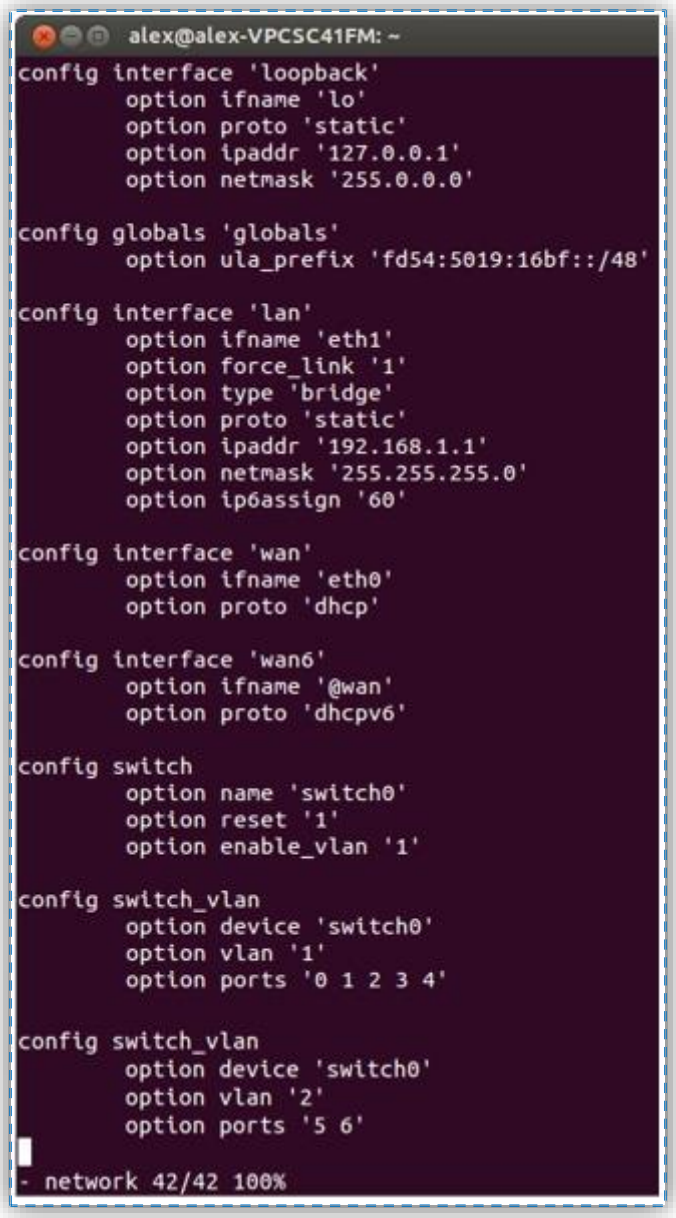


Fig. 4. 22 Configuración inicial del archivo Network

Este archivo se lo modificó teniendo en cuenta algunas consideraciones tales como: la distribución de puertos del router ya que al cambiarle el firmware los puertos cambian de configuración en la figura 4.23 se muestra esta configuración:

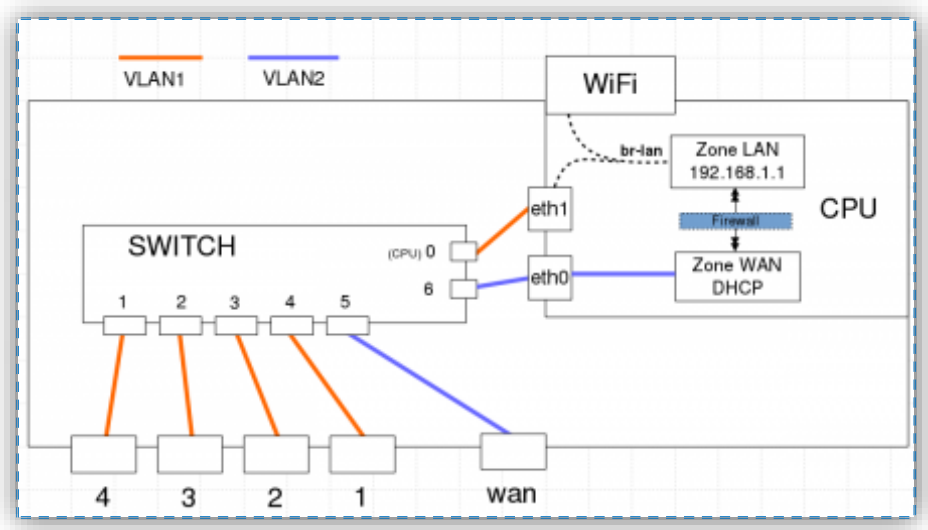
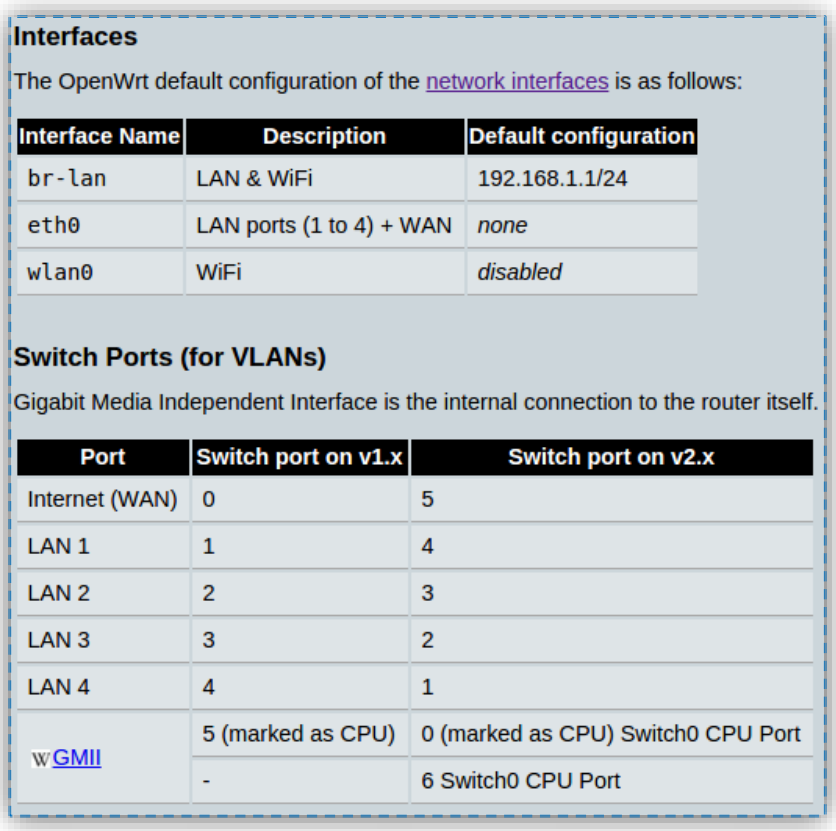


Fig. 4. 23 Configuración de los puertos del router TP-LINK TL-WR1043ND

Fuente: http://wiki.openwrt.org/toh/start

Otra cosa que debemos tomar encuentra es que en la versión 2 del router TP-LINK TL-WR1043ND las interfaces LAN y WAN ya no se referencian a través de las interfaces eth0.1/eth0.2 como en versiones anteriores sino que la WAN se referencia mediante la interfaz eth0 y la LAN se hace referencia mediante eth1. Además el puerto 0 del router es un puerto común que permite establecer la conexión entre los puertos físicos del router y el sistema operativo OpenWRT.

La interfaz WAN está en la VLAN 2, y no debe ser modificada, porque se perderá el puerto WAN. Por lo cual para la configuración del archivo de red la vlan 2 no debe ser modificada, y se deben asignar las interfaces eth1.x en dependencia de los puertos que se desee habilitar. En la figura 4.24 se muestra la distribución de puertos e interfaces del router



Interfaces

The OpenWrt default configuration of the network interfaces is as follows:

Interface Name	Description	Default configuration
br-lan	LAN & WiFi	192.168.1.1/24
eth0	LAN ports (1 to 4) + WAN	*none*
wlan0	WiFi	*disabled*

Switch Ports (for VLANs)

Gigabit Media Independent Interface is the internal connection to the router itself.

Port	Switch port on v1.x	Switch port on v2.x
Internet (WAN)	0	5
LAN 1	1	4
LAN 2	2	3
LAN 3	3	2
LAN 4	4	1
GMII	5 (marked as CPU)	0 (marked as CPU) Switch0 CPU Port
	-	6 Switch0 CPU Port

Fig. 4. 24 Distribución de puertos e interfaces del router

Fuente: http://wiki.openwrt.org/toh/start

A continuación se muestra el archivo de red modificado:

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd2e:33f7:2381::/48'

config interface 'lan'
        option ifname 'eth1.5'
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.1'
```

```
        option netmask '255.255.255.0'
        option ip6assign '60'
        option dns '8.8.8.8'

config interface 'wan'
        option ifname 'eth0'
        option proto 'dhcp'

config interface 'wan6'
        option ifname '@wan'
        option proto 'dhcpv6'

config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '1'
        option enable_learning '0'

config switch_vlan
        option device 'switch0'
        option vlan '1'
        option ports '0t 1'
        option vid '1'

config switch_vlan
        option device 'switch0'
        option vlan '3'
        option ports '0t 2'
        option vid '3'

config switch_vlan
        option device 'switch0'
        option vlan '4'
        option ports '0t 3'
        option vid '4'

config switch_vlan
        option device 'switch0'
        option vlan '2'
        option ports '5 6'
        option vid '2'

config switch_vlan
        option device 'switch0'
        option vlan '5'
        option ports '0t 4'
        option vid '5'
```

```
config interface
        option ifname 'eth1.1'
        option proto 'static'

config interface
        option ifname 'eth1.3'
        option proto 'static'

config interface
        option ifname 'eth1.4'
        option proto 'static'

config interface
        option ifname 'eth1.5'
        option proto 'static'
```

Para la correcta configuración de este archivo se debe crear las vlans como se muestra a continuación:

```
config switch_vlan
        option device 'switch0'
        option vlan '1'
        option ports '0t 1'
        option vid '1'
```

Dentro de esta vlan se debe relacionar al puerto que deseemos habilitar con el Puerto 0 más la etiqueta (t) lo cual permite crear un puente entre los puertos físicos del router y el sistema operativo OpenWRT.

Adicionalmente se debe crear la interfaz que se le asignara a este puerto:

```
config interface
        option ifname 'eth1.1'
        option proto 'static'
```

- Configuración del archivo OpenFlow

Finalmente se debe modificar el archivo OpenFlow, en la figura 4.25 se muestra el archivo original de OpenFlow al cual se lo modificó de la siguiente manera.

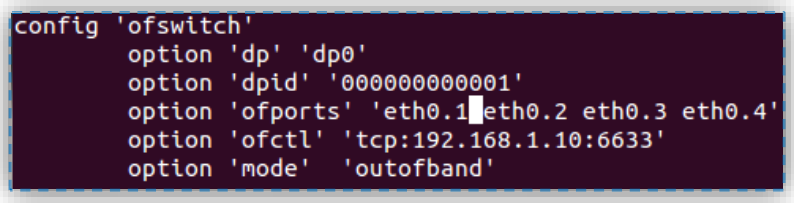


Fig. 4. 25 Configuración inicial del archivo OpenFlow

```
config 'ofswitch'
        option 'dp' 'dp0'
        option 'dpid' '000000000003'
        option 'ofports' 'eth1.1 eth1.3 eth1.4'
        option 'ofctl' 'tcp:192.168.1.100:6633'
        option 'mode'  'outofband'
```

En este archivo se deben configurar los siguientes parámetros.

Dp: es el nombre del datapath.

Ofports: son los puertos que habilitados para trabajar con el protocolo OpenFlow.

Ofctl: en esta opción se debe modificar los parámetros del controlador, la dirección ip y el puerto.

Mode: 'inband' o 'outofband' acorde a la red.

Para que los cambios surjan efecto es necesario reiniciar tanto el archivo network como el archivo OpenFlow en la figura 4.26 se muestra este proceso:

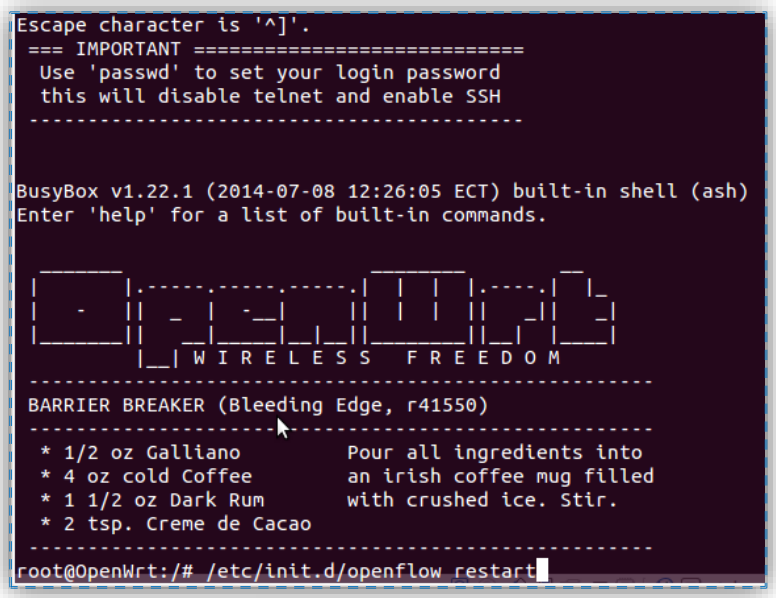


Fig. 4. 26 Ejecución de OpenFlow

Al conectar el router al controlador se deberá observar el siguiente mensaje (ver figura 4.27) [39] [40] [33].

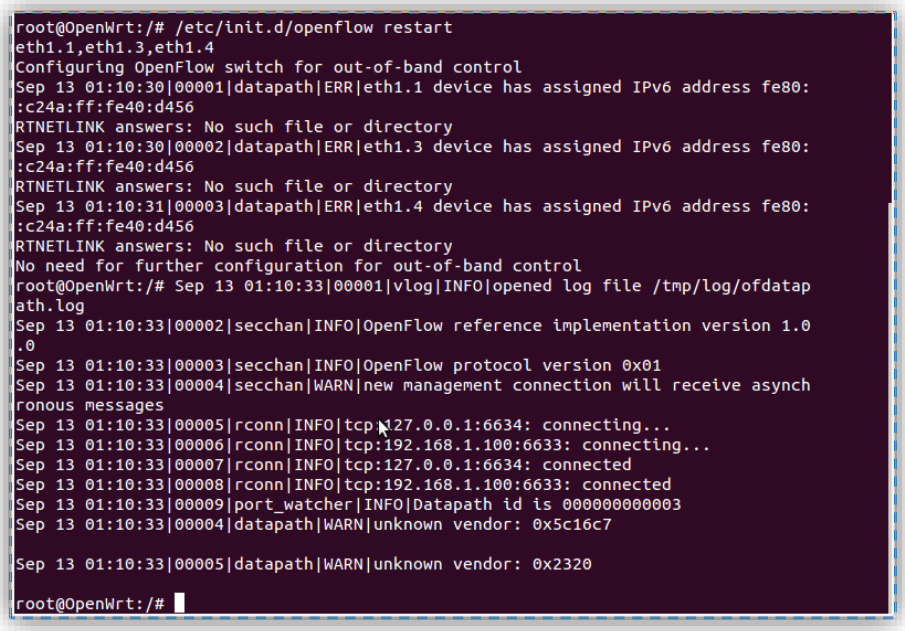


Fig. 4. 27 Conexión del router al controlador

4.5 Creación de módulos para los controladores de la red.

Luego de configurar los controladores elegidos, se procede a crear los siguientes módulos para el prototipo de red:

4.5.1 Creación de módulos en Beacon

- **Creación del proyecto que contendrá al modulo**

Para crear un nuevo proyecto en Beacon es necesario realizar los siguientes procesos:

1. Dirigirse a: File/New/Other/Plug-in Development/Plugin-in Project.
2. Establecer el nombre del proyecto que debe estar dentro del directorio de los paquetes de Beacon como se muestra a continuación:

   net.beaconcontroller.[nombre del módulo]

3. Deshabilitar la opción de guardar en la ubicación por defecto ya que el módulo se lo debe almacenar en la dirección donde se encuentran todos los paquetes de Beacon que es la siguiente:

   ~/git/Beacon/net.beaconcontroller.[nombre del módulo]

4. Cambiar la carpeta fuente donde se almacenan los paquetes del módulo correspondiente a:

   src/main/java

5. Cambiar el **Target Platform** a **an OSGi Framework** y seleccionar **Equinox.** En la figura 4.28 se mesutra este proceso.

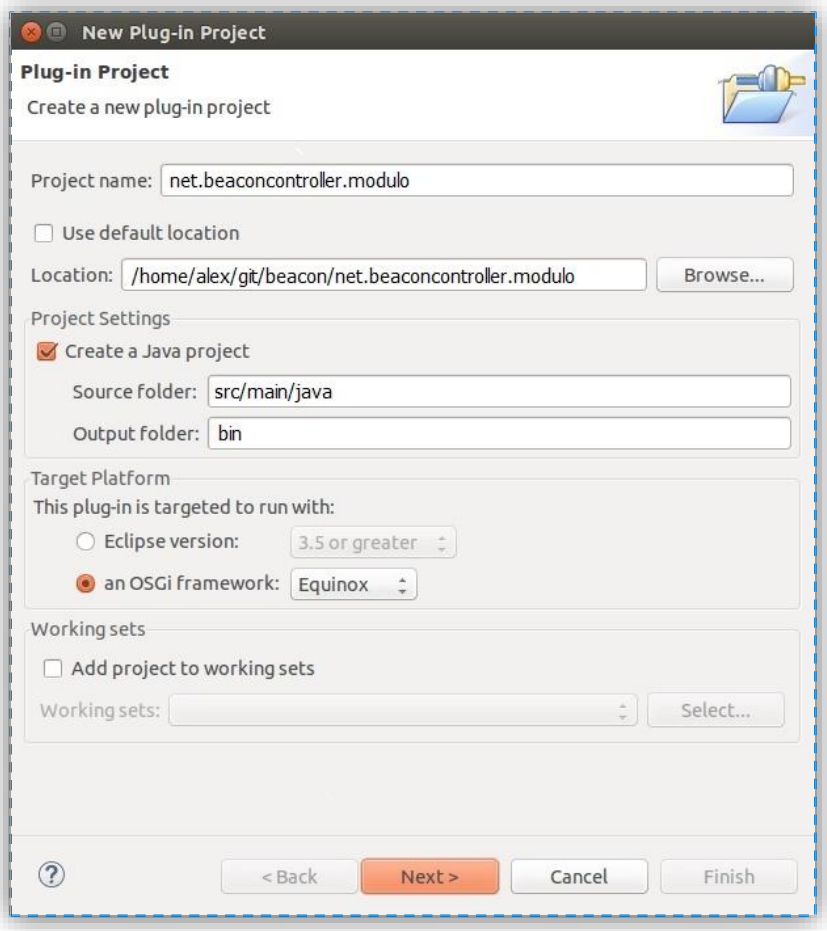


Fig. 4. 28 Configuración de los parámetros del nuevo proyecto en Beacon

6. Establecer el nombre del proyecto en la opción Name como se muestra a continuación:

net.beaconcontroller.[nombre del proyecto]

7. Deshabilitar la opción generar un activador. Como se muestra en la figura 4.29.

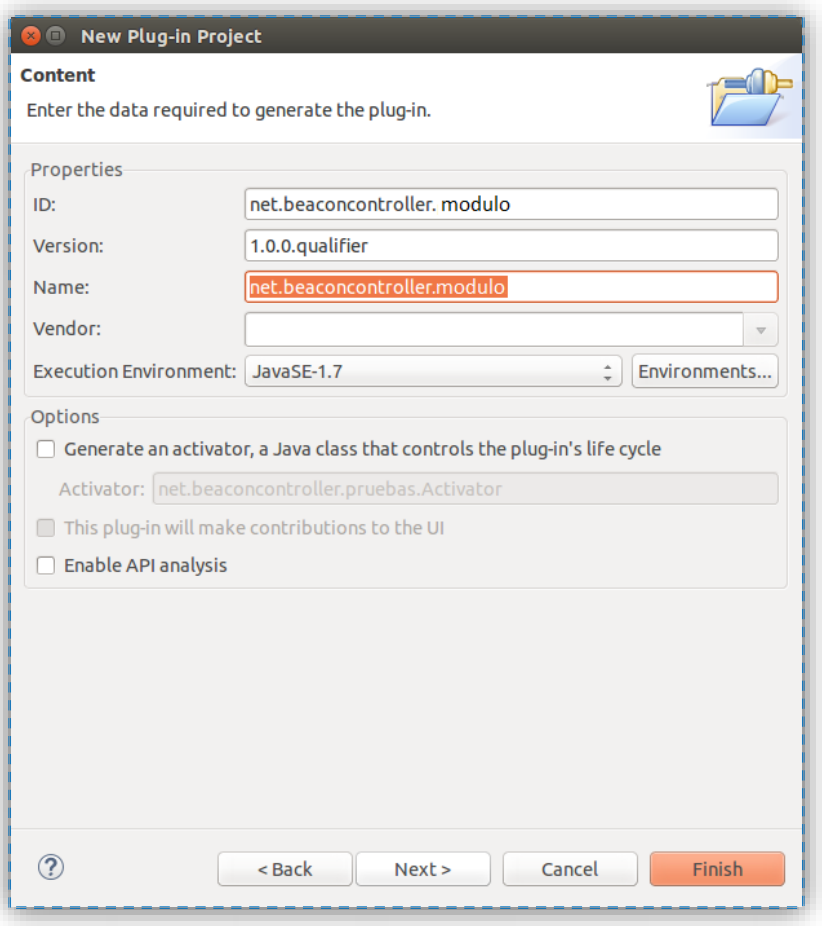


Fig. 4. 29 Establecimiento del nombre del proyecto

8. Deshabilitar la opción crear un plugin a partir de una plantilla.

Al terminar este proceso deberemos observar en la parte izquierda de eclipse un nuevo proyecto llamado **net.beaconcontroller.[nombre del proyecto]**, y su **MANIFEST.MF** se encontrará abierto en el editor de Manifiesto de Eclipse como muestra la figura 4.30.

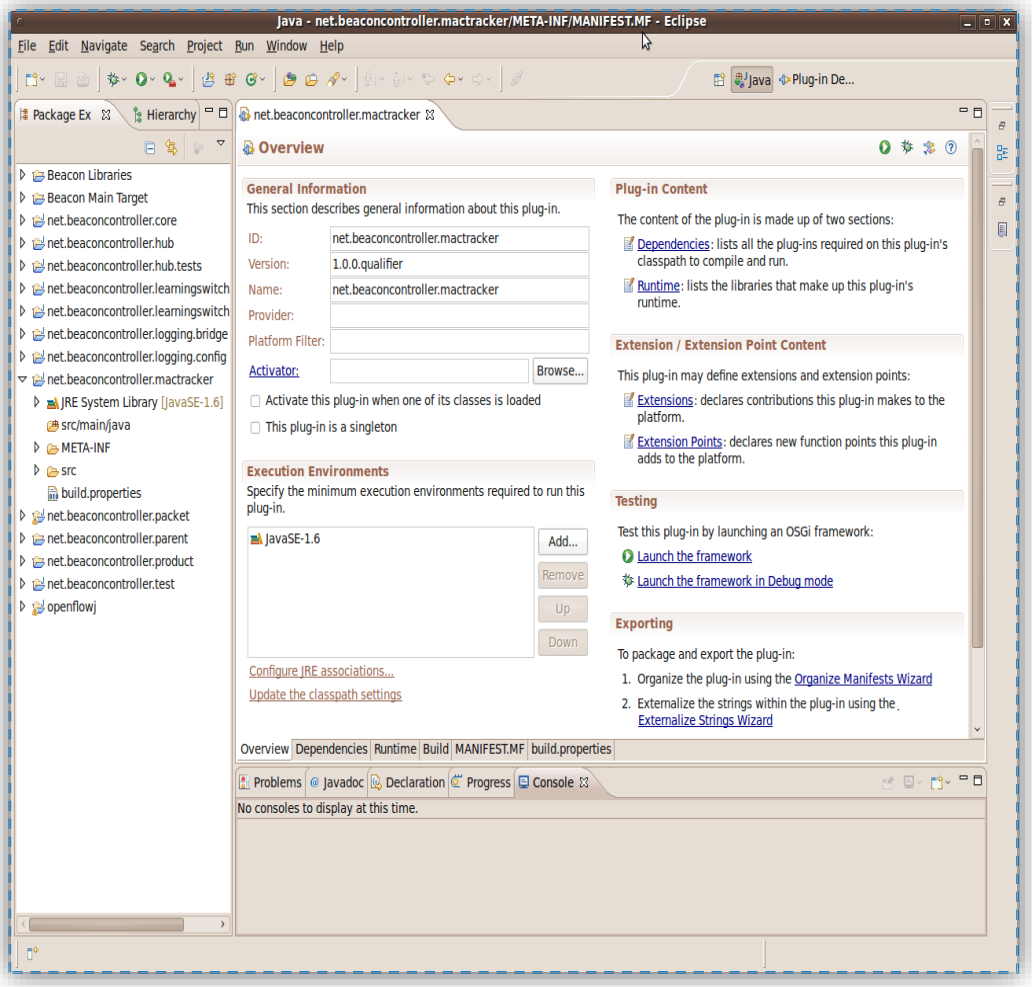


Fig. 4. 30 Ambiente de trabajo del controlador Beacon

- **Adición de las dependencias del proyecto.**

En este paso es necesario agregar las dependencias que se utilizará en la programación del módulo, para ello es necesario realizar los siguientes pasos:

1. Abrir el proyecto anteriormente creado
2. Abrir el editor de manifiestos que se encuentra en la siguiente dirección: **META-INF/MANIFEST.MF**
3. Dirigirse a la pestaña **Dependencias**
4. Presionar el link ***Automated Management of Dependencies*** y agregar las siguientes dependencias:

net.beaconcontroller.core

org.openflow

5. Guardar los cambios realizados. Este proceso es necesario para que se puedan añadir las dependencias. En la figura 4.31 se muestra dicho proceso.

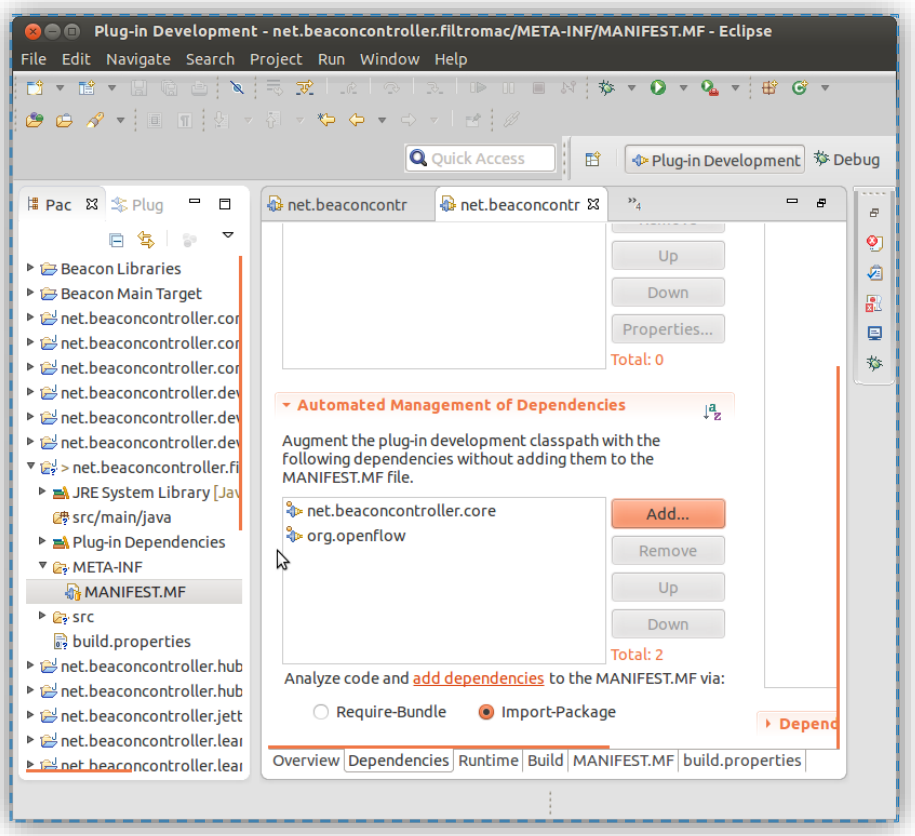


Fig. 4. 31 Establecimiento de dependencias en Beacon

- **Creación de paquetes y clases del módulo**

En este paso se creó los paquetes y clases necesarios para crear el código, estos archivos se almacenan en la carpeta fuente designada como **src /main / java.** Por ello es necesario realizar los siguientes pasos:

1. Click derecho en la carpeta **src /main / java**
2. Seleccionar **New/Package.**
3. Establecer el nombre del paquete a: **net.beaconcontroller.[nombre del**

proyecto].

4. Click derecho en el paquete recién creado y seleccionar **New-Class** y establecer el nombre de la clase.
5. Añadir las interfaces: **IOFMessageListener** y **IOFSwitchListener** necesarias para interactuar con los switches y los mensajes OpenFlow. Para añadir estas interfaces se debe dar click en el botón añadir debajo del nombre de la clase ya establecida. En la figura 4.32 se muestra este proceso.

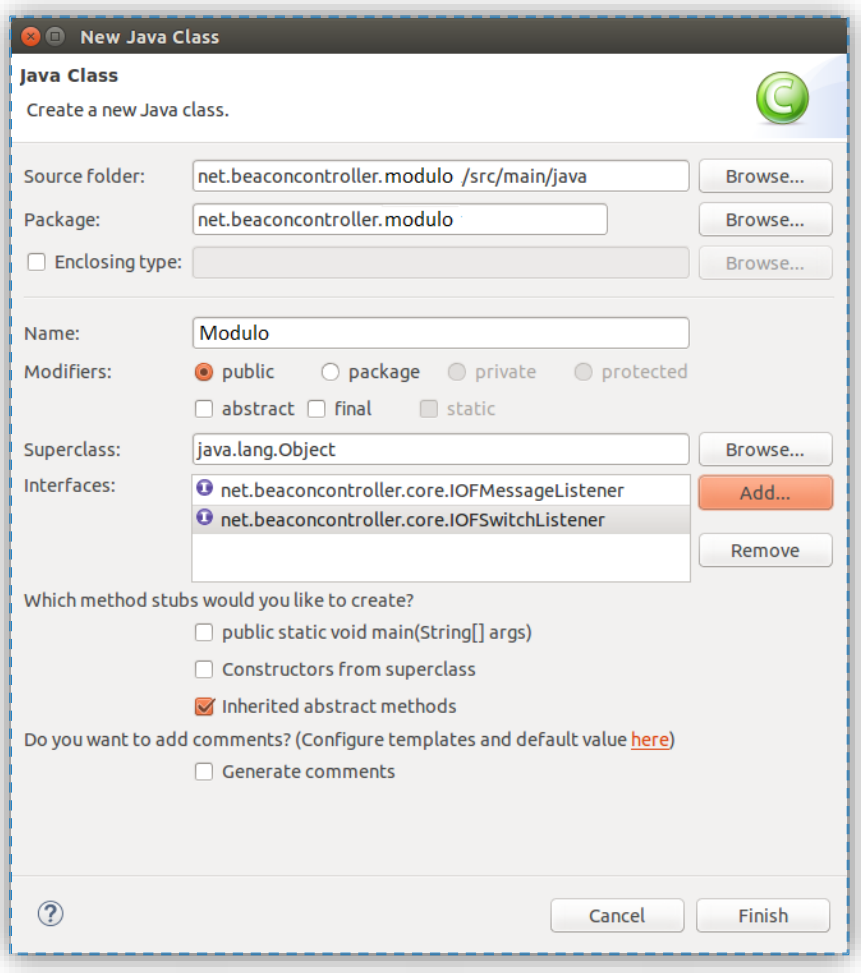


Fig. 4. 32 Elección de las interfaces de Beacon

- **Registro del módulo creado**

En este paso se registró el módulo en el núcleo de Beacon a través de los siguientes procesos:

1. Adquirir el objeto **IBeaconProvider**.

2. Crear una variable miembro **IBeaconProvider** en nuestra clase y el método **setbeaconProvider** como se muestra a continuación:

```
package net.beaconcontroller.modulo;
import java.io.IOException;
import org.openflow.protocol.OFMessage;
import net.beaconcontroller.core.IBeaconProvider;
import net.beaconcontroller.core.IOFMessageListener;
import net.beaconcontroller.core.IOFSwitch;
import net.beaconcontroller.core.IOFSwitchListener;

public class Modulo implements IOFMessageListener,
IOFSwitchListener {

    protected IBeaconProvider beaconProvider;
    @Override
    public void addedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub
    }
        public void startUp(){
    }
        public void shutDown(){
    }
      public void setBeaconProvider(IBeaconProvider
      beaconProvider) {
        this.beaconProvider = beaconProvider;
    }
    @Override
    public void removedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub
    }
    @Override
      public Command receive(IOFSwitch sw, OFMessage msg)
      throws IOException {
        // TODO Auto-generated method stub
        return null;
    }
```

```
    @Override
    public String getName() {
        // TODO Auto-generated method stub
        return "FiltroMac";
    }
}
```

3. Click derecho en la carpeta **META-INF** del proyecto.
4. Seleccionar **New/Folder**, y establecer el nombre de la carpeta como **spring,** como se muestra en la figura 4.33.

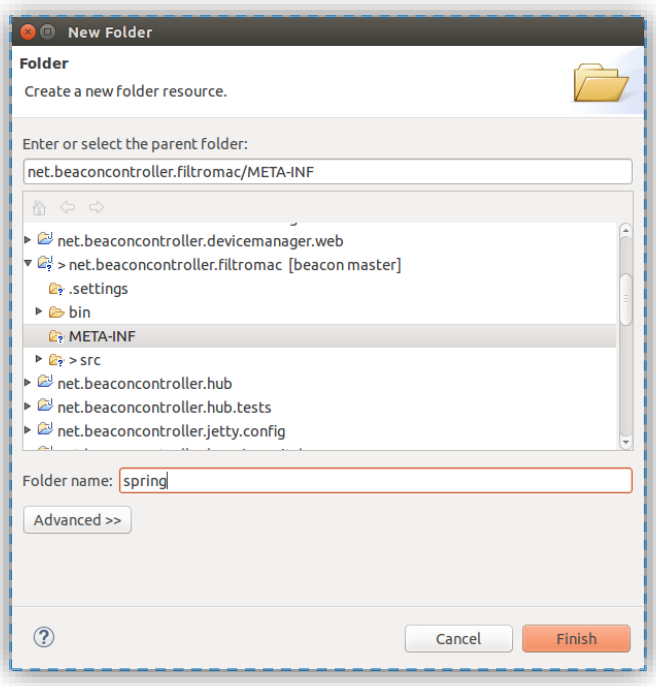


Fig. 4. 33 Creación de la carpeta spring

5. Click derecho en la carpeta **spring**, y seleccionar **New/file**,
6. Establecer el nombre del archivo como **context.xml** y colocar el siguiente contenido dentro de la pestaña source de este archivo.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation="*http://www.springframework.org/schema/beans*
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="*modulo*" class="*net.beaconcontroller.modulo.Modulo*"
init-method="*startUp*" destroy-method="*shutDown*">
<property name="*beaconProvider*" ref="*beaconProvider*"/>
</bean>
</beans>

El **Spring** lee todos los archivos **xml** dentro de la carpeta **META-INF / spring** al iniciar. El archivo **context**, indica que un **bean** será creado con el nombre de **módulo**, la clase utilizada para crearlo, y los métodos init y destroy a llamar.

Dentro de la referencia del **bean** también hay un elemento de propiedad, diciéndole al **Spring** que el **bean** del **módulo** requiere un **bean** llamado **beaconProvider**, y el nombre de la propiedad que será utilizada para establecerlo es **"beaconProvider"**. En la figura 4.34 se muestra este proceso.

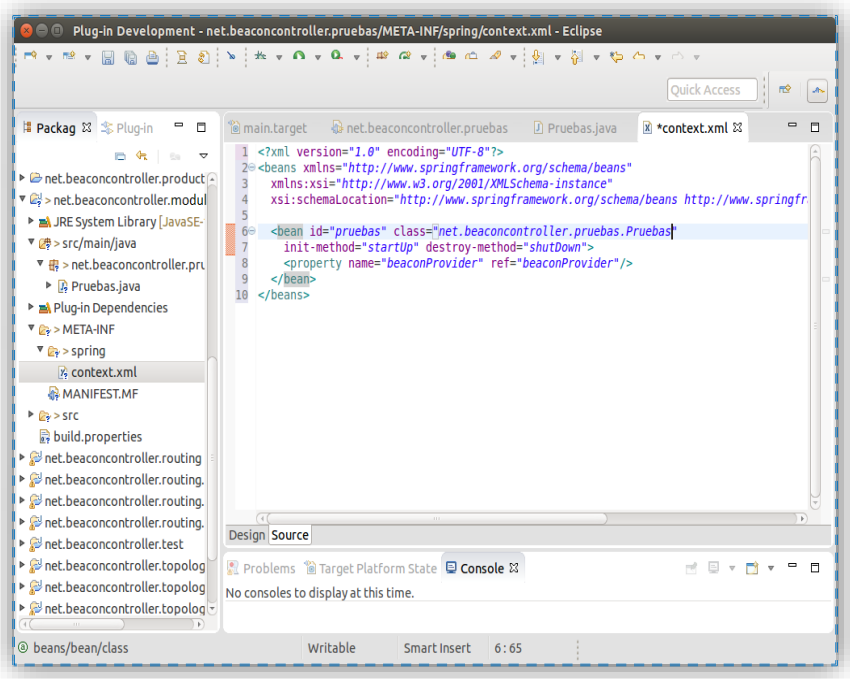


Fig. 4. 34 Creación del archivo de registro context.xml

7. Crear un segundo archivo, con el nombre **osgi.xml**, y colocar el siguiente contenido.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>
<beans xmlns=*"http://www.springframework.org/schema/beans"*
xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
xmlns:osgi=*"http://www.springframework.org/schema/osgi"*
xsi:schemaLocation=*"http://www.springframework.org/schema/beans*
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/osgi
http://www.springframework.org/schema/osgi/spring-osgi-1.2.xsd">

<osgi:reference id=*"beaconProvider"* cardinality=*"1..1"*
interface=*"net.beaconcontroller.core.IBeaconProvider">*
</osgi:reference>

</beans>

Este archivo requiere el **namespace osgi xml**. Esto declara un **bean** referenciado con el nombre de **beaconProvider**, el mismo que recupera este **bean** desde la declarativa de servicios de **OSGi** que coincida con una interfaz exportada de **net.beaconcontroller.core.IBeaconProvider**.

La **cardinalidad** es **1..1** indicando que el **Spring** debe resolver este **bean** antes de empezar la aplicación **context**. Una vez que el **Spring** ha resuelto **beaconProvider** esto creará una instancia de nuestra clase, llamando a su método **init** especificado, y terminar de cargar la aplicación **context**.

8. Crear los métodos los **start up** y **shut down**.

```
public void startUp(){
        beaconProvider.addOFMessageListener(OFType.PACKET_IN,
this);
    }
public void shutDown(){
```

```
beaconProvider.removeOFMessageListener(OFType.PACKET_IN,
this);
    }
```

Esto métodos se crean debido a que es importante que empecemos y terminemos un módulo correctamente ya que los **bundles** pueden iniciarse y detenerse dentro de un contenedor **OSGi**, incluyendo el **núcleo de Beacon**.

- **Adición de dependencias para generación de mensajes de error y depuración.**

1. Dirigirse al editor de manifiestos y agregar **slf4j.api** a la lista de gestión automatizada de dependencias **(Automated Management of Dependencies).**

2. Añadir el registrador estático **(static logger)** en el código como se muestra a continuación:

```
package net.beaconcontroller.modulo;

import java.io.IOException;
import java.util.logging.Logger;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFType;
import org.slf4j.LoggerFactory;
import net.beaconcontroller.core.IBeaconProvider;
import net.beaconcontroller.core.IOFMessageListener;
import net.beaconcontroller.core.IOFSwitch;
import net.beaconcontroller.core.IOFSwitchListener;

public class Modulo implements IOFMessageListener,
IOFSwitchListener {

    protected IBeaconProvider beaconProvider;
    protected static Logger logger = (Logger)
LoggerFactory.getLogger(Modulo.class);

    @Override
    public void addedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub
    }
```

```
    public void startUp(){
        beaconProvider.addOFMessageListener(OFType.PACKET_IN,
this);
    }

    public void shutDown(){

beaconProvider.removeOFMessageListener(OFType.PACKET_IN,
this);
    }

    public void setBeaconProvider(IBeaconProvider
beaconProvider) {
        this.beaconProvider = beaconProvider;
    }
    @Override
    public void removedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub
    }
    @Override
    public Command receive(IOFSwitch sw, OFMessage msg)
throws IOException {
        // TODO Auto-generated method stub
        return null;
    }
    @Override
    public String getName() {
        // TODO Auto-generated method stub
        return "Modulo";
    }
}
```

3. Cargar el manifiesto en el editor de manifiestos, en la pestaña **Dependencies** debajo de **Automatied Management of Dependencies.**

4. Asegurarse de que **Import-Package** este seleccionado.

5. Click en el link **add dependencias**.

6. Localizar los paquetes añadidos en la lista **Imported Packages.**

7. Borrar la versión del paquete **org.slf4j** haciendo click en el botón propiedades.

Terminado todo este proceso tenemos un base de código desde el cual se crean todos los módulos con sus diferentes funciones a partir de este punto el programador es el encargado de crear el código de acuerdo a sus necesidades.

- **Ejecución de un módulo en Beacon**

En este punto se procederá a explicar la manera de ejecutar los módulos creados en Beacon una vez que han sido realizados.

1. Abrir la lista debug configurations, haciendo click en la flecha junto al icono de error, luego en Debug Configurations o Run-Debug Configurations.

2. Seleccionar la configuración de ejecución Beacon debajo de OSGi Framework en el panel izquierdo, haga clic derecho y seleccione Duplicar

3. Cambiar el nombre de la nueva configuración a **[nombre del proyecto],** y en la pestaña de los paquetes (***bundles***) seleccione el bundle correspondiente al proyecto (si este paquete no esta visible en la lista, asegurarse de que no este seleccionada la opción ***Only show selected bundles***). Cabe recalcar que si se encuentra seleccionados los módulos de hub y learning switch pueden interferir con el correcto funcionamiento del nuevo módulo por lo cual estos no deben estar seleccionados.

4. Click en Aplicar y, a continuación, haga clic en Depurar para ejecutar el módulo como se muestra en la figura 4.35. [41]

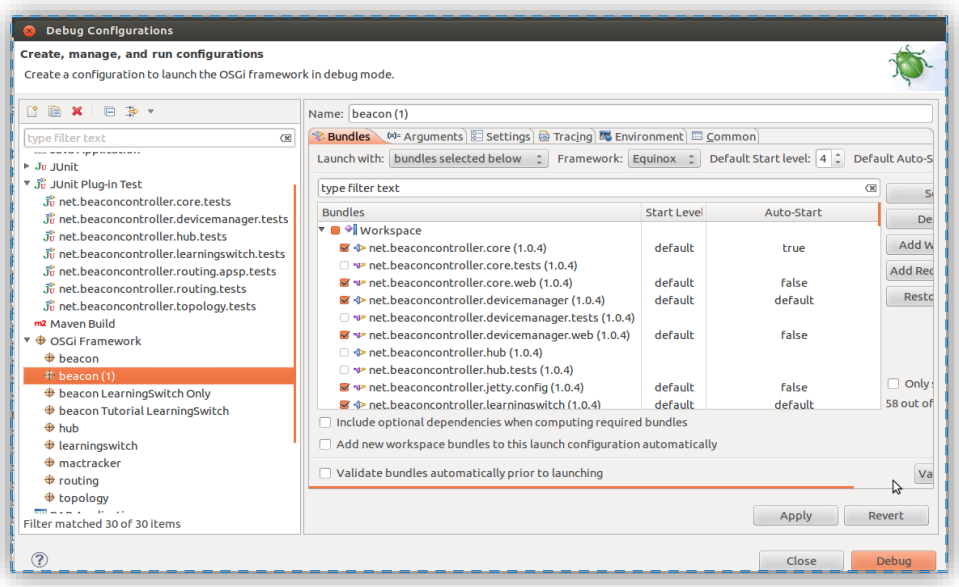


Fig. 4. 35 Ejecución de Beacon

4.5.2 Descripción de los módulos desarrollados en Beacon

A continuación se describirán los módulos realizados en Beacon

- **Modulo para inserción de reglas estáticas**

La primera línea de código se genera automáticamente al momento de crear la clase correspondiente al módulo y hace referencia al nombre del paquete.

```
package net.beaconcontroller.modulo;
```

La siguientes líneas de código pertenecen a las librerías necesarias para la construcción del proyecto, algunas de estas librerías se generan automáticamente al momento de añadir las interfaces que se emplearán y otras es necesario importarlas en función de los requerimientos del módulo.

```
import java.io.IOException;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.ConcurrentSkipListSet;
```

```
import net.beaconcontroller.packet.Ethernet;
import net.beaconcontroller.packet.IPv4;
import net.beaconcontroller.core.IBeaconProvider;
import net.beaconcontroller.core.IOFMessageListener;
import net.beaconcontroller.core.IOFSwitchListener;
import net.beaconcontroller.core.IOFSwitch;

import org.openflow.protocol.OFFlowMod;
import org.openflow.protocol.OFMatch;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFPacketIn;
import org.openflow.protocol.OFPacketOut;
import org.openflow.protocol.OFPort;
import org.openflow.protocol.action.OFAction;
import org.openflow.protocol.action.OFActionOutput;
import org.openflow.protocol.OFType;
import org.openflow.util.HexString;
import org.openflow.util.U16;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

La línea de código que se describe a continuación es aquella que establece el nombre de la clase en este caso **Modulo** y a partir de la cual se realizará el código.

```
public class Modulo implements IOFMessageListener,
IOFSwitchListener {
```

La línea siguiente crea un registrador que permite generar mensajes de depuración y error.

```
protected static Logger logger =
LoggerFactory.getLogger(Modulo.class);
```

Posteriormente se crea el objeto IBeaconProvider con sus respectivos métodos set y get, este objeto permite establecer la conexión con los switches conectados al controlador.

```
protected IBeaconProvider beaconProvider;
public IBeaconProvider getBeaconProvider() {
        return beaconProvider;
    }
public void setBeaconProvider(IBeaconProvider beaconProvider) {
        this.beaconProvider = beaconProvider;
}
```

Los siguientes métodos que se crean permiten iniciar y finalizar el módulo e iniciar la acción de recepción de flujos, los cuales ingresan al controlador transformados en mensajes PACKET_IN.

```
public void startUp() {
        beaconProvider.addOFMessageListener(OFType.PACKET_IN,
this);
    }
public void shutDown() {
        beaconProvider.removeOFMessageListener(OFType.PACKET_IN,
this);
    }
```

Posteriormente se crea un método el cual devuelve el nombre del módulo.

```
public String getName() {
        return "Modulo";
}
```

El método Command receive es la parte más importante del código ya que es el encardo de manejar los eventos que ocurren en el controlador y donde se realiza todo el proceso para interactuar con los mensajes y switches OpenFlow.

```
public Command receive(IOFSwitch sw, OFMessage msg) throws
IOException {

        OFPacketIn pi = (OFPacketIn) msg;
        Reglas(sw,pi);
        return Command.CONTINUE;
}
```

Posteriormente se procederá a explicar el proceso de inserción de las entradas de flujo desde el controlador, estas reglas se encuentran dentro del método Reglas para lo cual se realiza el siguiente proceso: [5] [20] [42] [43] [44] [45].

1. Creación del objeto PACKET_IN

Este objeto permite interactuar con los flujos que el switch no ha podido procesar y han sido enviados al controlador. En la siguiente línea de código se muestra la creación de este objeto.

```
public Command receive(IOFSwitch sw, OFMessage msg) throws
IOException {
        //Creación del objeto packet_in
        OFPacketIn pi = (OFPacketIn) msg;
        Reglas(sw,pi);
        return Command.CONTINUE;
   }
```

2. Identificación del switch donde se insertará la entrada de flujo.

   En el código que se ha desarrollado se ha definido un método denominado Reglas el cual se llama cada vez que un flujo de datos ingresa al controlador para establecer una entrada de flujo para el mismo. En este módulo hemos definido algunas variables las cuales permitirán cambiar parámetros en caso de realizar pruebas con otras infraestructuras, estas variables referencian los identificadores de los switches (dpids) los cuales deben ser de un formato long así como las direcciones ips de los host las cuales deben ser de un formato string.

```
public void Reglas(IOFSwitch sw, OFPacketIn pi) throws
IOException{

//INGRESE DATOS
    /**SWITCH1*/
    //DPID
    long id1=5894009871857728L;
    //long id1=1;
    //IP HOST1
    String ip1="192.168.111.13";
    //IP HOST2
    String ip2="192.168.111.14";

    /**SWITCH2*/
    //DPID
    long id2=5629651061111680L;
    //long id2=2;
    //IP HOST3
    String ip3="192.168.111.15";


    /**SWITCH3*/
    //DPID
    long id3=3;
    //IP HOST4
    String ip4="192.168.111.16";
```

3. Identificación del switch mediante el comando **sw.getId.**

```
if(sw.getId()==id1){
```

4. Creación de la entrada de flujo.

5. Para crear una entrada de flujo en el controlador se debe crear un objeto FlowMod y añadirla mediante el parámetro **setCommand** como se muestra en las siguientes líneas de código.

```
OFFlowMod fm=new OFFlowMod();
fm.setCommand(OFFlowMod.OFPFC_ADD);
```

El parámetro setCommand permite realizar varias acciones como añadir, borrar o modificar las entradas las entradas de flujo.

Las siguientes líneas de código pertenecen al objeto FlowMod y son aquellas que permiten establecer los tiempos de expiración de los paquetes así como su prioridad.

```
fm.setBufferId(-1);
fm.setIdleTimeout((short)30);
fm.setHardTimeout((short)30);
fm.setPriority((short)20);
```

Para la selección de los campos que entrarán dentro del proceso de comparación, en el controlador es necesario crear un objeto Match dentro del cual se debe establecer la información que contendrán los campos que se desea que sean comparados para formar parte del tráfico de la red como se muestra a continuación:

```
OFMatch match =OFMatch.load(pi.getPacketData(),
pi.getInPort());

      match.setDataLayerType(Ethernet.TYPE_ARP);
      match.setInputPort((short)3);
      match.setNetworkDestination(IPv4.toIPv4Address(ip3))
```

Un factor muy importante a tomar en cuenta en el controlador son las wildcards ya que son necesarias definirlas para determinar que campos intervendrán en el proceso de comparación por lo cual emplearemos el comando setWildcards.

Existen varias maneras de establecer las wildcards una de ellas es incluir que campos queremos que formen parte del flujo mediante el conector (^) o excluir aquellos que no queremos que formen parte del mismo mediante el conector (|) como se ha realizado en este módulo.

```
match.setWildcards(OFMatch.OFPFW_DL_DST|
                   OFMatch.OFPFW_DL_SRC|
```

```
OFMatch.OFPFW_DL_VLAN|
OFMatch.OFPFW_DL_VLAN_PCP|
OFMatch.OFPFW_NW_PROTO|
OFMatch.OFPFW_NW_SRC_ALL|
OFMatch.OFPFW_NW_TOS|
OFMatch.OFPFW_TP_DST|
OFMatch.OFPFW_TP_SRC);
```

Una vez que se han definido los campos del objeto Match se debe agregar todo este objeto al objeto Flowmod por lo cual se llama al comando setMatch el cual nos permite realizar esta acción.

```
fm.setMatch(match);
```

6. Determinación de las acciones establecidas para un flujo

Para establecer las acciones en el controlador se debe crear un objeto Action dentro en el cual se define que acción se realizará con el flujo, en este caso se envía los paquetes por el puerto 1 del switch.

```
OFAction action=new OFActionOutput((short)1);
fm.setActions(Collections.singletonList((OFAction)action));
```

Como último paso se procede a escribir la regla en el switch mediante el comando **sw.getOutputStream().write(fm);** enviando como argumento la variable que contiene la información del Objeto FlowMod denominada como **fm.**

```
sw.getOutputStream().write(fm);
```

El proceso que se realizó en el módulo para la inserción de las reglas en los otros switches es esencialmente el mismo solamente se modificó la información de los campos del objeto match acorde a lo necesitado en el proyecto la totalidad del código de este módulo se muestra en el anexo 1 [5] [20] [46] [41] [42] [43] [44].

- **Modulo filtrado por Mac.**

Este módulo se modificó a partir del módulo learning switch contenido en Beacon en el directorio net.beaconcontroller.tutorial desarrollado por David Erickson. Este módulo brinda la funcionalidad de un switch el cual realiza una inundación de paquetes la primera vez que el flujo llega al controlador este proceso se realiza para poder recuperar la dirección mac y el puerto de destino del paquete una vez que se han recuperado estos datos ya no es necesario enviar paquetes a través de todas la interfaces del switch sino que solamente se envía al host de destino por la interfaz asignada a este, lo cual evita colisiones y retardos.

En adición a este proceso se ha modificado el código de este módulo con el objetivo de permitir esta comunicación pero únicamente entre los host que el administrador desee habilitar a través de sus direcciones Mac a continuación se describirán las partes más importantes de este módulo.

1. Creación del Objeto Packet-In y llamado de los métodos initMACTable y FiltroMac.

```
public Command receive(IOFSwitch sw, OFMessage msg) throws
IOException {

            initMACTable(sw);
            OFPacketIn pi = (OFPacketIn) msg;
            FiltroMac(sw,pi);
            return Command.CONTINUE;
        }
```

2. Creación del método initMACTable

```
private void initMACTable(IOFSwitch sw) {
Map<Long,Short> macTable = macTables.get(sw);

        if (macTable == null) {
            macTable = new HashMap<Long,Short>();
            macTables.put(sw, macTable);
          }
    }
```

Este método permite crear una tabla que almacena las direcciones mac recibidas cuando un paquete ingresa al controlador así como el puerto por el

cual ingreso con la finalidad de que el switch aprenda estos puertos al momentos enviar paquetes.

3. Creación del método FiltroMac.

En este método se realizará el código encargado de realizar la inserción de reglas en los switch en dependencia de si se habilitó o no el acceso a un determinado host.

```
public void FiltroMac(IOFSwitch sw, OFPacketIn pi) throws
IOException
    {
```

4. Iniciación de la tabla de direcciones Mac.

```
Map<Long,Short> macTable =macTables.get(sw);
```

5. Creación del Objeto match para recuperar el puerto de entrada del paquete y la dirección Mac.

```
OFMatch match = OFMatch.load(pi.getPacketData(),
pi.getInPort());

String MAC_ORIGEN
=HexString.toHexString(match.getDataLayerSource());
```

6. Almacenamiento de las direcciones mac de los host que se habilitará

```
String M1="00:1c:c0:de:08:2d";
String M2="00:1c:c0:55:2e:b4";
String M3="00:1c:c0:de:0a:e1";
```

7. Creación de la condición para evaluar las direcciones mac habilitadas

```
if ((M1.equals( MAC_ORIGEN )||M2.equals( MAC_ORIGEN
)||M3.equals( MAC_ORIGEN )))
{
```

8. Adición del puerto de las direcciones Mac designadas para establecer comunicación.

```
macTable.put(Ethernet.toLong(match.getDataLayerSource()),
pi.getInPort());
```

9. Recuperar el puerto previamente aprendido para la dirección MAC destino

del paquete.

```
Short puerto_salida=
macTable.get(Ethernet.toLong(match.getDataLayerDestination())
);
```

10. Evaluación de los puertos almacenados.

Si no se ha almacenado ningún puerto realizar una inundación de paquetes para determinar cuál es el host destino.

```
if (puerto_salida==null)
{

    OFPacketOut po=new OFPacketOut
    OFAction action = new
    OFActionOutput(OFPort.OFPP_FLOOD.getValue());
    po.setActions(Collections.singletonList(action));
    po.setInPort(pi.getInPort());

    po.setBufferId(pi.getBufferId());

    if (pi.getBufferId()==OFPacketOut.BUFFER_ID_NONE)
    {
        po.setPacketData((pi.getPacketData()));
    }
    sw.getOutputStream().write(po);
}
```

Si el puerto destino ya se conoce enviar el flujo hacia él

```
else
{
   OFFlowMod fm = new OFFlowMod();
   fm.setBufferId(pi.getBufferId());
   fm.setCommand(OFFlowMod.OFPFC_ADD);
   fm.setIdleTimeout((short) 15);
   fm.setMatch(match);

  OFAction action = new OFActionOutput(puerto_salida);
  fm.setActions(Collections.singletonList((OFAction)action));

   sw.getOutputStream().write(fm);

   if (pi.getBufferId() == OFPacketOut.BUFFER_ID_NONE)
    {
       OFPacketOut po = new OFPacketOut();
       action = new OFActionOutput(puerto_salida);
       po.setActions(Collections.singletonList(action);
       po.setBufferId(OFPacketOut.BUFFER_ID_NONE);
       po.setInPort(pi.getInPort());
```

```
            po.setPacketData(pi.getPacketData());
            sw.getOutputStream().write(po);
        }
```

La totalidad del código se encuentra en el anexo 2 [42] [43] [44] [45].

4.5.3 Creación de módulos en Floodlight

A diferencia de Beacon Floodlight se distingue por su sencillez para crear módulos y utilizar dependencias.

Para crear un nuevo proyecto en Floodlight se debe realizar los siguientes pasos:

1. Click derecho en la carpeta **src/main/java**

2. Crear una nueva clase tal como se muestra en la figura 4.36.

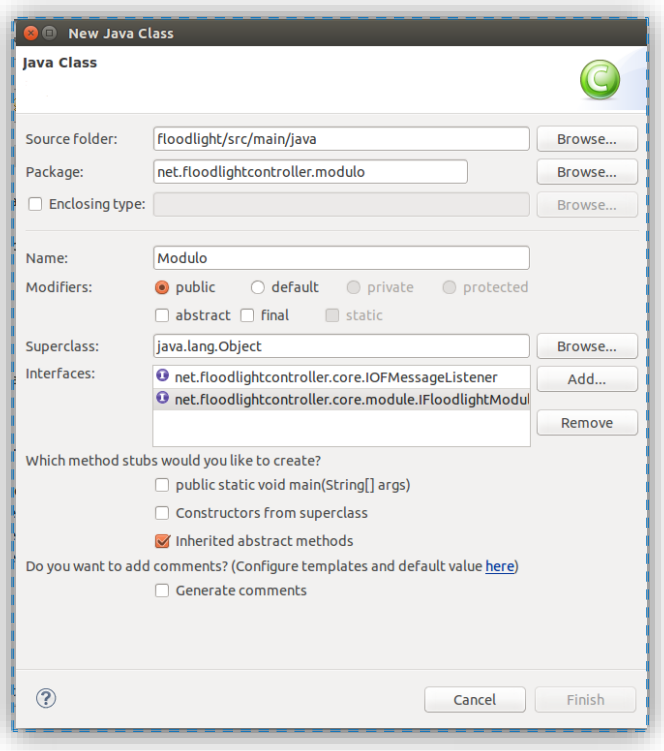


Fig. 4. 36 Creación de un nuevo proyecto en Floodlight

3. Registrar el nombre del paquete que se encuentra bajo el directorio **net.floodlightcontroller.[nombre del modulo].** Así como el nombre de la

clase.

4. Añadir las interfaces IOFMessageListener e IFloodlightModule.

5. Registrar el módulo para que podamos ejecutarlo para ello deberemos dirigirnos a la carpeta **src/main/resources** y localizar el archivo **net.floodlightcontroller.core.module.IFloodlightModule** donde se encuentran registrados todos los módulos existentes en el proyecto Floodlight como se muestra en la figura 4.37.

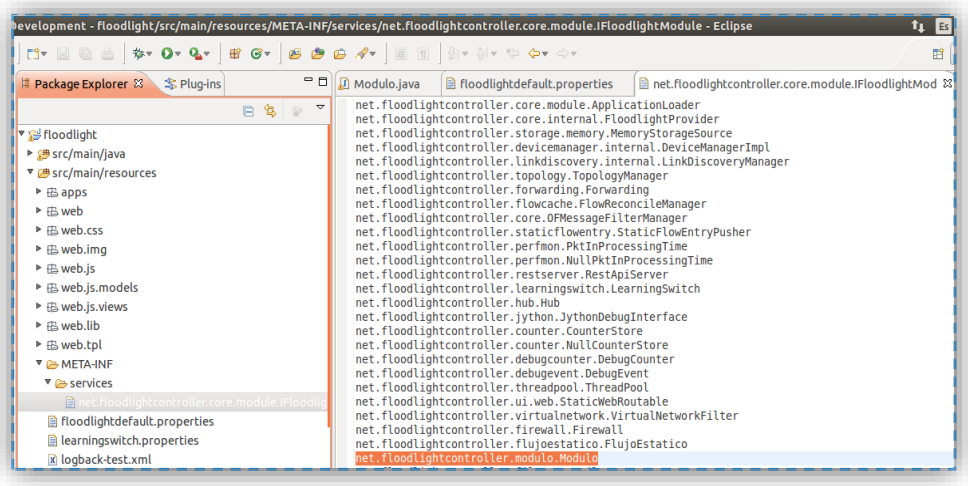


Fig. 4. 37 Registro del proyecto en el archivo IFloodlightModule

6. Registrar el módulo dentro del archivo **floodlightdefault.properties** en el cual se encuentran todos los módulos que arrancaran al momento de ejecutar Floodlight en eclipse como se muestra en la figura 4.38 [36] [47].

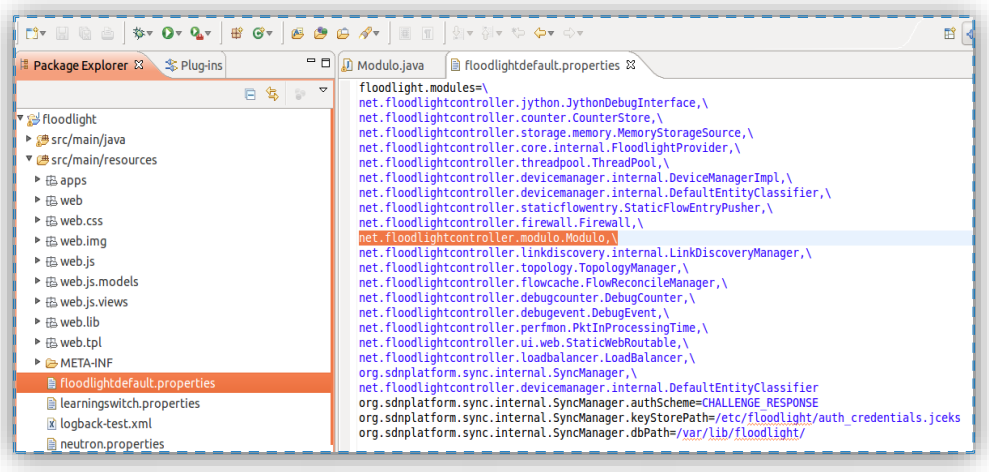


Fig. 4. 38 Registro del proyecto en el archivo floodlightdefault.properties

4.5.4 Descripción de los módulos desarrollados en Floodlight

- **Modulo Filtro Mac a través de REST API.**

En Floodlight se ha creado el módulo de Filtrado por dirección Mac el mismo que se realizó en Beacon con la particularidad de que este módulo se encuentra expuesto a través de REST API. El servicio rest permite que el administrador de red pueda interactuar con el módulo desde una aplicación externa a través del comando curl. Esta es una utilidad que no se encuentra disponible en Beacon y brinda al administrador un mejor control de la red al poder interactuar con el controlador una vez que este se encuentre en ejecución añadiendo o modificando ciertos parámetros del módulo en función de la información que se le proporcione [48] [49].

Para el desarrollo de este módulo se realizó los siguientes pasos que se describen a continuación:

- Proceso de creación del módulo

1. Click derecho dentro de la carpeta **src/main/java**.

2. Seleccionar **New/Class**.

3. Añadir las interfaces IFloodlightModule e IOFMessageListener.

   Estas interfaces al igual que en Beacon sirven para interactuar con los switches y los mensajes OpenFlow. En la figura 4.39 se muestra este proceso.

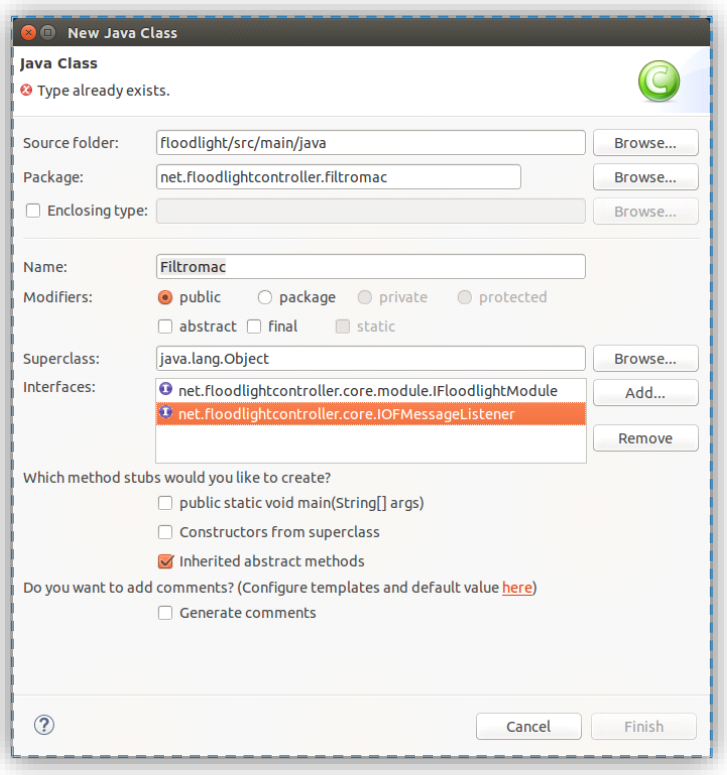


Fig. 4. 39 Creación del módulo Filtro Mac

- Establecimiento de las dependencias

1. Declarar la variable **floodlightProvider** para establecer la comunicación con los switches OpenFlow.

```
public class Filtromac implements IFloodlightModule,
IOFMessageListener {
protected IFloodlightProviderService floodlightProvider;
```

2. Conectar el módulo al cargador de módulos añadiendo el siguiente código al método **getModule Service().**

```
@Override
public Collection<Class<? extends IFloodlightService>>
getModuleServices() {
```

```
        Collection<Class<? extends IFloodlightService>> l =
        new ArrayList<Class<? extends IFloodlightService>>();
        l.add(IFiltromacService.class);
        return l;
}
```

3. Creación del código para adquisición y manejo de las direcciones Mac

```
@Override
public String getBuffer() {
        String ret;
        String key;

        // "{\"key1\":\"value1\",\"key2\":\"value2\"}"
        ret = "{";
        for (int i=0; i < buffer.size(); i++) {
                key = buffer.get(i);
                ret += "\"" + key + "\""+ ":";
                ret += "\"\"";
                if (i < (buffer.size() - 1))
                        ret += ",";
        }
        ret += "}";
        return ret;
}

@Override
public void showBuffer() {
        for (String entry: buffer) {
                System.out.println(entry);
        }

}

@Override
public void setBuffer(String key) {
        buffer.add(key);
}

@Override
public void removeEntry(String key) {
        for (int i=0; i < buffer.size(); i++) {
                if ((buffer.get(i)).equals(key)) {
                        buffer.remove(i);
                        return;
                }
        }
}

@Override
public void removeAllEntries() {
        buffer.clear();
}
}
```

- Creación del archivo FiltromacResource encargado de manejar las peticiones curl.

1. Seleccionar el paquete "net.floodlightcontroller.filtromac".

2. Click derecho y seleccionar **New/Class**.

3. Establecer el nombre a **FiltromacResource** y luego finalizar.

   En la figura 4.40 se muestra este proceso

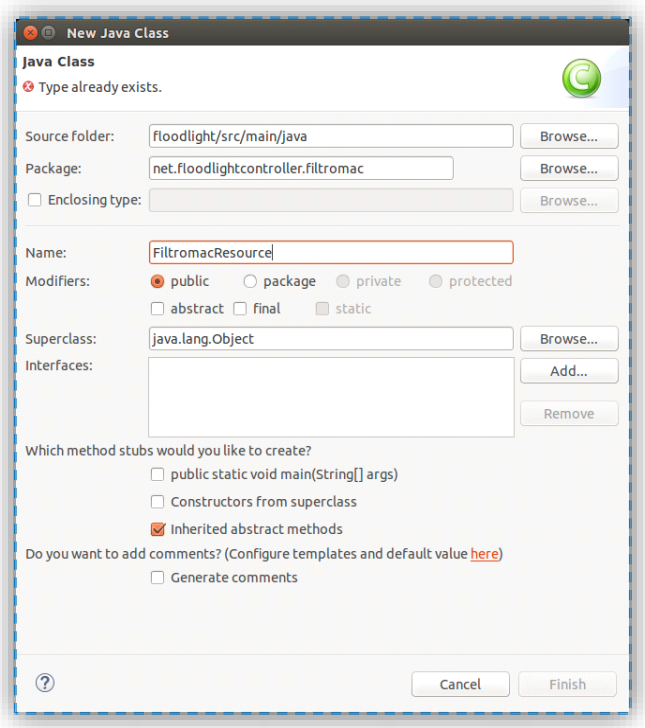


Fig. 4. 40 Creación del archivo FiltromacResource

4. Creación del código para manejar las peticiones curl

```
public class FiltromacResource extends ServerResource {

   // Obtener la solicitud y devolver los datos
   @Get("json")
   public Object handleReqeust() {
         IFiltromacService filtro =

   (IFiltromacService)getContext().getAttributes().

   get(IFiltromacService.class.getCanonicalName());
```

```
            return filtro.getBuffer();
      }

      @Post
      public String handlePost(String str) {
            String key,value;
            IFiltromacService filtro =

      (IFiltromacService)getContext().getAttributes().

      get(IFiltromacService.class.getCanonicalName());
            MappingJsonFactory f = new MappingJsonFactory();
            JsonParser jp;

            try {
                  jp = f.createJsonParser(str);
                  jp.nextToken();
                  while (jp.nextToken() != JsonToken.END_OBJECT)
{
                        key = jp.getCurrentName();
                        jp.nextToken();
                        filtro.setBuffer(key);
                  }
            } catch (IOException e) {
                  System.err.println(e);
            }

            return "{\"add\":\"success\"}";
      }
      @Delete
      public String handleDelete(String str) {
            String identifier, key;
            IFiltromacService filtro =

      (IFiltromacService)getContext().getAttributes().

      get(IFiltromacService.class.getCanonicalName());

            MappingJsonFactory f = new MappingJsonFactory();
            JsonParser jp;

            try {
                  jp = f.createJsonParser(str);
                  jp.nextToken();
                  while (jp.nextToken() != JsonToken.END_OBJECT)
{
                        identifier = jp.getCurrentName();
                        jp.nextToken();
                        key = jp.getText();
                        filtro.removeEntry(key);
                  }
            } catch (IOException e) {
                  System.err.println(e);
            }

            return "{\"delete\":\"success\"}";
      }
}
```

- Creación del archivo FiltomacWebRoutable.java.

1. Seleccionar el paquete "net.floodlightcontroller.filtromac".

2. Click derecho **New/Class**.

3. Establecer el nombre a **FiltomacWebRoutable** y luego finalizar.

En esta parte del código se establece la sintaxis para ejecutar los comandos que se enviaran al momento de ingresar, eliminar o borrar una dirección mac.

```
package net.floodlightcontroller.filtromac;

import org.restlet.Context;
import org.restlet.Restlet;
import org.restlet.routing.Router;

import net.floodlightcontroller.restserver.RestletRoutable;

public class FiltromacWebRoutable implements RestletRoutable
{

   @Override
   public Restlet getRestlet(Context context) {
         Router router = new Router(context);
         router.attach("/addmac", FiltromacResource.class);
         router.attach("/clear",
FiltromacClearResource.class);
         return router;
   }

   @Override
   public String basePath() {
         return "/wm/filtro";
   }
}
```

- Creación de la interfaz denominada IFiltromacService

1. Seleccionar el paquete "net.floodlightcontroller.filtromac".

2. Click derecho **New/Interfaz**.

3. Establecer el nombre a **IFiltromacService** y luego finalizar.

En esta interfaz se definen los métodos que nos permitirán obtener, añadir y borrar las direcciones mac de la lista y se muestran a continuación:

```
package net.floodlightcontroller.filtromac;

import
net.floodlightcontroller.core.module.IFloodlightService;

public interface IFiltromacService extends IFloodlightService
{

   //función para visualizar las direcciones mac
   public String getBuffer();
   //función para añadir las direcciones mac
   public void setBuffer(String key);
   //función para remover una dirección mac
   public void removeEntry(String key);
   //función para borrar todas las direcciones mac
   public void removeAllEntries();
   public void showBuffer();
}
```

- Crear la clase denominada FiltromacClearResource

1. Seleccionar el paquete "net.floodlightcontroller.filtromac".

2. Click derecho **New/Class**.

3. Establecer el nombre a **FiltromacClearResource** y luego finalizar.

Esta clase permite borrar la lista de las todas las direcciones mac almacenadas

```
package net.floodlightcontroller.filtromac;

import org.restlet.resource.Get;
import org.restlet.resource.ServerResource;

public class FiltromacClearResource extends ServerResource {

   // get request and remove all entries in a buffer
   @Get("json")
   public Object handleRequest() {
         IFiltromacService filtro =

   (IFiltromacService)getContext().getAttributes().

   get(IFiltromacService.class.getCanonicalName());

         filtro.removeAllEntries();
         return "{\"delete all\":\"success\"}";
   }
}
```

- Comandos usados para la visualización, adición y eliminación de direcciones Mac.

  Visualizar direcciones mac almacenadas

  curl –s http://localhost:8080/wm/filtro/addmac | python -mjson.tool

  Almacenar direcciones mac

  curl -X POST -d '{"key1":"value1","key2":"value2"}'
  http://localhost:8080/wm/filtro/addmac

  Borrar los datos uno por uno o varios a la vez

  curl -X DELETE -d '{"key":"value1"}'
  http://localhost:8080/wm/filtro/addmac
  curl -X DELETE –d
  '{"key1":"value1","key2":"value2"}' http://localhost:8080/wm/filtro/addmac

  Borrar toda la información de la lista

  curl http://localhost:8080/wm/filtro/clear

El código de todos los archivos de este módulo se encuentra en el anexo 3

♦ **Módulo de inserción de reglas estáticas en Floodlight.**

Este módulo se creó con la finalidad de familiarizarse con las sentencias de programación de Floodlight, a pesar de que son muy similares a las de Beacon existen algunas consideraciones que se deben tomar en cuenta. Tales como:

1. La manera de establecer las acciones.

Para establecer acciones en Floodlight es necesario crear un array list y declarar la lista que vamos a utilizar algo que no es necesario en Beacon. A continuación se muestra este código:

```
List<OFAction> action = new ArrayList<OFAction>();
```

```
OFAction actions=new OFActionOutput((short) 1);

fm.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionOutput.
MINIMUM_LENGTH));

action.add(actions);

fm.setActions(action);
```

2. Envió de la regla de flujo al switch

En Floodlight se debe enviar la regla de flujo dentro de la instrucción try para evitar errores: mientras que Beacon esto no es necesario. A continuación se muestra el código para enviar la entrada de flujo.

```
try {

        sw.write(fm, null);

   } catch (IOException e) {

        e.printStackTrace();

   }
```

El código de este módulo se encuentra en el anexo 4.

- **Módulo Balanceo de carga desarrollado en Floodlight.**

Este módulo fue desarrollado por la universidad de Washington y se lo levanto con la finalidad de observar el comportamiento de un balanceador de carga en una red SDN.

Este módulo funciona de la siguiente manera: cuando un dispositivo quiere establecer comunicación debe hacerlo a través de la dirección IP del balanceador de carga el cual responderá a estas peticiones. Esta dirección IP en realidad es una IP lógica por lo cual físicamente no existe, este balanceador de carga también debe tener una dirección MAC lógica cuando el dispositivo establece conexión con balanceador de carga este modifica la dirección ip de destino para direccionar dichas peticiones a uno de los dos servidores reales los cuales responderán estas solicitudes y estarán definidos como los 2 primeros host en la topología que se

utilizará con las direcciones mac 00:1c:c0:de:08:31, 00:1c:c0:de:0a:e1 y con las direcciones ip 192.168.111.13/24, 192.168.111.14/24 mientras que la dirección lógica del balanceador de carga será la 192.168.111.254 y la mac 00:00:00:00:00: FE.

Es importante definir que como la ip del balanceador de carga es una ip lógica es necesario asignar rutas de manera estática en los hosts manualmente [50] [51] [52] [53] [54] [55].

A continuación se describe las partes más importantes de este módulo:

1. Creación de la IP y dirección MAC para el balanceador de carga

```
private final static int LOAD_BALANCER_IP =
IPv4.toIPv4Address("192.168.111.254");
private final static byte[] LOAD_BALANCER_MAC =
Ethernet.toMACAddress("00:00:00:00:00:FE");
```

2. Creación de las variables y funciones para las direcciones IP, MAC y puerto del servidor.

```
private static class Server{

      private int ip;
      private byte[] mac;
      private short port;

      public Server(String ip, String mac, short port) {
            this.ip = IPv4.toIPv4Address(ip);
            this.mac = Ethernet.toMACAddress(mac);
            this.port = port;
      }

      public int getIP() {
            return this.ip;
      }

      public byte[] getMAC() {
            return this.mac;
      }

      public short getPort() {
            return this.port;
      }
}
```

3. Establecimiento de las direcciones IP, MAC y puertos de los servidores

```
final static Server[] SERVERS = {
new Server("192.168.111.13", "00:1c:c0:de:08:31",(short)1),
new Server("192.168.111.14", "00:1c:c0:de:0a:e1",(short)2)
};
```

4. Creación del contador para controlar el número de peticiones de los servidores.

```
private int lastServer = 0;
```

5. Evaluaciones de las peticiones Mac e IP dirigidas al balanceador de carga.

```
if (match.getDataLayerType() != Ethernet.TYPE_IPv4 &&
match.getDataLayerType() != Ethernet.TYPE_ARP) {
      return Command.CONTINUE;
      }
      if (match.getNetworkDestination() !=
      LOAD_BALANCER_IP) {
          return Command.CONTINUE;
      }

      if (match.getDataLayerType() == Ethernet.TYPE_ARP) {

            logger.info("Petcion ARP dirigida al
            balanceador de carga recivida ");
                  handleARPRequest(sw, pi, cntx);
              } else {
                  logger.info("Paquete IPv4 distinado al
                  balanceador de carga recibido ");
                  loadBalanceFlow(sw, pi, cntx);
              }

                  return Command.STOP;
      }
```

6. Envió del mensaje Packet-out al switch.

```
private void pushPacket(IOFSwitch sw, OFPacketIn pi,
ArrayList<OFAction> actions, short actionsLength) {


OFPacketOut po = (OFPacketOut)
floodlightProvider.getOFMessageFactory().getMessage(OFType.PA
CKET_OUT);

     po.setInPort(pi.getInPort());
     po.setBufferId(pi.getBufferId());

     po.setActions(actions);
     po.setActionsLength(actionsLength);

  if (pi.getBufferId() == OFPacketOut.BUFFER_ID_NONE) {
```

```
                byte[] packetData = pi.getPacketData();
                po.setLength(U16.t(OFPacketOut.MINIMUM_LENGTH
                 + po.getActionsLength() + packetData.length));
                po.setPacketData(packetData);
    } else {
                po.setLength(U16.t(OFPacketOut.MINIMUM_LENGTH
                + po.getActionsLength()));
    }

    try {
        sw.write(po, null);
    } catch (IOException e) {
        logger.error("failed to write packetOut: ", e);
    }
}
```

7. Manejar las solicitudes ARP y responderlas con la dirección mac del balanceador de carga.

```
private void handleARPRequest(IOFSwitch sw, OFPacketIn pi,
FloodlightContext cntx) {

   logger.debug("Handle ARP request");
   Ethernet eth =
   IFloodlightProviderService.bcStore.get(cntx,IFloodlightProv
   iderService.CONTEXT_PI_PAYLOAD);
   System.out.println(eth);
   if (! (eth.getPayload() instanceof ARP))
         return;
   ARP arpRequest = (ARP) eth.getPayload();
   IPacket arpReply = new Ethernet()
   .setSourceMACAddress(LoadBalancer.LOAD_BALANCER_MAC)
   .setDestinationMACAddress(eth.getSourceMACAddress())
   .setEtherType(Ethernet.TYPE_ARP)
   .setPriorityCode(eth.getPriorityCode())
   .setPayload(new ARP()
         .setHardwareType(ARP.HW_TYPE_ETHERNET)
         .setProtocolType(ARP.PROTO_TYPE_IP)
         .setHardwareAddressLength((byte) 6)
         .setProtocolAddressLength((byte) 4)
         .setOpCode(ARP.OP_REPLY)
         .setSenderHardwareAddress(LoadBalancer.LOAD_BALANCER_
         MAC).setSenderProtocolAddress(LoadBalancer.LOAD_BALAN
         CER_IP).setTargetHardwareAddress(arpRequest.getSender
         HardwareAddress()).setTargetProtocolAddress(arpReques
         t.getSenderProtocolAddress()));

   sendARPReply(arpReply, sw, OFPort.OFPP_NONE.getValue(),
   pi.getInPort());
   }
```

8. Envió de la respuesta ARP al switch

```
private void sendARPReply(IPacket packet, IOFSwitch sw, short
inPort, short outPort) {

        OFPacketOut po = (OFPacketOut)
        floodlightProvider.getOFMessageFactory()
                    .getMessage(OFType.PACKET_OUT);
        po.setBufferId(OFPacketOut.BUFFER_ID_NONE);
        po.setInPort(inPort);

        List<OFAction> actions = new ArrayList<OFAction>();
        actions.add(new OFActionOutput(outPort, (short)
        0xffff));
        po.setActions(actions);
        po.setActionsLength((short)
        OFActionOutput.MINIMUM_LENGTH);

        byte[] packetData = packet.serialize();
        po.setPacketData(packetData);
        po.setLength((short) (OFPacketOut.MINIMUM_LENGTH +
        po.getActionsLength() + packetData.length));

        try {
              sw.write(po, null);
              sw.flush();
        } catch (IOException e) {
              logger.error("error al escribir el mensaje
              packet-out", e);
        }
}
```

9. Realización del balanceo de carga basado en el mensaje Packet-in de un paquete IPv4 destinado al balanceador de carga lógico.

```
private void loadBalanceFlow(IOFSwitch sw, OFPacketIn pi,
FloodlightContext cntx) {

   Server server = getNextServer();
   Ethernet eth = IFloodlightProviderService.bcStore.get(cntx,
   IFloodlightProviderService.CONTEXT_PI_PAYLOAD);

        OFFlowMod rule = new OFFlowMod();
        rule.setType(OFType.FLOW_MOD);
        rule.setCommand(OFFlowMod.OFPFC_ADD);

        OFMatch match = new OFMatch()
        .setDataLayerDestination(LOAD_BALANCER_MAC)
        .setDataLayerSource(eth.getSourceMACAddress())
        .setDataLayerType(Ethernet.TYPE_IPv4)
        .setNetworkDestination(LOAD_BALANCER_IP)
        .setNetworkSource(((IPv4)
        eth.getPayload()).getSourceAddress())
        .setInputPort(pi.getInPort());

        match.setWildcards(OFMatch.OFPFW_NW_PROTO);
```

```
rule.setMatch(match);

rule.setIdleTimeout(IDLE_TIMEOUT);
rule.setHardTimeout(HARD_TIMEOUT);

rule.setBufferId(OFPacketOut.BUFFER_ID_NONE);

ArrayList<OFAction> actions = new
ArrayList<OFAction>();

OFAction rewriteMAC = new
OFActionDataLayerDestination(server.getMAC());
actions.add(rewriteMAC);

OFAction rewriteIP = new
OFActionNetworkLayerDestination(server.getIP());
actions.add(rewriteIP);


OFAction outputTo = new
OFActionOutput(server.getPort());
actions.add(outputTo);

rule.setActions(actions);
short actionsLength =
(short)(OFActionDataLayerDestination.MINIMUM_LENGTH
+ OFActionNetworkLayerDestination.MINIMUM_LENGTH+
OFActionOutput.MINIMUM_LENGTH);

rule.setLength((short) (OFFlowMod.MINIMUM_LENGTH +
actionsLength));

logger.debug("Actions length="+ (rule.getLength() -
OFFlowMod.MINIMUM_LENGTH));

logger.debug("Instalar regla para dirección de avance
para el flujo: " + rule);

try {
      sw.write(rule, null);
} catch (Exception e) {
      e.printStackTrace();
}

OFFlowMod reverseRule = new OFFlowMod();
reverseRule.setType(OFType.FLOW_MOD);
reverseRule.setCommand(OFFlowMod.OFPFC_ADD);

OFMatch reverseMatch = new OFMatch()
.setDataLayerSource(server.getMAC())
.setDataLayerDestination(match.getDataLayerSource())
.setDataLayerType(Ethernet.TYPE_IPv4)
.setNetworkSource(server.getIP())
.setNetworkDestination(match.getNetworkSource())
.setInputPort(server.getPort());

reverseMatch.setWildcards(OFMatch.OFPFW_NW_PROTO);
reverseRule.setMatch(reverseMatch);
```

```
            reverseRule.setIdleTimeout(IDLE_TIMEOUT);
            reverseRule.setHardTimeout(HARD_TIMEOUT);

            reverseRule.setBufferId(OFPacketOut.BUFFER_ID_NONE);

            ArrayList<OFAction> reverseActions = new
            ArrayList<OFAction>();

            OFAction reverseRewriteMAC = new
            OFActionDataLayerSource(LOAD_BALANCER_MAC);
            reverseActions.add(reverseRewriteMAC);

            OFAction reverseRewriteIP = new
            OFActionNetworkLayerSource(LOAD_BALANCER_IP);
            reverseActions.add(reverseRewriteIP);

            OFAction reverseOutputTo = new
            OFActionOutput(pi.getInPort());
            reverseActions.add(reverseOutputTo);

            reverseRule.setActions(reverseActions);

            reverseRule.setLength((short)
            (OFFlowMod.MINIMUM_LENGTH      +
            OFActionDataLayerSource.MINIMUM_LENGTH +
            OFActionNetworkLayerSource.MINIMUM_LENGTH+
            OFActionOutput.MINIMUM_LENGTH));

            logger.debug("Instalar regla para dirección inversa
            de flujo: " + reverseRule);

            try {
                  sw.write(reverseRule, null);
            } catch (Exception e) {
                  e.printStackTrace();
            }

            pushPacket(sw, pi, actions, actionsLength);
      }
```

10. Determinación del siguiente servidor al cual el switch debe enviar el flujo

```
      private Server getNextServer() {
            lastServer = (lastServer + 1) % SERVERS.length;
            return SERVERS[lastServer];
      }

}
```

4.6 Simulación del prototipo de la red (SDN) en el software Mininet.

Para fines de investigación se propone una arquitectura híbrida para evaluar el comportamiento de esta red SDN. Creando una topología personalizada en Mininet que cumpla con este propósito la misma que se estableció en la figura 4.1.

Para la creación de esta topología en mininet fue necesario modificar el archivo **topo-2sw-2host.py** contenido en el directorio **~/mininet/custom** contenido en mininet, a partir del cual se puede crear nuevas topologías mediante la adición de switches, hosts y sus respectivos enlaces como se muestra en la figura 4.41. Ya que las opciones para creación de topologías en Mininet descritas en el capítulo II no permiten crear una topología como la que se ha definido [5] [23].

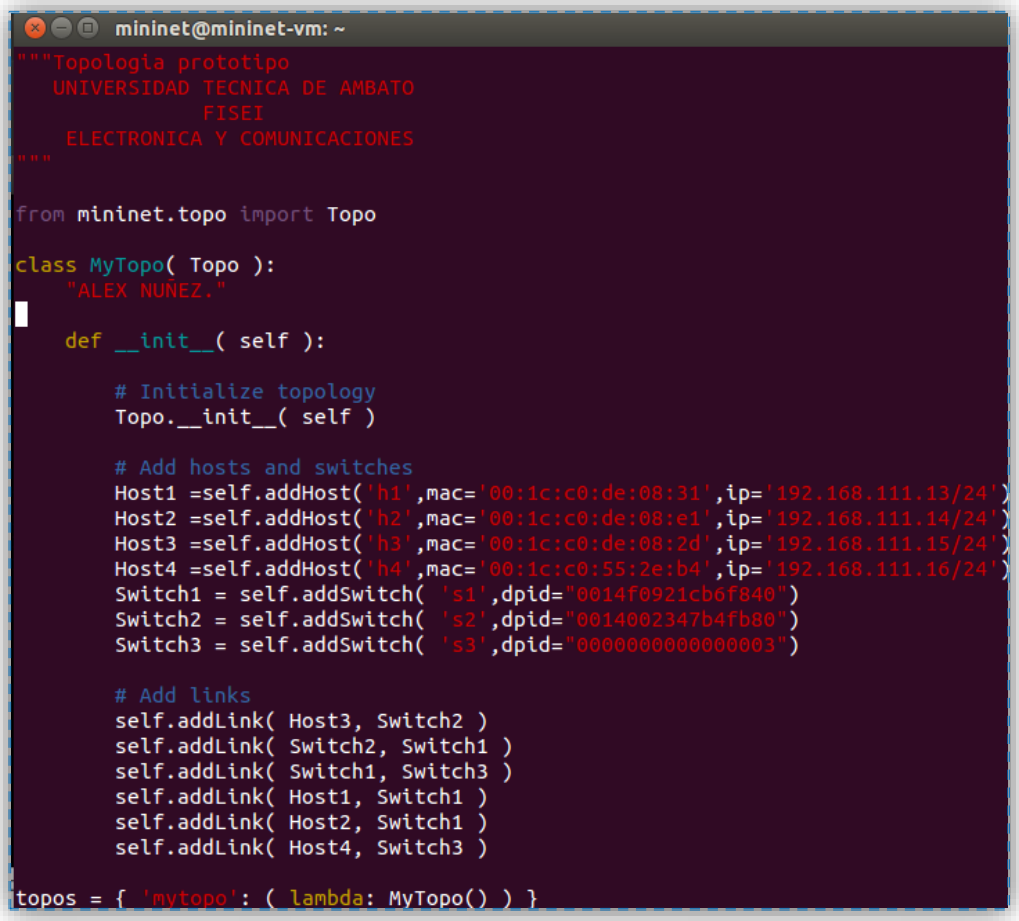


Fig. 4. 41 Creación de topologías personalizadas en Mininet

A esta topología se puede acceder mediante el comando que se describe a continuación:

--sudo mn --custom ~/mininet/custom/<Nombre del archivo>.py --topo mytopo

4.6.1 Pruebas del módulo para inserción de reglas estáticas

Este módulo se ha diseñado para que se otorgue conexión solamente entre el host 1 y el host 3, el host 2 y el host 4 conectados en los puertos de los switches especificados en la topología de la red y que posean las direcciones ip definidas en esta topología ya que el módulo ha sido creado para que inserte las entradas de flujo solamente si estos parámetros son iguales a los que designó en el proceso de comparación o matching.

Al trabajar con mininet es muy importante usar un gestor de ventanas el cual Mininet no posee por defecto, por lo que es necesario ocupar el gestor de ventanas de Ubuntu. Para ello se debe acceder a la máquina virtual donde se encuentra instalado Mininet mediante una conexión ssh la cual se realiza mediante el siguiente comando:

$ ssh –X mininet@<ip de la máquina virtual de mininet>

Estas ventanas son útiles al momento de analizar un determinado dispositivo de la red y se accede al mismo de la siguiente manera:

$ xterm [nombre del dispositivo/s (Ej.: h1, h2, s1 etc.)]

Para poder empezar la simulación es necesario realizar los siguientes pasos:

1. Asignar una dirección IP a mininet mediante el siguiente comando:

   $ sudo ifconfig [interfaz asignada (eth0)] [dirección ip para mininet]/24 up

2. Enlazar el módulo creado a la topología mediante el siguiente comando. Tal como se muestra en la figura 4.42.

   $ --sudo mn --custom ~/mininet/custom/<Nombre del archivo>.py --topo mytopo –controller remote,ip=[dirección IP del controlador].

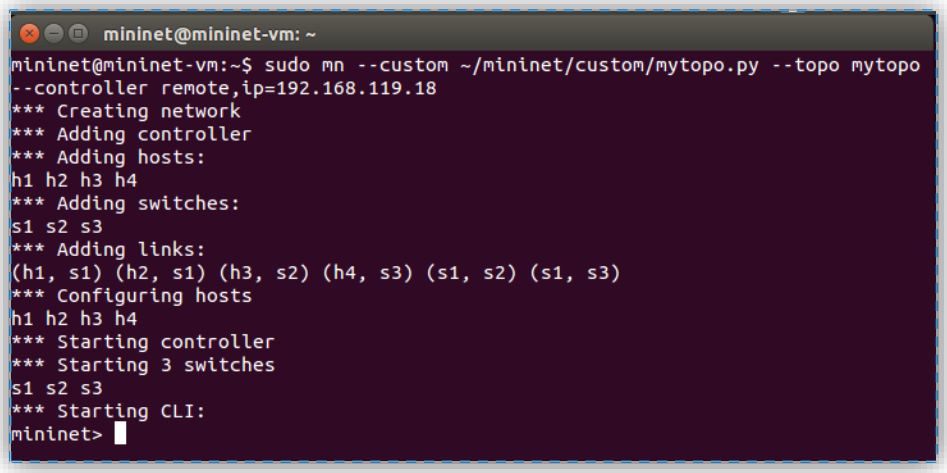


Fig. 4. 42 Conexión del módulo realizado con Mininet

- **Pruebas de conectividad**

Para para revisar que la conexión sea exitosa solamente entre los host que se ha establecido se realiza un **Pingall** el cual permite realizar todas las formas de conectividad entre los host tal como muestra la figura 4.43.

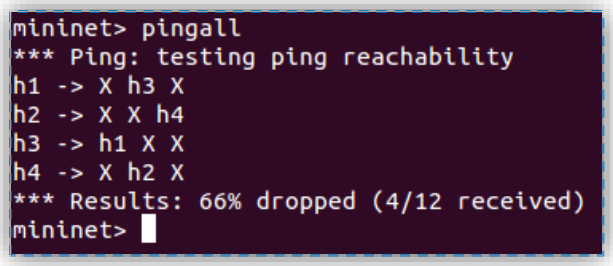


Fig. 4. 43 Pruebas de conexión mediante el comando Pingall

Al observar la figura 4.43 se determina que el módulo funciona de la manera correcta al proveer conexión solamente entre los host asignados, ahora se procederá a observar las reglas que se han establecido en los switches mediante la herramienta dpctl.

Cabe recalcar que en Mininet se hace referencia a los switches mediante el puerto que empieza en el 6634 por lo cual siempre al primer switch se lo hará referencia

mediante este puerto y a los demás se incrementa el valor del puerto en 1, siendo de esta manera el puerto 6635 correspondiente al segundo switch [29] [30] [46].

Para visualizar las reglas insertadas en el switch se usa el siguiente comando:

Dpctl dump-flows tcp:127.0.0.1:6634

En la figuras 4.44, 4.45 y 4.46 se muestra las reglas de flujo insertadas en los tres switches.

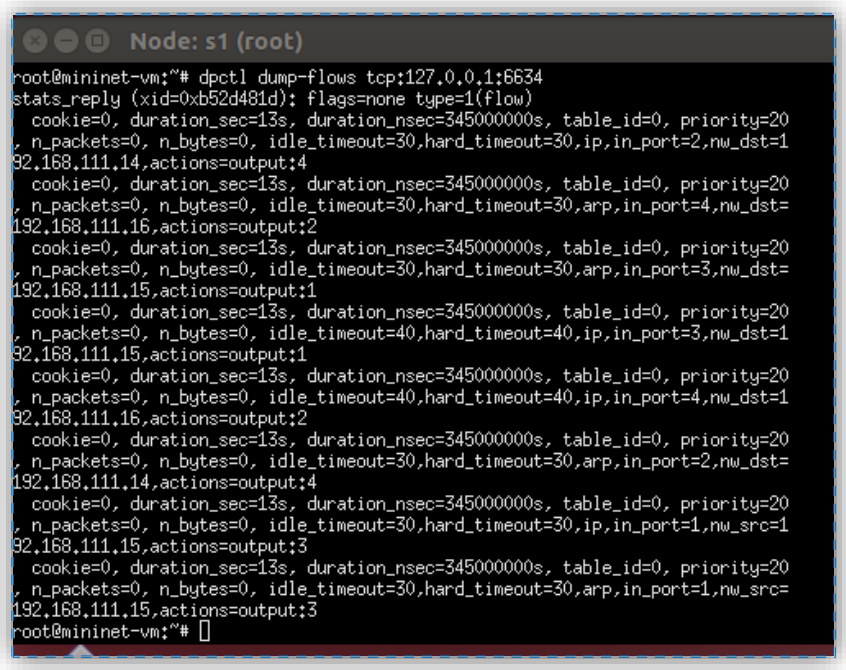


Fig. 4. 44 Visualización de las reglas de flujo insertadas en el switch 1

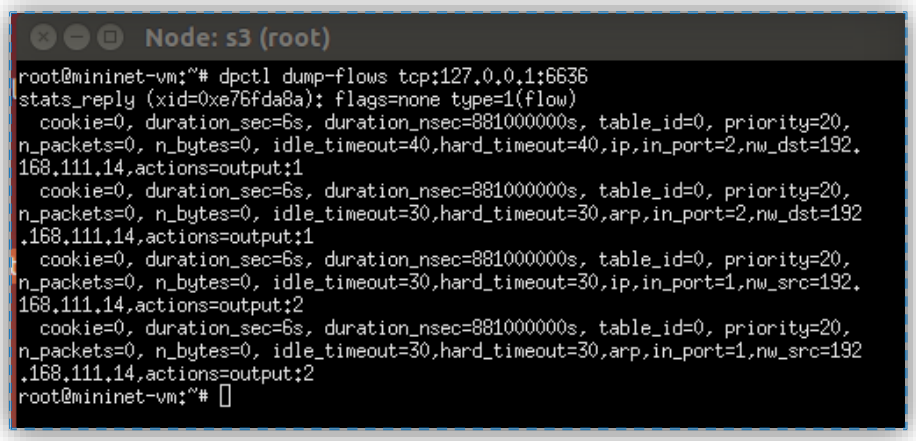


Fig. 4. 45 Visualización de las reglas de flujo insertadas en el switch 2

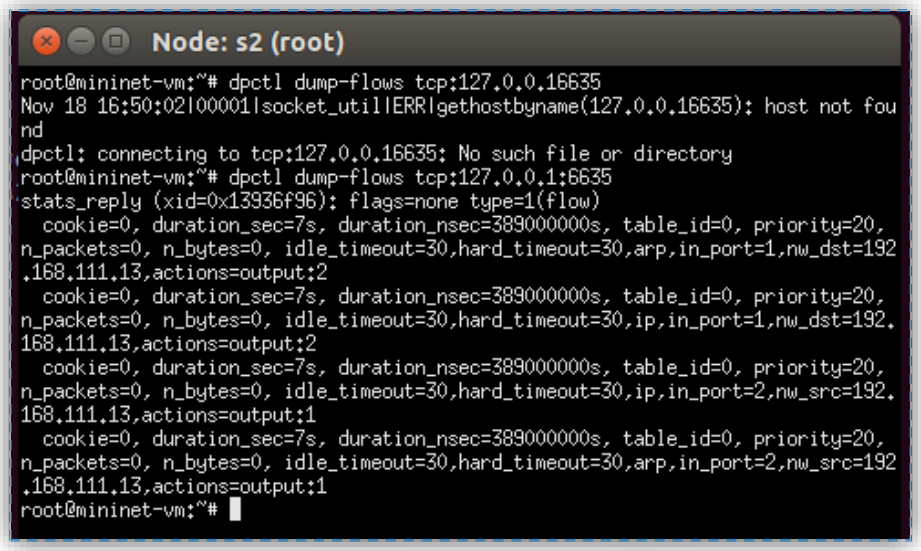


Fig. 4. 46 Visualización de las reglas de flujo insertadas en el switch 3

- **Análisis de los paquetes enviados**

Para el análisis de los paquetes se realizó una prueba de conexión entre el host 1 y 3 y se observó que la conexión entre estos hosts fue exitosa como se muestra en la figura 4.47.

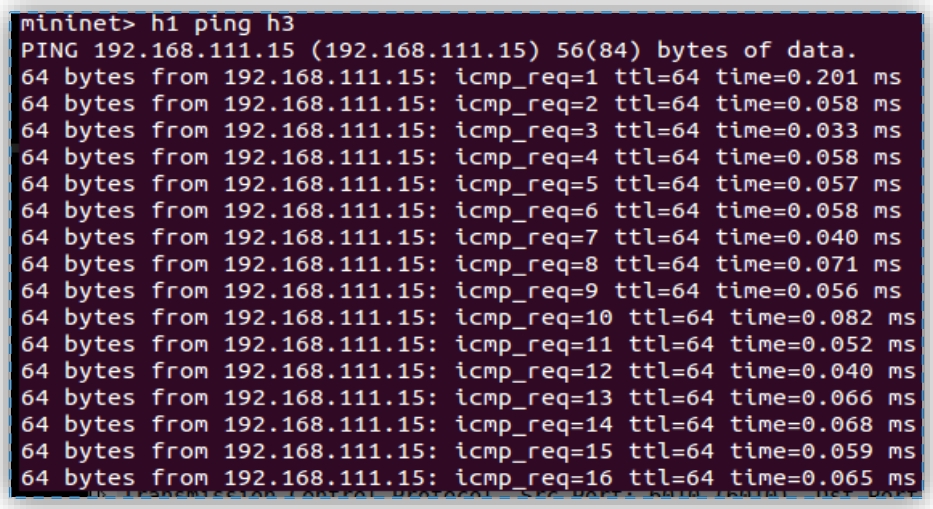


Fig. 4. 47 Conexión entre los host 1 y 3

Se capturó los paquetes en el host 3 con el objetivo de examinar cómo se genera el tráfico tal como se muestra en la figura 4.48, de lo cual observamos que el primer paquete que llega es una solicitud arp preguntando cual maquina posee la dirección 192.168.111.15 como el host 3 posee esta dirección responde a esta solicitud proveyéndole su dirección mac 00:1c:c0:de:08:2d, ahora que el host h1 conoce la dirección mac y el puerto de destino se establece la conexión por lo cual se envían las solicitudes ICMP. En la figura 4.47 se muestra el proceso de envió de paquetes entre los hosts h1 y h3 y luego se realizó una prueba de conexión entre los host 2 y 3 que poseen las direcciones ip 192.168.111.14 y 192.168.111.15 respectivamente y observábamos que solamente se envían las solicitudes arp pero no se envían las respuestas puesto que no se dispuso una regla para que estos dos host puedan establecer conexión.

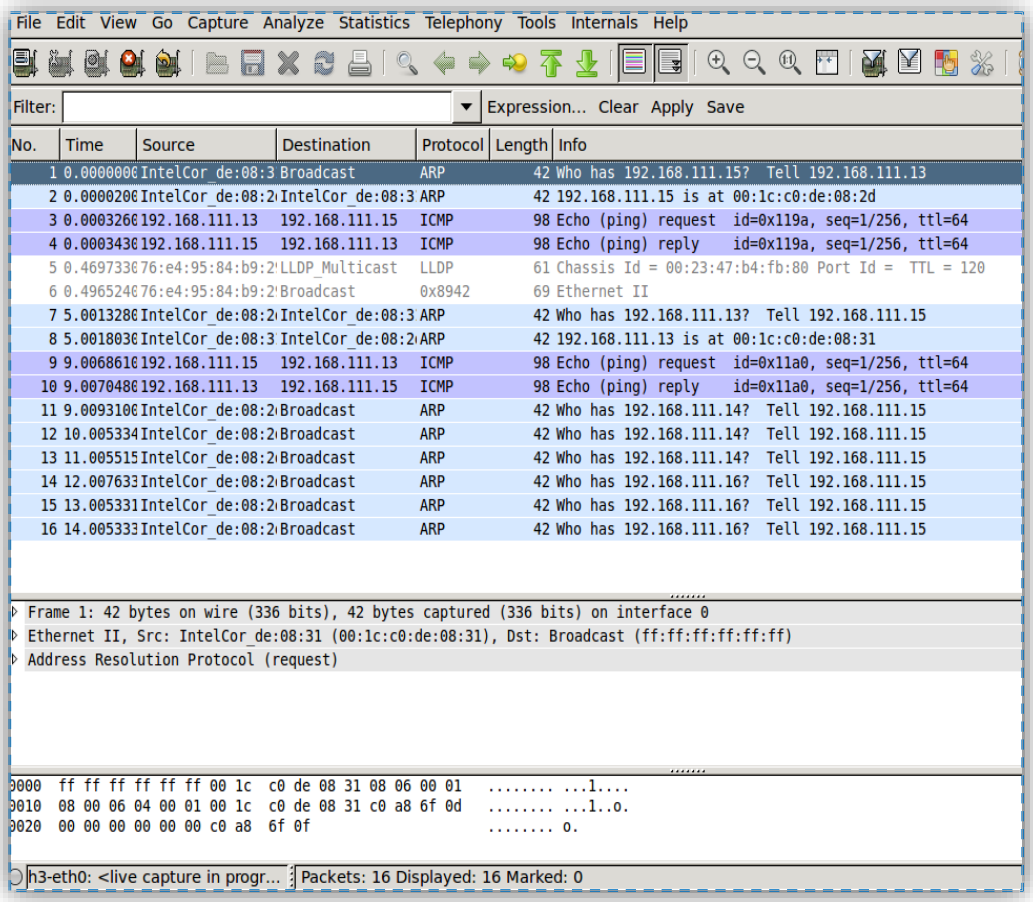


Fig. 4. 48 Captura de paquetes mediante el programa wireshark

Otra alternativa al programa **wireshark** es el comando **tcpdump –n** el cual también nos permite realizar capturas de paquetes dentro del mismo terminal de los host virtuales de Mininet la ventaja de usar es este comando es la comodidad que brinda al no tener que abrir tantas ventanas ya que cuando se trabaja con varios host la falta de espacio en pantalla es uno de los principales problemas. Su desventaja en comparación con wireshark es que la información que podemos obtener es mucho más limitada.

En la figura 4.49 se muestra la utilización de este comando en la captura de paquetes que llegan al host 1.

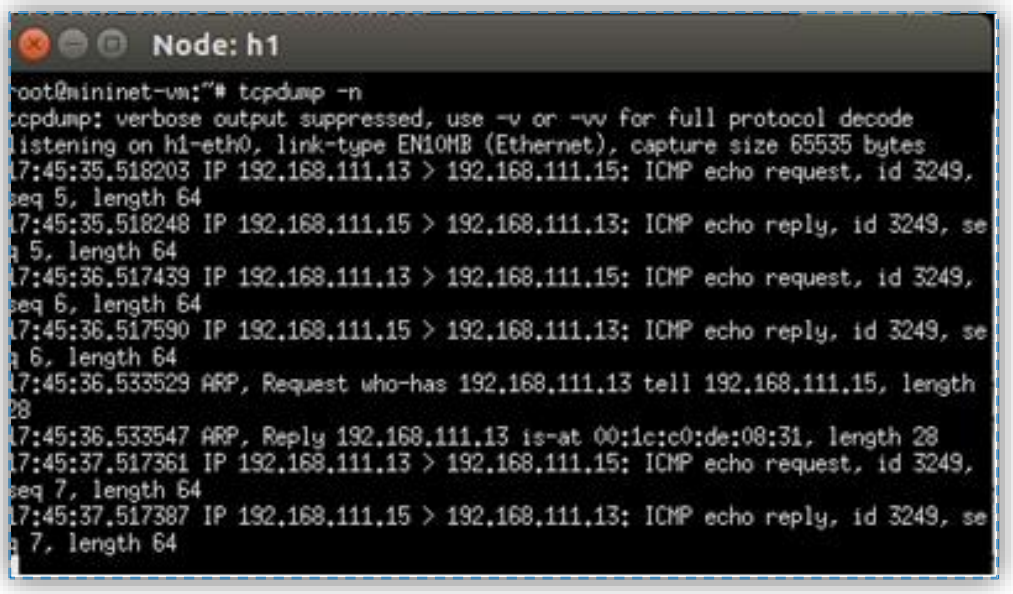


Fig. 4. 49 Captura de paquetes mediante el comando tcpdump –n

- **Revisión de la interfaz web del controlador**

Adicionalmente se verificará como se encuentra la topología en la interfaz web de Floodlight y la información de los dispositivos de la red tal como se muestra en las figuras 4.50 y 4.51.

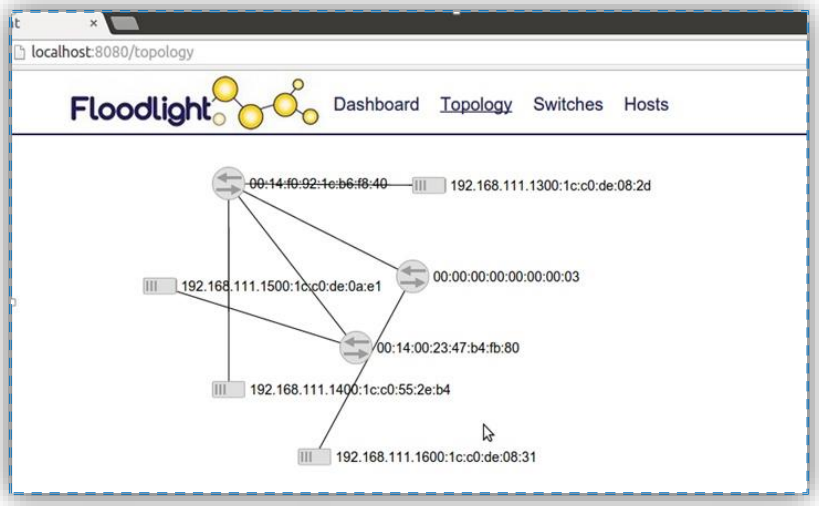


Fig. 4. 50 Topología visualizada en la interfaz Web de Floodlight

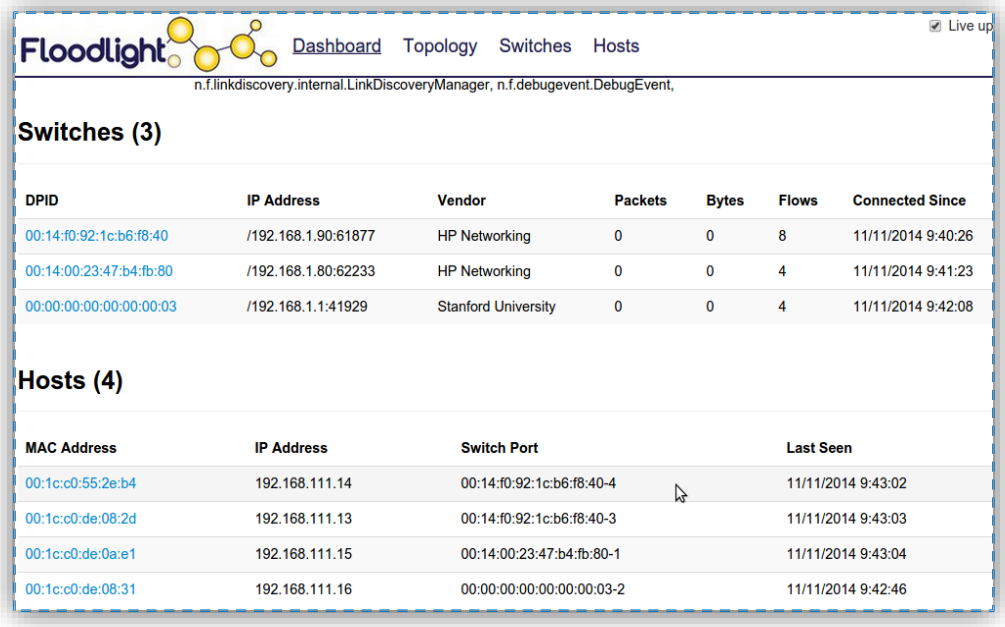


Fig. 4. 51 Visualización de la topología en la interfaz web de Floodlight

Esta herramienta es muy útil ya que nos permite además verificar las entradas de flujo instaladas. Tal como se observa en la figura 4.52.

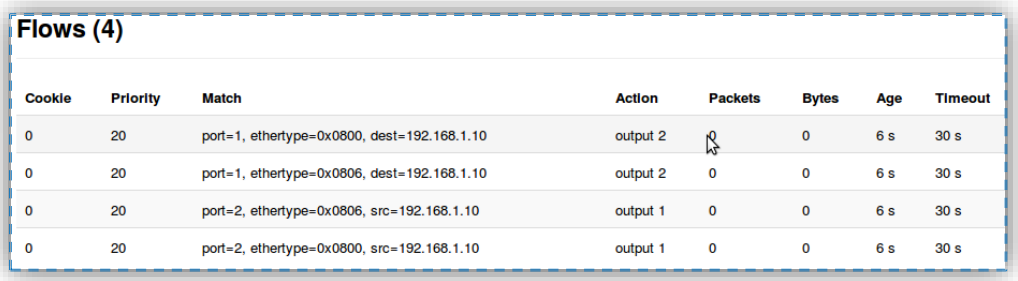


Fig. 4. 52 Visualización del flujo mediante la interfaz web de Floodlight

4.6.2 Pruebas del módulo Filtrado por Mac

Para revisar cómo está funcionando el módulo se realizaron los siguientes pasos

- Ejecutar el modulo junto con la simulación de la misma manera que con el módulo anterior.
- Introducir las direcciones mac de los host que se habilitará la conexión a través

del comando curl.

1. Visualización de direcciones Mac almacenadas

   En la figura 4.53 observamos el comando para visualizar las direcciones mac que se han almacenado.

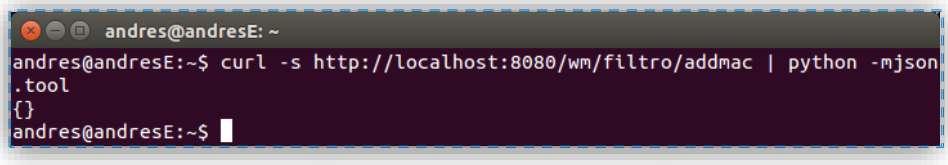


Fig. 4. 53 Visualización de las direcciones MAC

   Como se puede observar al iniciar el modulo no se tienen direcciones MAC. Por lo que es necesario ingresar de forma manual.

2. Ingreso de direcciones MAC

   En la figura 4.54 observamos el comando para introducir las direcciones

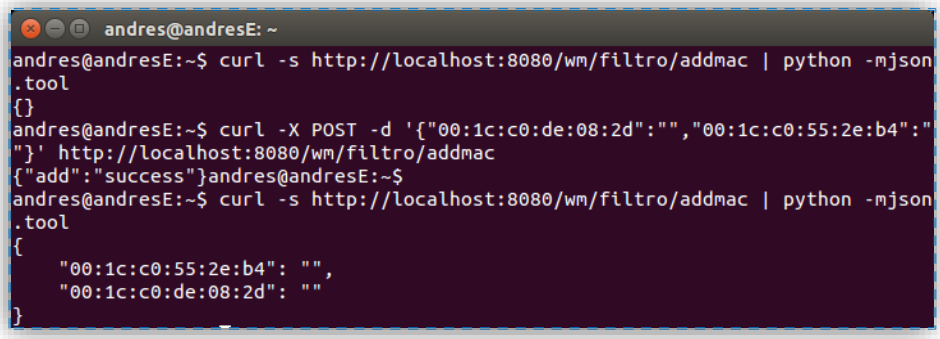


Fig. 4. 54 Introducción de la direcciones MAC

- Pruebas de conexión

Posterior al ingreso de direcciones MAC se puede observar el conjunto de direcciones almacenadas, posteriormente procederemos a realizar una prueba de conexión entre los host que poseen estas direcciones para verificar la conexión como se muestra en la figura 4.55.

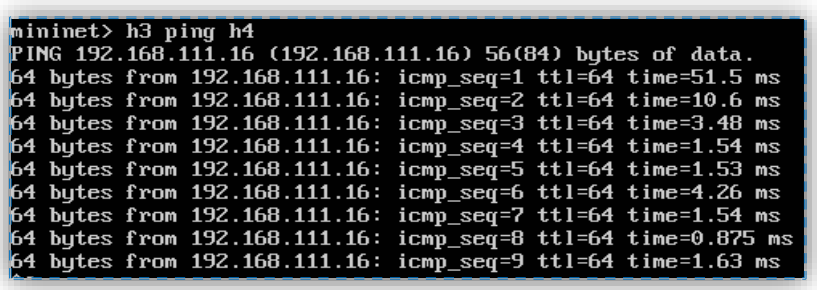


Fig. 4. 55 Pruebas de conexión entre los host 3 y 4

Ahora procedemos a almacenar todas las direcciones mac de los host de la topología con lo que conseguimos que todas las maquinas tengan conexión como se muestra en las figuras 4.56 y 4.57.

Fig. 4. 56 Ingreso de las direcciones MAC de todos los hosts

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Fig. 4. 57 Pruebas de conectividad

4.6.3 Pruebas del módulo balanceo de carga

Para la realización de pruebas de este módulo se realizó una topología que dispone de un switch con cuatro host de los cuales los dos primeros se encuentran conectados en las interfaces 1 y 2 y serán los servidores que atenderán las peticiones mientras los dos últimos serán los clientes. En la figura 4.58 se muestra el código utilizado para la creación de esta topología en mininet.

```
"""
UNIVERSIDAD TECNICA DE AMBATO
          FISEI
ELECTRONICA Y COMUNICACIONES
"""

from mininet.topo import Topo

class MyTopo( Topo ):
    "ALEX NUÑEZ."

    def __init__( self ):

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        Host1 =self.addHost('h1',mac='00:1c:c0:de:08:31',ip='192.168.111.13/24')
        Host2 =self.addHost('h2',mac='00:1c:c0:de:08:e1',ip='192.168.111.14/24')
        Host3 =self.addHost('h3',mac='00:1c:c0:de:08:2d',ip='192.168.111.15/24')
        Host4 =self.addHost('h4',mac='00:1c:c0:55:2e:b4',ip='192.168.111.16/24')
        Switch1 = self.addSwitch( 's1',dpid="0014f0921cb6f840")

        # Add links
        self.addLink( Switch1, Host1 )
        self.addLink( Switch1, Host2 )
        self.addLink( Switch1, Host3 )
        self.addLink( Switch1, Host4 )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Fig. 4. 58 Código para la creación de la topología del Balanceador de carga

Se ha realizado la prueba de conectividad desde uno de los clientes hacia la ip lógica del balanceador se carga y se ha obtenido conexión lo cual demuestra el correcto funcionamiento del módulo como se muestra en la figura 4.59.

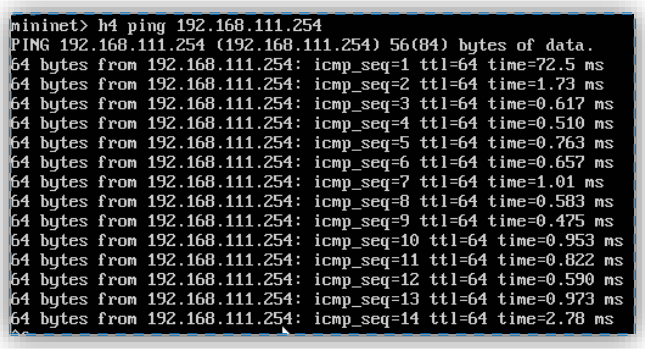


Fig. 4. 59 Pruebas de conexión modulo balanceo de carga

4.7 Implementación del prototipo de la red utilizando conmutadores físicos.

Para la implementación del prototipo físico se ha decido usar dos switches hp 3500-24 y un router wifi TP-LINK TL-WR1043ND versión 2.1 al mismo que es fue necesario cambiarle su firmware con el objetivo de habilitarle para que trabaje con el protocolo OpenFlow, de igual manera a los dos switches HP se los configuró para que puedan trabajar con el protocolo OpenFlow.

A continuación se muestra la topología física implementa en las figuras 4.60 y 4.61.

Fig. 4. 60 Switches utilizados en la infraestructura de red

Fig. 4. 61 Topología física

4.7.1 Realización de pruebas del módulo para inserción de reglas estáticas

En la simulación se explicó el funcionamiento de este módulo en este ítem se procederá a realizar las pruebas con los equipos físicos para corroborar su correcto funcionamiento.

Pruebas de conectividad del host 1

El host uno esta designado con la dirección IP 192.168.111.13/24 de manera que solamente podrá establecer comunicación con el host 3 designado con la dirección IP 192.168.111.15/24.

A continuación se realizó las pruebas de conectividad desde el host 1 hacia los demás host. Como se muestra en la figura 4.62.

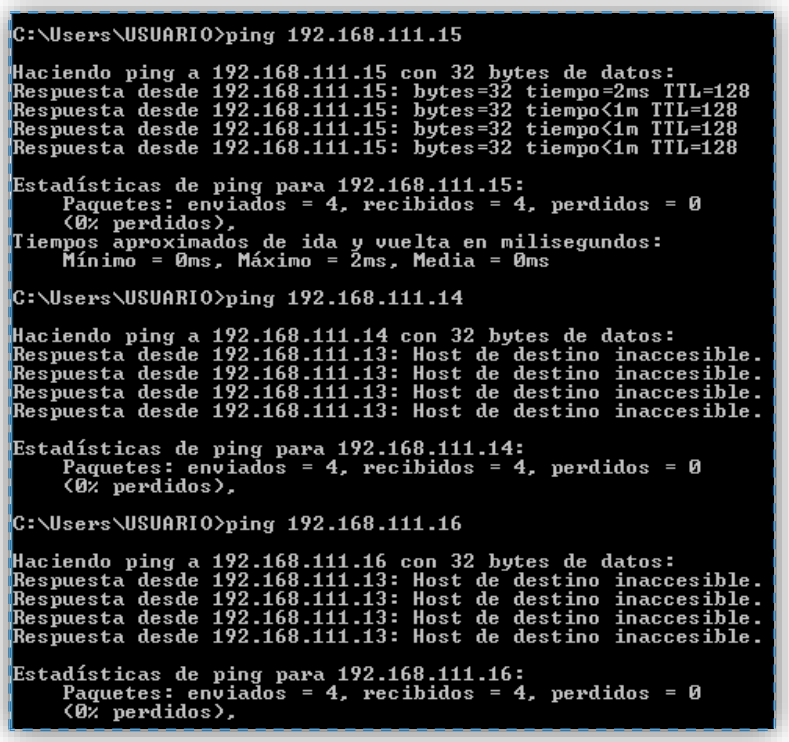


Fig. 4. 62 Pruebas de conectividad del host 1

Como se puede observar en la imagen el host uno solamente puede establecer comunicación con la ip 192.168.111.15 y mientras que a las otras direcciónes el acceso no es posible, con lo cual comprobamos que el módulo está funcionando correctamente.

Pruebas de conectividad del host 2

De la misma manera procedemos a establecer conexión con todas las maquinas desde el host 2 designado con la dirección ip 192.168.111.14/24 y obtenemos como resultado una conexión exitosa hacia el host 4 designado con la dirección IP 192.168.116/24. Como se muestra en la figura 4.63.

Fig. 4. 63 Pruebas de conexión del host 2

De esta manera se puede comprobar el correcto funcionamiento del módulo, el cual provee conectividad solamente entre las maquinas asignadas.

Posteriormente observamos la topología en la interfaz web del controlador así como la información de los switches y host. Como se muestra en las figuras 4.64 y 4.65.

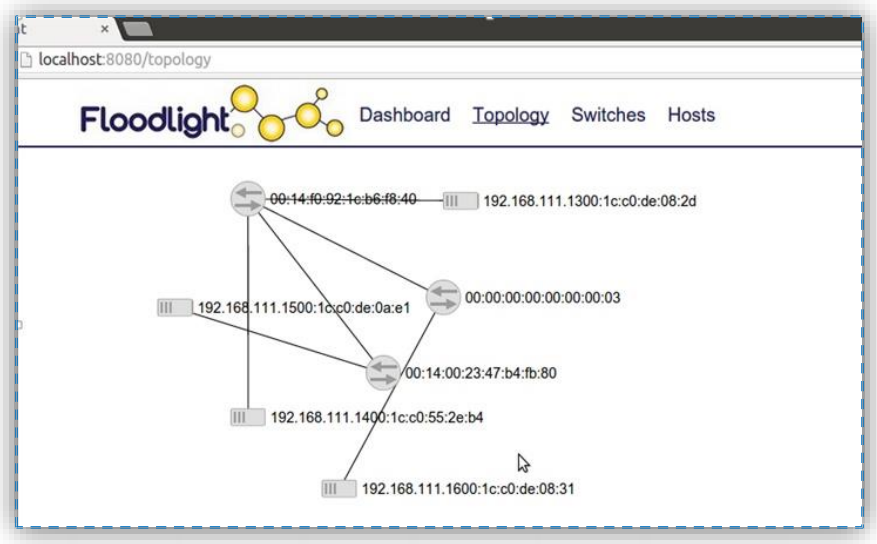


Fig. 4. 64 Visualización de la topología en la interfaz web de Floodlight

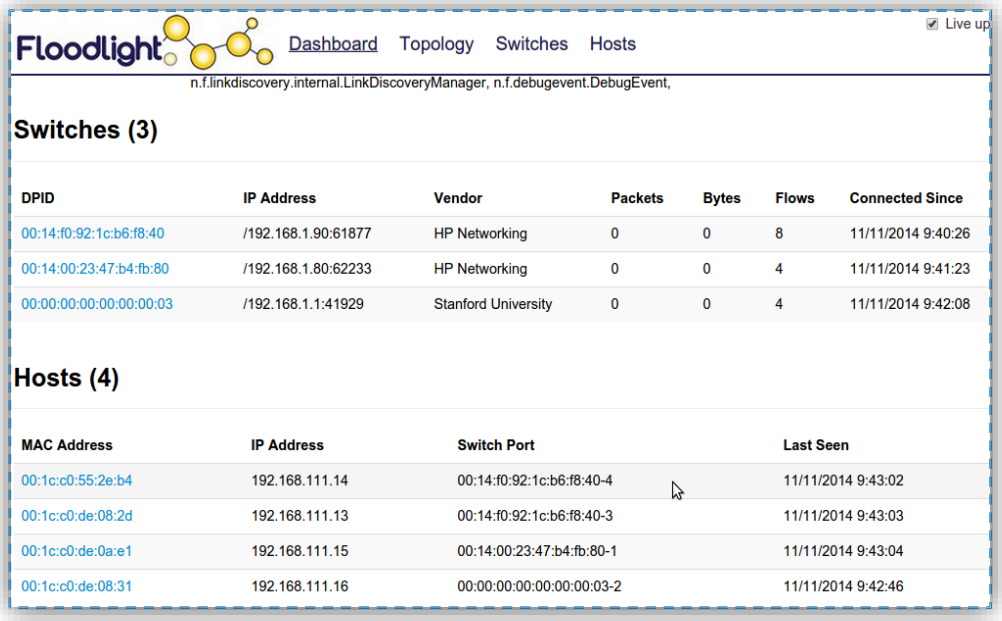


Fig. 4. 65 Visualización de la información de los dispositivos de la topología

Además se verificó las entradas de flujo que se insertó en cada uno de los switches. Como se muestra en las figuras 4.66, 4.67 y 4.68

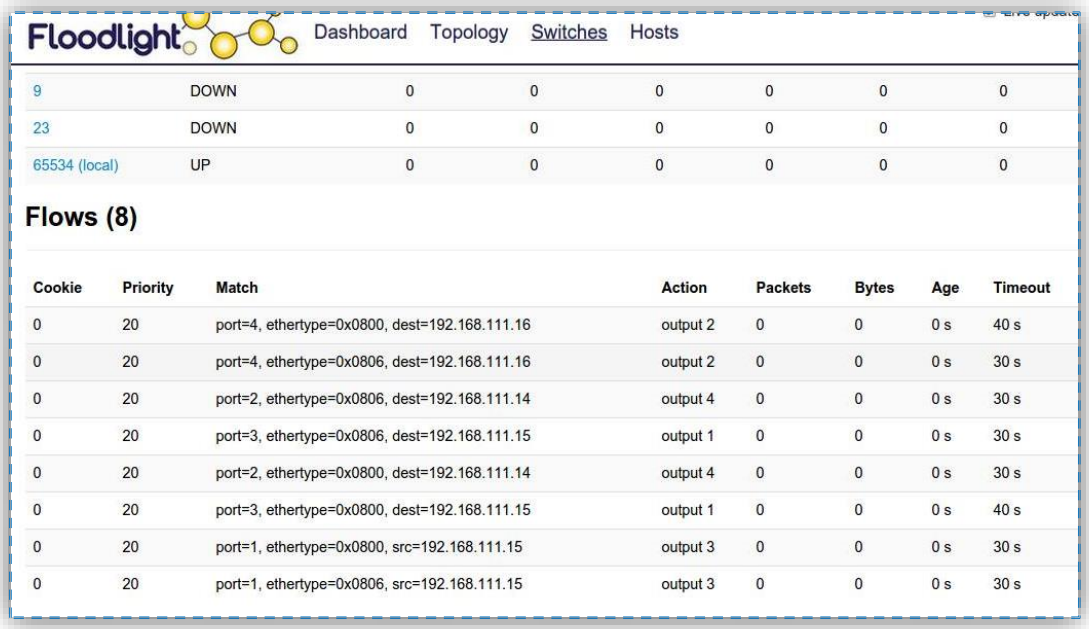


Fig. 4. 66 Entradas de flujo del Switch 1

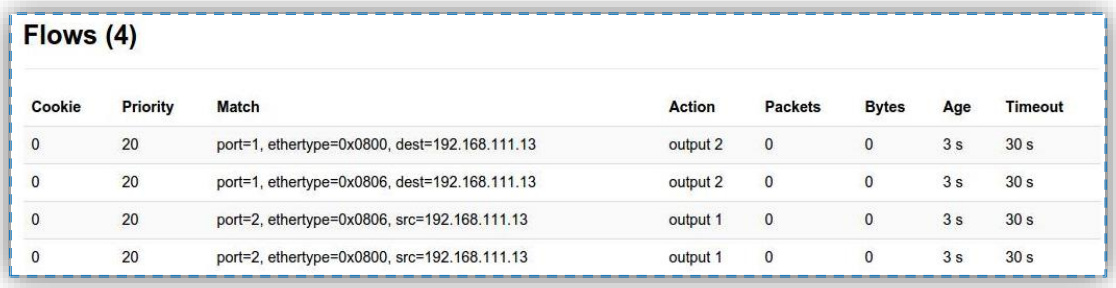


Fig. 4. 67 Entradas de flujo del Switch 2

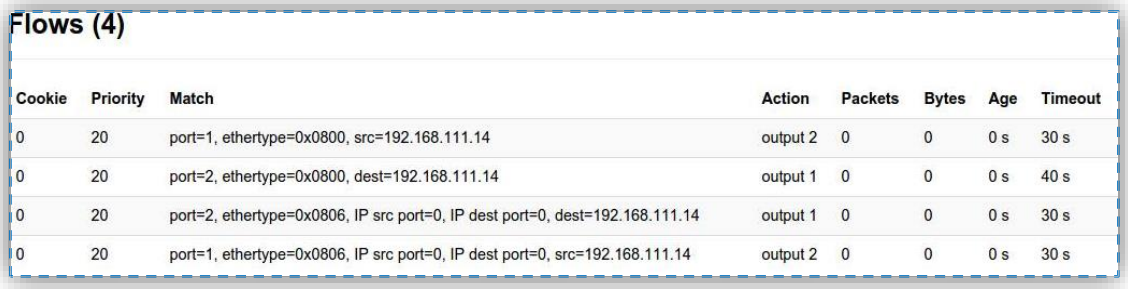


Fig. 4. 68 Entradas de flujo del Switch 3

4.7.2 Pruebas del módulo Filtrado por Mac

Para la simulación de este módulo se realizaron los siguientes pasos:

- Verificación de las direcciones MAC que se han almacenado en el módulo como se muestra en la siguiente figura 4.69.

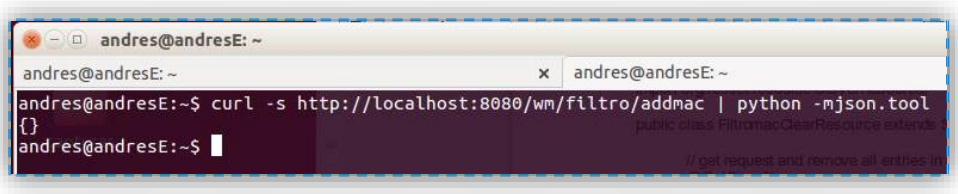


Fig. 4. 69 Visualización de las direcciones MAC almacenadas

Como podemos observar en la figura no se ha almacenado ninguna dirección mac, debido a ello se procedio a almacenar las direcciones mac de dos host como se muestra en la figura 4.70.

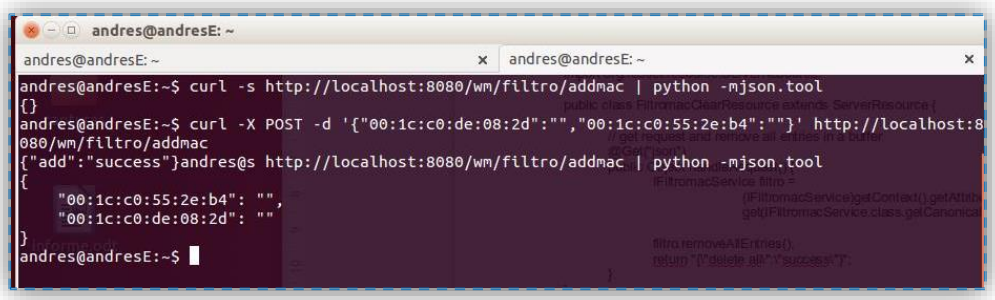


Fig. 4. 70 Almacenamiento de las dirección mac través del comando curl

Ahora que tenemos las direcciones mac de dos host se procedio a realizar las pruebas de conexión, cabe mencionar que las direcciones mac que se han almacenado pertenecen a las IP 192.168.111.13 y 192.168.111.14.

- Pruebas de conexión

Pruebas de conexión del host 1

En la figura 4.71 se muestra las pruebas de conexión del host 1

Fig. 4. 71 Pruebas de conexión del host 1

Como podemos observar la dirección MAC que corresponde al host 192.168.111.16 no se almaceno por lo cual no se puede establecer conexión con este host

Pruebas del host 2

En la figura 4.72 se muestra las pruebas de conexión del host 2

Fig. 4. 72 Pruebas de conexión del host 2

Posteriormente se almacenó la siguiente dirección mac perteneciente al host que posee la dirección ip 192.168.111.16. Como se muestra en la figura 4.73.

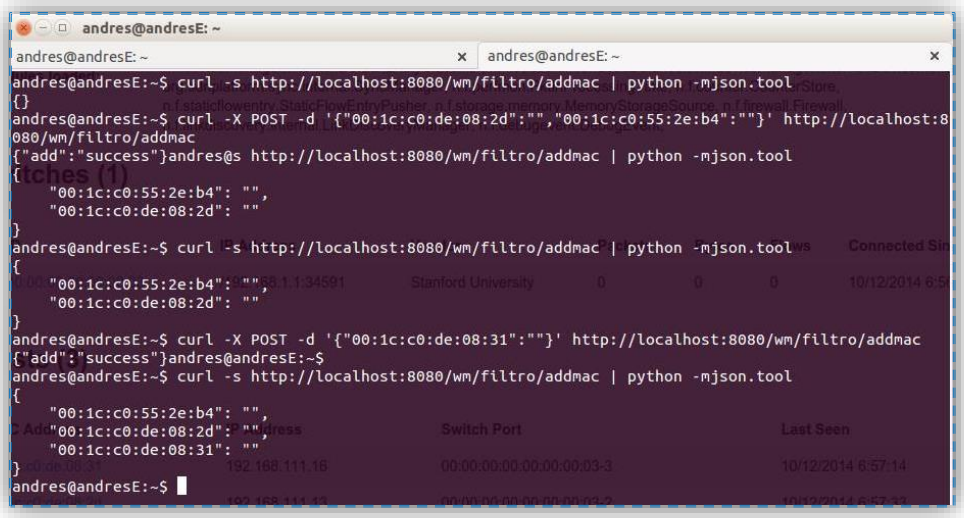


Fig. 4. 73 Almacenamiento de la dirección mac del host 3

A continuación se realizó las pruebas de conexión hacia este host con la finalidad de observar que el módulo funciona de manera correcta dejando pasar el tráfico solamente si asignamos las direcciones mac de los host a los cuales se desee establecer conexión. En las figuras 4.74 y 4.75.

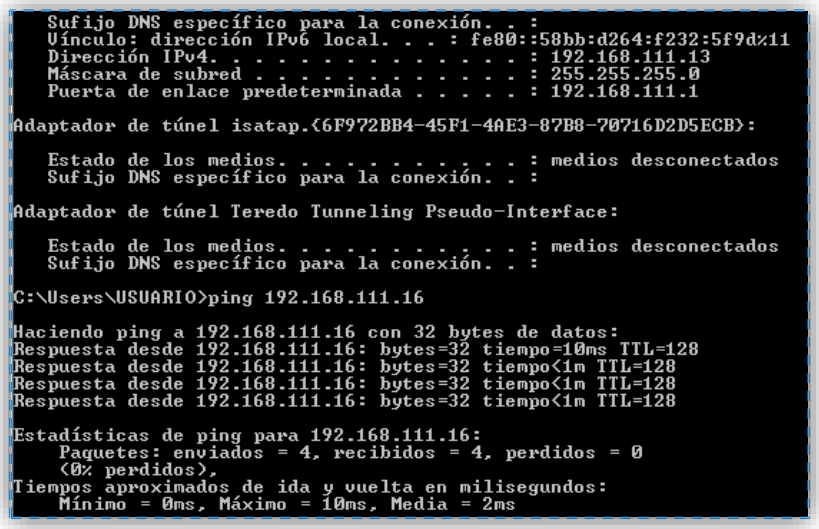
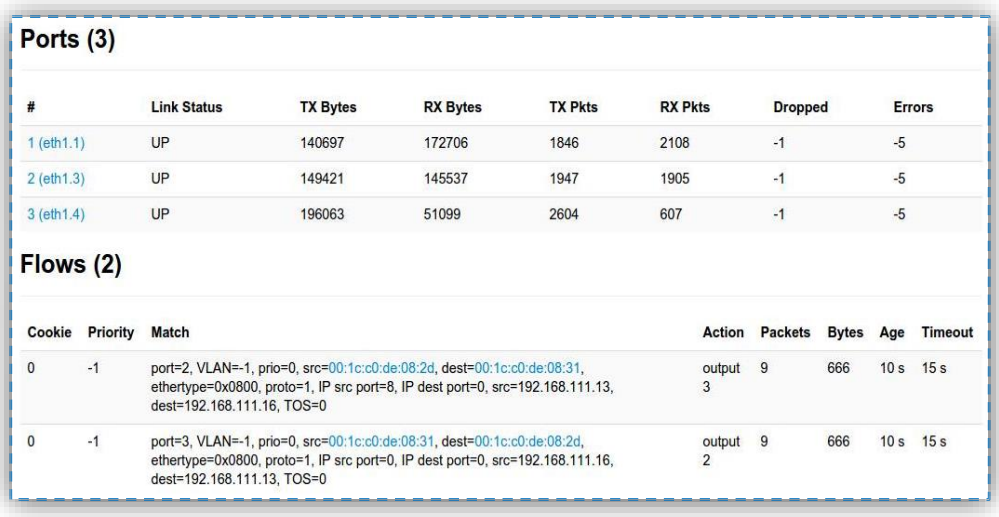


Fig. 4. 74 Pruebas de conexión del host 1

Fig. 4. 75 Pruebas de conexión del host 2

Como último paso procederemos a verificar el flujo que se ha instalado en el switch a través del interfaz web de Floodlight. Tal como se muestra en la figura 4.76.



Ports (3)

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
1 (eth1.1)	UP	140697	172706	1846	2108	-1	-5
2 (eth1.3)	UP	149421	145537	1947	1905	-1	-5
3 (eth1.4)	UP	196063	51099	2604	607	-1	-5

Flows (2)

Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
0	-1	port=2, VLAN=-1, prio=0, src=00:1c:c0:de:08:2d, dest=00:1c:c0:de:08:31, ethertype=0x0800, proto=1, IP src port=8, IP dest port=0, src=192.168.111.13, dest=192.168.111.16, TOS=0	output 3	9	666	10 s	15 s
0	-1	port=3, VLAN=-1, prio=0, src=00:1c:c0:de:08:31, dest=00:1c:c0:de:08:2d, ethertype=0x0800, proto=1, IP src port=0, IP dest port=0, src=192.168.111.16, dest=192.168.111.13, TOS=0	output 2	9	666	10 s	15 s

Fig. 4. 76 Flujo instalado en el switch

4.7.3 Pruebas del balanceador de carga.

Se ha levantado un balanceador de carga el mismo que atiende peticiones ip, este módulo está diseñado para que los clientes realicen peticiones a los servidores que están conectados a las interfaces 1 y 2 del switch. En la figura 4.77 se muestra esta topología.

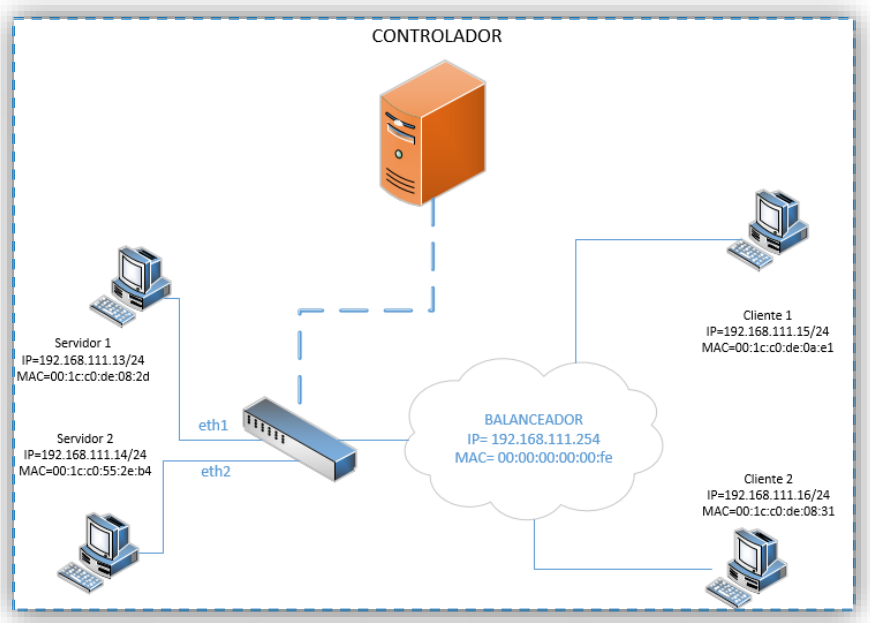


Fig. 4. 77 Topología del balanceador de carga

Este módulo funciona de manera que los clientes realizan peticiones a la dirección del balanceador de carga denominada como 192.168.111.254 cuando las peticiones se realizan el módulo las redirecciona hacia uno de los servidores y cambia de servidor en función del número de peticiones que se tenga.

Como primer paso para realizar las pruebas de conexión se debe almacenar todas las direcciones mac de los host de la topología como se muestra en la figura 4.78.

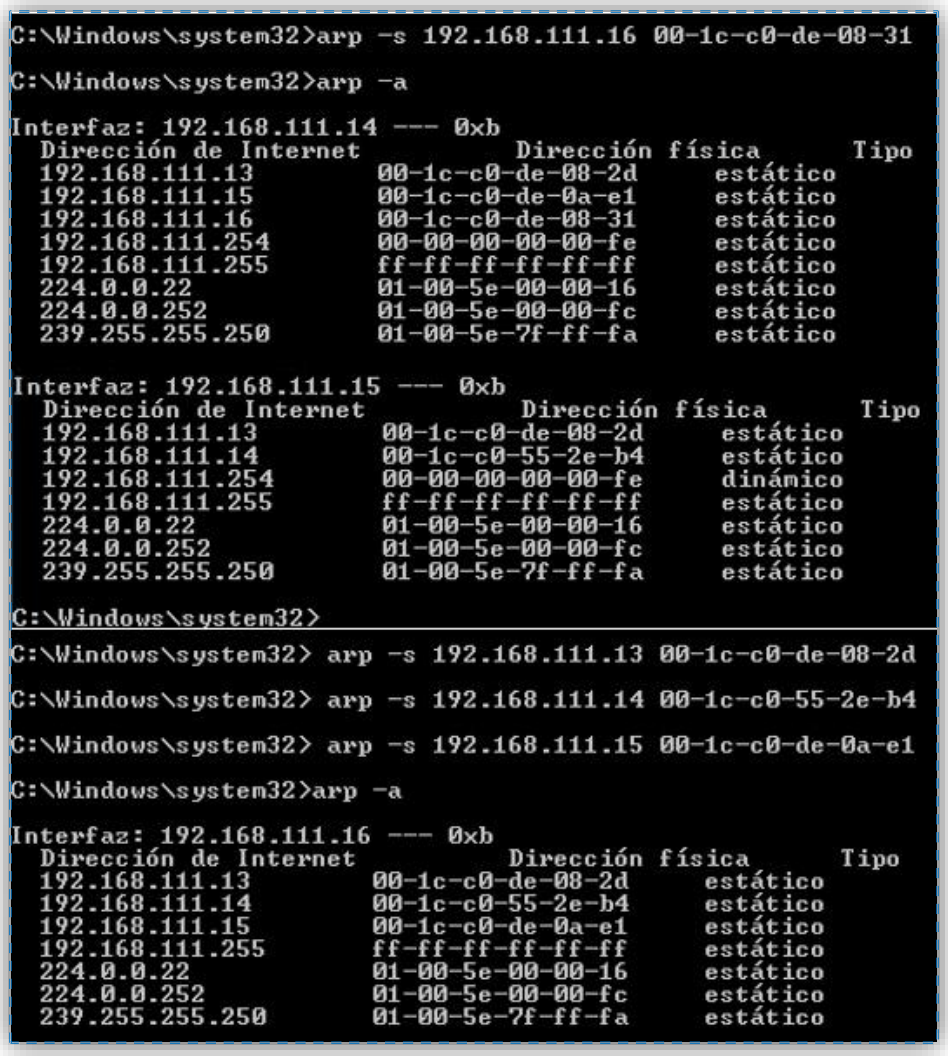


Fig. 4. 78 Adición manual de las tablas ARP

Una vez que se han almacenado las direcciones Mac de los clientes y servidores se procedio a realizar las pruebas de conexión al realizar un ping desde uno de los

clientes hacia la dirección IP del balanceador de carga. Como se muestra en la figura 4.79.

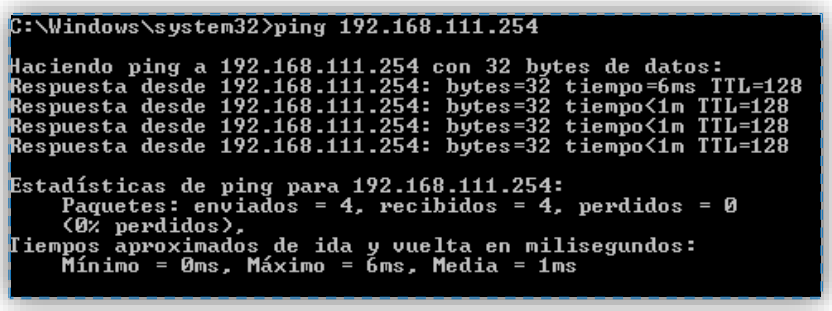


Fig. 4. 79 Pruebas de conexión del balanceador de carga

Como podemos observar las pruebas de conectividad realizadas hacia la ip del balanceador son exitosas.

Por ultimo deberemos ver como fluyen los flujos a través del switch por lo cual verificaremos en la interfaz web del controlador. Como se muestra en la figura 4.80.

Flows (2)

Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
0	0	port=2, VLAN=-1, prio=0, src=00:1c:c0:55:2e:b4, dest=00:1c:c0:de:0a:e1, ethertype=0x0800, src=192.168.111.14, dest=192.168.111.15, TOS=0	src=00:00:00:00:00:fe, src=-1062703106, output 3	10	740	11 s	60 s
0	0	port=3, VLAN=-1, prio=0, src=00:1c:c0:de:0a:e1, dest=00:00:00:00:00:fe, ethertype=0x0800, src=192.168.111.15, dest=192.168.111.254, TOS=0	dest=00:1c:c0:55:2e:b4, dest=-1062703346, output 2	9	666	11 s	60 s

Fig. 4. 80 Flujo instalado en el switch

En la imagen se puede observar que la petición se realizó desde el cliente 1 con la dirección ip 192.168.111.15 hacia la ip del balanceador de carga 192.168.111.254 al ser esta una ip lógica se direccionó dicha petición al servidor 2 con la dirección 192.168.111.14. En la figura 4.81 se puede observar los puertos de entrada y salida de las peticiones realizadas.

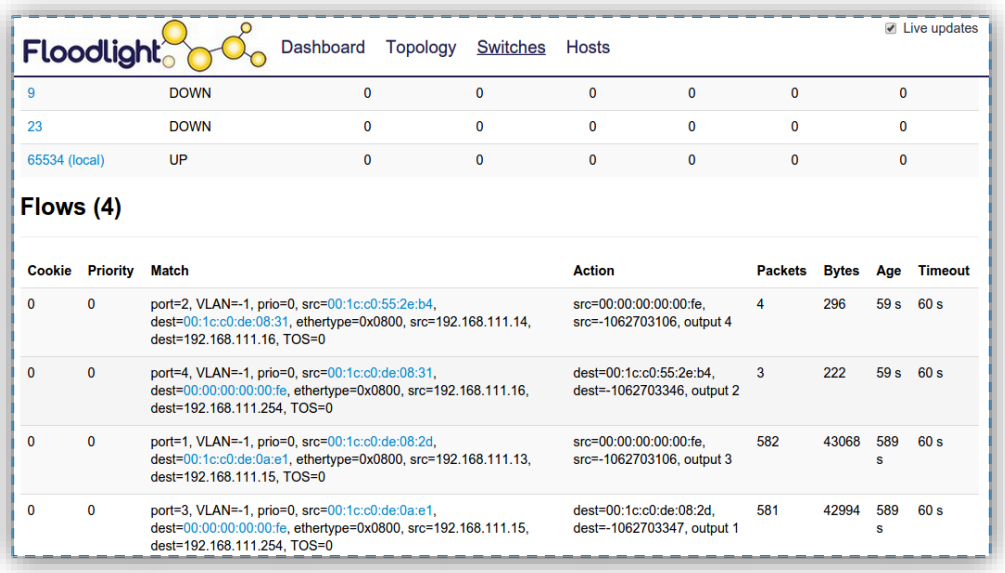


Fig. 4. 81 Identificación de los puertos de entrada y salida

De la misma manera se realizarón peticiones desde los dos clientes y se observó que en función del número de peticiones cambio el servidor que atiende a las mismas, en la imagen se observa que ahora el cliente 1 con la dirección 192.168.111.15 realiza la petición pero ahora el que responde las peticiones es el servidor 1 con la dirección ip 192.168.111.13 en lugar del servidor 2 como sucedió anteriormente, mientras que el servidor 2 atiende las peticiones del cliente 2.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Al finalizar el presente proyecto de titulación se obtuvieron las siguientes conclusiones:

- Las redes definidas por software surgen debido a la incapacidad de las redes convencionales de permitir cambios en los patrones de tráfico de forma dinámica, mediante la adición, eliminación o modificación de reglas de flujo en los dispositivos de interworking que soporte OpenFlow.

- Previo a la implementación del prototipo de red se llevó a cabo la actualización del firmware del router TP-LINK modelo TL-WR1043ND Versión 2.1 concluyendo que esta versión del router si soporta OpenFlow.

- Al utilizar los controladores evaluados en el proyecto, se pudo concluir que Floodlight presenta mejores características que Beacon, debido a su facilidad par crear nuevos módulos, agregar dependencias y utilizar librerías. Además presenta el servicio Rest Api para interactuar con los módulos del controlador en forma remota.

5.2 Recomendaciones

- Al configurar el archivo de red del router habilitado TP-LINK , se concluyó que para habilitar los puertos del router se los debe hacer referenciándose a ellos con el mismo número de la vlan en la que están asignados es decir si el puerto 1 está asignado en la vlan 4 la interfaz que se habilitará para este puerto es la eth 1.4

- Al programar un módulo en Beacon es necesario definir Wildcards en el proceso de comparación, para no incluir todos los campos de la cabecera del paquete.

- Al programar un módulo en Floodlight es necesario definir la longitud que poseen los paquetes al momento de definir las acciones, ya que si no se lo hace el controlador genera un error al momento de escribir la entrada de flujo en el switch.

Bibliografía

[1] C. A. Sánchez, «Proyecto Arpanet,» 20 Agosto 2009. [En línea]. Available: http://www.icesi.edu.co/blogs_estudiantes/emicasanchez/2009/08/09/proyecto-arpanet/.

[2] A. L. Valdivieso y L. I. Barona, «Evolution and Challenges of Software Defined Networking,» 2013. [En línea]. Available: http://sites.ieee.org/sdn4fns/files/2013/11/presentaci%C3%B3n_v3-Evolution-and-Challenges-of-Software-Defined-Networking.pdf.

[3] A. M. Rojas, «Propuesta para la implementacion de un laboratorio de acceso remoto usando redes definidas en software,» Universidad ICESI, Santiago de Cali, Colombia, 2012. [En línea]. Available: http://bibliotecadigital.icesi.edu.co/biblioteca_digital/bitstream/10906/68434/1/propuesta_implementacion_laboratorio.pdf.

[4] I. Gavilan, «Fundamentos de SDN,» 26 Agosto 2013. [En línea]. Available: http://es.slideshare.net/igrgavilan/20130805-introduccion-sdn.

[5] J. C. Chico, «Implementacion de un prototipo de una Red Definida por Software (SDN) empleando una solucion basada en hardware,» Escuela Politecnica Nacional, Quito, Ecuador, 2013. [En línea]. Available: http://www.icesi.edu.co/blogs_estudiantes/emicasanchez/2009/08/09/proyecto-arpanet/.

[6] D. G. Morrillo, «Implementacion de un prototipo de una Red Definida por Software (SDN) empleando una solucion basada en software software,» Mayo 2014. [En línea]. Available: http://bibdigital.epn.edu.ec/bitstream/15000/6681/1/CD-5065.pdf.

[7] S. Rodríguez, «Mecanismos de control de las comunicaciones en la internet del futuro a través de OpenFlow,» Universidad de Cantabria, Cantabria, España, 2012. [En línea]. Available: http://repositorio.unican.es/xmlui/bitstream/handle/10902/1165/Sergio%20Rodriguez%20Santamaria.pdf?sequence=1.

[8] A. L. Valdivieso, A. B. Peral, L. I. Barona y L. J. García, «International Journal of Distributed Sensor Networks,» SDN: Evolution and Opportunities in the Development IoT Applications, 4 Mayo 2014. [En línea]. Available: http://www.hindawi.com/journals/ijdsn/2014/735142/.

[9] N. Figuerola, «SDN - Redes definidas por Software,» Octubre 2013. [En línea]. Available: http://articulosit.files.wordpress.com/2013/10/sdn.pdf.

[10] OPEN NETWORKING FOUNDATION, «Software-Defined Networking: The New Norm for Networks,» 13 Abril 2012. [En línea]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[11] OPEN NETWORKING FOUNDATION, «Software-Defined Networking (SDN) Definition,» 1 Mayo 2013. [En línea]. Available: www.opennetworking.org/sdn-resources/sdn-definition.

[12] S. Azodolmolky, Software Defined Networking with OpenFlow, Packt Publishing Ltd, 2013.

[13] PHILOSOPHIAE NATURALIS PRINCIPIA TECHNOLOGICA, «Investigación de Redes Definidas por Software,» Marzo 2014. [En línea]. Available: http://principiatechnologica.files.wordpress.com/2014/03/sdn-principia-technologica-3.pdf.

[14] NESSYS, «Introducción a las Redes Definidas por Software SDN,» 12 Octubre 1012. [En línea]. Available: http://www.nessys.es/introduccion-a-las-redes-definidas-por-software-%C2%B7-sdn/.

[15] D. Savu y S. Stancu, «http://openlab.web.cern.ch/publications/presentations/software-defined-networking-technology-details-and-openlab-research,» 14 Febrero 2014. [En línea]. Available: http://openlab.web.cern.ch/sites/openlab.web.cern.ch/files/presentations/0%5B1%5D.pdf.

[16] S. Seetharaman, «OpenFlow/ Software Defined Networking,» Noviembre 2011. [En línea]. Available: http://es.slideshare.net/openflow/openflow-tutorial.

[17] A. Astudillo, G. Dreier, L.Bertholdo y L. Tarouco, «Introducción SDN y OpenFlow,» 14 Noviembre 2012. [En línea]. Available: http://eventos.redclara.net/indico/getFile.py/access?contribId=0&resId=0&materialId=slides&confId=197.

[18] B. Salisbury, «TCAMs and OpenFlow – What Every SDN Practitioner Must Know,» 1 Julio 2012. [En línea]. Available: https://www.sdncentral.com/technology/sdn-openflow-tcam-need-to-know/2012/07/.

[19] OPEN NETWORKING FOUNDATION , «Open Flow Switch Specification,» 31 Diciembre 2009. [En línea]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf.

[20] D. Mejía e I. Bernal, «Prototipo de redes definidas por software,» Noviembre 2013. [En línea]. Available: http://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB0QFjAA&url=http%3A%2F%2Fticec.cedia.org.ec%2Fdmdocuments%2FDocumentacion%2FDia2%2Fponencia8.pptx&ei=qC76U8jIH4zlsAS5h4CYAQ&usg=AFQjCNHa4Vq5ew9bDAxWIpZAEQ0tWmp4sQ.

[21] A. Astudillo, G. Dreier, L. Bertholo y L. Tarouco, «Controladores para OpenFlow,» 16 Noviembre 2012. [En línea]. Available: http://eventos.redclara.net/indico/getFile.py/access?contribId=0&resId=4&materialId=slides&confId=197.

[22] T. Nadeau y K. Gray, «SDN Software Defines Networks,» de *An Authoritative Review of Network Programmability Technologies*, O'Reilly Media, Inc, 2013.

[23] Vargas y A. Velásquez, «Emulación de una red definida por software utilizando MiniNet,» Escuela Técnica Superior de Ingenieros en Telecomunicaciones (ETSIT - UPM), [En línea]. Available: https://www.academia.edu/5730624/Emulacion_de_una_red_definida_por_software_utilizando_MiniNet.

[24] Mininet Team, «Mininet Overview,» 2014. [En línea]. Available: http://mininet.org/overview/.

[25] GitHub, «Introduction to Mininet,» [En línea]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[26] ClaraTeach, «TUTORIAL MININET,» [En línea]. Available: http://eventos.redclara.net/indico/getFile.py/access?contribId=1&resId=4&materialId=slides&confId=197.

[27] Mininet, «Mininet Walkthrough,» 2014. [En línea]. Available: http://mininet.org/walkthrough/.

[28] O. R. Hervás, «Software Defined Networking,» Universidad Politecnica de Cataluña, [En línea]. Available: http://upcommons.upc.edu/pfc/bitstream/2099.1/21633/4/Memoria.pdf.

[29] HP, «HP Networking OpenFlow Workshop,» 102. [En línea]. Available: http://eventos.redclara.net/indico/event/197/material/slides/3?contribId=1.

[30] OpenFlow, «DPCTL,» Mayo 2008. [En línea]. Available: http://ranosgrant.cocolog-nifty.com/openflow/dpctl.8.html#lbAH.

[31] Boletín Tecnológico AREJ , «SDN (software-defined networking) la red definida por software,» 24 Abril 2013. [En línea]. Available: https://archive.org/details/redes_definido_por_software.

[32] J. Bastida, «Redes Virtuales Distribuidas Basadas en OpenFlow,,» 13 Septiembre 2013. [En línea]. Available: http://es.scribd.com/doc/211359097/TFM-josebastida-1-2.

[33] PANTOU, «OpenFlow 1.0 for OpenWRT,» 29 Octubre 2012. [En línea]. Available: http://archive.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT.

[34] D. Erickson, «Beacon Quick STart,» 13 Septiembre 2013. [En línea]. Available: https://openflow.stanford.edu/display/Beacon/Quick+Start.

[35] K. Parraga y . J. Ching, «Floodlight Instalation guide,» 13 Diciembre 2013. [En línea]. Available: http://docs.projectfloodlight.org/display/floodlightcontroller/Installation+Guide.

[36] B. Salisbury, «Tutorial to build a Floodlight SDN OpenFlow Controller Module,» 11 Noviembre 2012. [En línea]. Available: http://networkstatic.net/tutorial-to-build-a-floodlight-sdn-openflow-controller-module/.

[37] D. Rubert, «Instalar OpenWRT,» 4 Abril 2014. [En línea]. Available: http://tombatossals.github.io/instalar-openwrt/.

[38] V. Shahane, «Vishal Shahane's blog,» 26 Abril 2014. [En línea]. Available: http://vishalshahane.blogspot.com/2014/04/compiling-custom-firmware-for-wireless.html.

[39] OpenWRT, «TP-Link TL-WR1043ND,» [En línea]. Available: http://wiki.openwrt.org/toh/tp-link/tl-wr1043nd.

[40] V. Shahane, «Configuring TP-LINK WR1043ND v2.1 as OpenFlow switch on OpenWRT,» 26 Abril 2014. [En línea]. Available: http://vishalshahane.blogspot.com/2014/04/configuring-tp-link-wr1043nd-v21-as.html.

[41] D. Erickson, «Your First Bundle,» 28 Agosto 2012. [En línea]. Available: https://openflow.stanford.edu/display/Beacon/Your+First+Bundle.

[42] D. Erickson, «Create Learning Switch,» 14 Abril 2013. [En línea]. Available: https://github.com/onstutorial/onstutorial/wiki/Create-Learning-Switch.

[43] GItHub, «Openflow tutorial,» 12 Septiembre 2012. [En línea]. Available: https://github.com/mininet/openflow-tutorial/wiki/Create-a-Learning-Switch.

[44] McCauley, A. Al-Shabibi y Murphy, «POX Wiki,» 24 Enero 2012. [En línea]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki.

[45] Stanford, «OPENFLOW,» 7 Diciembre 2010. [En línea]. Available: https://openflow.stanford.edu/forums/.

[46] ClaraTeach, «Curso de Openflow,» 13 Noviembre 2012. [En línea]. Available: http://eventos.redclara.net/indico/event/197/contribution/1/material/0/0.pdf.

[47] D. Kim, «Create a floodlight module,» 28 Enero 2014. [En línea]. Available: http://dannykim.me/danny/openflow/57682.

[48] D. Kim, «Create a floodlight module with REST API (get, add, remove data),» 29 Enero 2014. [En línea]. Available: http://dannykim.me/danny/openflow/62763.

[49] K. Parraga y C. Wang , «How to Add Services to a Module,» 24 Mayo 2013. [En línea]. Available: http://www.openflowhub.org/display/floodlightcontroller/How+to+Add+Services+to+a+Module.

[50] A. Krishnamurthy, «PMP Network Systems,» 29 Octubre 2013. [En línea]. Available: http://courses.cs.washington.edu/courses/csep561/13au/projects/proj2.html.

[51] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown y R. Johari, «Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow,» 2 Diciembre 2009. [En línea]. Available: http://conferences.sigcomm.org/sigcomm/2009/demos/sigcomm-pd-2009-final26.pdf.

[52] S. Govindraj, A. Jayaraman, N. Khanna y K. Ravi Prakash, «OpenFlow: Load Balancing in enterprise networks using,» 4 Mayo 2013. [En línea]. Available: http://morse.colorado.edu/~tlen5710/12s/OpenFlow.pdf.

[53] C. Heng Ke, «IP Load Balancer,» 2013. [En línea]. Available: http://csie.nqu.edu.tw/smallko/sdn/mySDN_Lab8.pdf.

[54] H. Uppal y D. Brandon, «OpenFlow Based Load Balancing,» 2009. [En línea]. Available: http://courses.cs.washington.edu/courses/cse561/10sp/project_files/cse561_openflow_project_report.pdf.

[55] B. Plattner, B. Ager, X. Dimitropoulos, S. Neuhaus, M. Happe, K. Hummel y V. Kotronis, «Build a simple load balancer,» 16 Octubre 2013. [En línea]. Available: http://www.csg.ethz.ch/education/lectures/ATCN/hs2013/exercises/exer1_assignment.

[56] OPENWRT, «Table of Hardware,» [En línea]. Available: http://wiki.openwrt.org/toh/tp-link/tl-wr1043nd.

ANEXOS

ANEXO 1: Módulo para inserción de reglas de flujo en Beacon

```
package net.beaconcontroller.modulo;

import java.io.IOException;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.ConcurrentSkipListSet;

import net.beaconcontroller.packet.Ethernet;
import net.beaconcontroller.packet.IPv4;
import net.beaconcontroller.core.IBeaconProvider;
import net.beaconcontroller.core.IOFMessageListener;
import net.beaconcontroller.core.IOFSwitchListener;
import net.beaconcontroller.core.IOFSwitch;

import org.openflow.protocol.OFFlowMod;
import org.openflow.protocol.OFMatch;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFPacketIn;
import org.openflow.protocol.OFPacketOut;
import org.openflow.protocol.OFPort;
import org.openflow.protocol.action.OFAction;
import org.openflow.protocol.action.OFActionOutput;
import org.openflow.protocol.OFType;
import org.openflow.util.HexString;
import org.openflow.util.U16;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Modulo implements IOFMessageListener,
IOFSwitchListener {

protected static Logger logger =
LoggerFactory.getLogger(Modulo.class);

protected IBeaconProvider beaconProvider;


public IBeaconProvider getBeaconProvider() {
        return beaconProvider;
    }

public void setBeaconProvider(IBeaconProvider beaconProvider) {
        this.beaconProvider = beaconProvider;
    }

public void startUp() {
        beaconProvider.addOFMessageListener(OFType.PACKET_IN,
this);
    }
```

```
public void shutDown() {
        beaconProvider.removeOFMessageListener(OFType.PACKET_IN,
this);
    }

    public String getName() {
        return "Modulo";
    }


public Command receive(IOFSwitch sw, OFMessage msg) throws
IOException {

        OFPacketIn pi = (OFPacketIn) msg;
        Reglas(sw,pi);

        return Command.CONTINUE;
    }

public void Reglas(IOFSwitch sw, OFPacketIn pi) throws
IOException{

    //INGRESE DATOS
    /**SWITCH1*/
    //DPID
    long id1=5894009871857728L;
    //long id1=1;
    //IP HOST1
    String ip1="192.168.111.13";
    //IP HOST2
    String ip2="192.168.111.14";

    /**SWITCH2*/
    //DPID
    long id2=5629651061111680L;
    //long id2=2;
    //IP HOST3
    String ip3="192.168.111.15";


    /**SWITCH3*/
    //DPID
    long id3=3;
    //IP HOST4
    String ip4="192.168.111.16";


if(sw.getId()==id1){

//////////////ARP//////////////////
OFFlowMod fm=new OFFlowMod();
fm.setCommand(OFFlowMod.OFPFC_ADD);
fm.setBufferId(-1);
      fm.setIdleTimeout((short)30);
fm.setHardTimeout((short)30);
fm.setPriority((short)20);

OFMatch match =OFMatch.load(pi.getPacketData(),
pi.getInPort());
```

```
        match.setDataLayerType(Ethernet.TYPE_ARP);
        match.setInputPort((short)3);
        match.setNetworkDestination(IPv4.toIPv4Address(ip3));
        match.setWildcards(OFMatch.OFPFW_DL_DST|
                           OFMatch.OFPFW_DL_SRC|
                           OFMatch.OFPFW_DL_VLAN|
                           OFMatch.OFPFW_DL_VLAN_PCP|
                           OFMatch.OFPFW_NW_PROTO|
                           OFMatch.OFPFW_NW_SRC_ALL|
                           OFMatch.OFPFW_NW_TOS|
                           OFMatch.OFPFW_TP_DST|
                           OFMatch.OFPFW_TP_SRC);
        fm.setMatch(match);

        OFAction action=new OFActionOutput((short)1);

        fm.setActions(Collections.singletonList((OFAction)action));
        sw.getOutputStream().write(fm);

    //ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

        OFFlowMod fm1=new OFFlowMod();
        fm1.setCommand(OFFlowMod.OFPFC_ADD);
        fm1.setBufferId(-1);
        fm1.setIdleTimeout((short)30);
        fm1.setHardTimeout((short)30);
        fm1.setPriority((short)20);

        OFMatch match1 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match1.setDataLayerType(Ethernet.TYPE_ARP);
        match1.setInputPort((short)1);
        match1.setNetworkSource(IPv4.toIPv4Address(ip3));
        match1.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_DST_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm1.setMatch(match1);
        OFAction action1=new OFActionOutput((short)3);

        fm1.setActions(Collections.singletonList((OFAction)action1)
        );
        sw.getOutputStream().write(fm1);

    //////////////IP//////////////////

        OFFlowMod fm2=new OFFlowMod();
        fm2.setCommand(OFFlowMod.OFPFC_ADD);
        fm2.setBufferId(-1);
        fm2.setIdleTimeout((short)40);
        fm2.setHardTimeout((short)40);
        fm2.setPriority((short)20);
```

```
        OFMatch match2 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());
        match2.setDataLayerType(Ethernet.TYPE_IPv4);
        match2.setInputPort((short)3);

        match2.setNetworkDestination(IPv4.toIPv4Address(ip3));
        match2.setWildcards(OFMatch.OFPFW_DL_DST|
           OFMatch.OFPFW_DL_SRC|
           OFMatch.OFPFW_DL_VLAN|
           OFMatch.OFPFW_DL_VLAN_PCP|
           OFMatch.OFPFW_NW_PROTO|
           OFMatch.OFPFW_NW_SRC_ALL|
           OFMatch.OFPFW_NW_TOS|
           OFMatch.OFPFW_TP_DST|
           OFMatch.OFPFW_TP_SRC);
        fm2.setMatch(match2);

        OFAction action2=new OFActionOutput((short)1);

        fm2.setActions(Collections.singletonList((OFAction)action2)
        );
        sw.getOutputStream().write(fm2);

    //ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

        OFFlowMod fm3=new OFFlowMod();
        fm3.setCommand(OFFlowMod.OFPFC_ADD);
        fm3.setBufferId(-1);
        fm3.setIdleTimeout((short)30);
        fm3.setHardTimeout((short)30);
        fm3.setPriority((short)20);

        OFMatch match3 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match3.setDataLayerType(Ethernet.TYPE_IPv4);
        match3.setInputPort((short)1);
        match3.setNetworkSource(IPv4.toIPv4Address(ip3));

        match3.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_DST_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm3.setMatch(match3);

        OFAction action3=new OFActionOutput((short)3);

        fm3.setActions(Collections.singletonList((OFAction)action3)
        );
        sw.getOutputStream().write(fm3);
```

```
        //-------------REGLAS PARA EL HOST 4-----------------

              ////////////////ARP/////////////////

        OFFlowMod fm4=new OFFlowMod();
        fm4.setCommand(OFFlowMod.OFPFC_ADD);
        fm4.setBufferId(-1);
        fm4.setIdleTimeout((short)30);
        fm4.setHardTimeout((short)30);
        fm4.setPriority((short)20);

        OFMatch match4 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match4.setDataLayerType(Ethernet.TYPE_ARP);
        match4.setInputPort((short)4);

        match4.setNetworkDestination(IPv4.toIPv4Address(ip4));
        match4.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm4.setMatch(match4);

        OFAction action4=new OFActionOutput((short)2);

        fm4.setActions(Collections.singletonList((OFAction)action4)
        );
        sw.getOutputStream().write(fm4);

    //ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

        OFFlowMod fm5=new OFFlowMod();
        fm5.setCommand(OFFlowMod.OFPFC_ADD);
        fm5.setBufferId(-1);
        fm5.setIdleTimeout((short)30);
        fm5.setHardTimeout((short)30);
        fm5.setPriority((short)20);

        OFMatch match5 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());


        match5.setDataLayerType(Ethernet.TYPE_ARP);
        match5.setInputPort((short)2);
        match5.setNetworkDestination(IPv4.toIPv4Address(ip2));
        match5.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
```

```
                        OFMatch.OFPFW_TP_SRC);
        fm5.setMatch(match5);

        OFAction action5=new OFActionOutput((short)4);

        fm5.setActions(Collections.singletonList((OFAction)action5)
        );
        sw.getOutputStream().write(fm5);

              //////////////IP//////////////////

        OFFlowMod fm6=new OFFlowMod();
        fm6.setCommand(OFFlowMod.OFPFC_ADD);
        fm6.setBufferId(-1);
        fm6.setIdleTimeout((short)40);
        fm6.setHardTimeout((short)40);
        fm6.setPriority((short)20);

        OFMatch match6 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match6.setDataLayerType(Ethernet.TYPE_IPv4);
        match6.setInputPort((short)4);
        match6.setNetworkDestination(IPv4.toIPv4Address(ip4));
        match6.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm6.setMatch(match6);

        OFAction action6=new OFActionOutput((short)2);

        fm6.setActions(Collections.singletonList((OFAction)action6)
        );
        sw.getOutputStream().write(fm6);

//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

        OFFlowMod fm7=new OFFlowMod();
        fm7.setCommand(OFFlowMod.OFPFC_ADD);
        fm7.setBufferId(-1);
        fm7.setIdleTimeout((short)30);
        fm7.setHardTimeout((short)30);
        fm7.setPriority((short)20);

        OFMatch match7 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match7.setDataLayerType(Ethernet.TYPE_IPv4);
        match7.setInputPort((short)2);
        match7.setNetworkDestination(IPv4.toIPv4Address(ip2));
        match7.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
```

```
                                OFMatch.OFPFW_DL_VLAN_PCP|
                                OFMatch.OFPFW_NW_PROTO|
                                OFMatch.OFPFW_NW_SRC_ALL|
                                OFMatch.OFPFW_NW_TOS|
                                OFMatch.OFPFW_TP_DST|
                                OFMatch.OFPFW_TP_SRC);
            fm7.setMatch(match7);

            OFAction action7=new OFActionOutput((short)4);

            fm7.setActions(Collections.singletonList((OFAction)action7)
            );
            sw.getOutputStream().write(fm7);
            }

      if(sw.getId()==id2)
      {

            OFFlowMod fm=new OFFlowMod();
            fm.setCommand(OFFlowMod.OFPFC_ADD);
            fm.setBufferId(-1);
            fm.setIdleTimeout((short)30);
            fm.setHardTimeout((short)30);
            fm.setPriority((short)20);

            OFMatch match =OFMatch.load(pi.getPacketData(),
            pi.getInPort());

            match.setDataLayerType(Ethernet.TYPE_ARP);
            match.setInputPort((short)2);
            match.setNetworkSource(IPv4.toIPv4Address(ip1));
            match.setWildcards(OFMatch.OFPFW_DL_DST|
                               OFMatch.OFPFW_DL_SRC|
                               OFMatch.OFPFW_DL_VLAN|
                               OFMatch.OFPFW_DL_VLAN_PCP|
                               OFMatch.OFPFW_NW_PROTO|
                               OFMatch.OFPFW_NW_DST_ALL|
                               OFMatch.OFPFW_NW_TOS|
                               OFMatch.OFPFW_TP_DST|
                               OFMatch.OFPFW_TP_SRC);
            fm.setMatch(match);

            OFAction action=new OFActionOutput((short)1);

            fm.setActions(Collections.singletonList((OFAction)action));
            sw.getOutputStream().write(fm);

      //ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

            OFFlowMod fm1=new OFFlowMod();
            fm1.setCommand(OFFlowMod.OFPFC_ADD);
            fm1.setBufferId(-1);
            fm1.setIdleTimeout((short)30);
            fm1.setHardTimeout((short)30);
            fm1.setPriority((short)20);

            OFMatch match1 =OFMatch.load(pi.getPacketData(),
            pi.getInPort());
```

```
        match1.setDataLayerType(Ethernet.TYPE_ARP);
        match1.setInputPort((short)1);
        match1.setNetworkDestination(IPv4.toIPv4Address(ip1));
        match1.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
       fm1.setMatch(match1);

       OFAction action1=new OFActionOutput((short)2);

        fm1.setActions(Collections.singletonList((OFAction)action1)
        );
        sw.getOutputStream().write(fm1);

                  //////////////IP//////////////////

        OFFlowMod fm2=new OFFlowMod();
        fm2.setCommand(OFFlowMod.OFPFC_ADD);
        fm2.setBufferId(-1);
        fm2.setIdleTimeout((short)30);
        fm2.setHardTimeout((short)30);
        fm2.setPriority((short)20);

        OFMatch match2 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match2.setDataLayerType(Ethernet.TYPE_IPv4);
        match2.setInputPort((short)2);

        match2.setNetworkSource(IPv4.toIPv4Address(ip1));
        match2.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_DST_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm2.setMatch(match2);

        OFAction action2=new OFActionOutput((short)1);

        fm2.setActions(Collections.singletonList((OFAction)action2)
        );
        sw.getOutputStream().write(fm2);

//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

        OFFlowMod fm3=new OFFlowMod();
        fm3.setCommand(OFFlowMod.OFPFC_ADD);
        fm3.setBufferId(-1);
        fm3.setIdleTimeout((short)30);
```

```
        fm3.setHardTimeout((short)30);
        fm3.setPriority((short)20);

        OFMatch match3 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match3.setDataLayerType(Ethernet.TYPE_IPv4);
        match3.setInputPort((short)1);
        match3.setNetworkDestination(IPv4.toIPv4Address(ip1));
        match3.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm3.setMatch(match3);

        OFAction action3=new OFActionOutput((short)2);

        fm3.setActions(Collections.singletonList((OFAction)action3)
        );
        sw.getOutputStream().write(fm3);
}

if(sw.getId()==id3)
   {

     //////////////ARP//////////////////
      OFFlowMod fm=new OFFlowMod();
      fm.setCommand(OFFlowMod.OFPFC_ADD);
      fm.setBufferId(-1);
      fm.setIdleTimeout((short)30);
      fm.setHardTimeout((short)30);
      fm.setPriority((short)20);

      OFMatch match =OFMatch.load(pi.getPacketData(),
      pi.getInPort());

      match.setDataLayerType(Ethernet.TYPE_ARP);
      match.setInputPort((short)1);
      match.setNetworkSource(IPv4.toIPv4Address(ip2));
      match.setWildcards(OFMatch.OFPFW_DL_DST|
                         OFMatch.OFPFW_DL_SRC|
                         OFMatch.OFPFW_DL_VLAN|
                         OFMatch.OFPFW_DL_VLAN_PCP|
                         OFMatch.OFPFW_NW_PROTO|
                         OFMatch.OFPFW_NW_DST_ALL|
                         OFMatch.OFPFW_NW_TOS|
                         OFMatch.OFPFW_TP_DST|
                         OFMatch.OFPFW_TP_SRC);
      fm.setMatch(match);

      OFAction action=new OFActionOutput((short)2);

      fm.setActions(Collections.singletonList((OFAction)action));
      sw.getOutputStream().write(fm);
```

```
//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

      OFFlowMod fm1=new OFFlowMod();
      fm1.setCommand(OFFlowMod.OFPFC_ADD);
      fm1.setBufferId(-1);
      fm1.setIdleTimeout((short)30);
      fm1.setHardTimeout((short)30);
      fm1.setPriority((short)20);

      OFMatch match1 =OFMatch.load(pi.getPacketData(),
      pi.getInPort());

      match1.setDataLayerType(Ethernet.TYPE_ARP);
      match1.setInputPort((short)2);

      match1.setNetworkDestination(IPv4.toIPv4Address(ip2));
      match1.setWildcards(OFMatch.OFPFW_DL_DST|
                          OFMatch.OFPFW_DL_SRC|
                          OFMatch.OFPFW_DL_VLAN|
                          OFMatch.OFPFW_DL_VLAN_PCP|
                          OFMatch.OFPFW_NW_PROTO|
                          OFMatch.OFPFW_NW_SRC_ALL|
                          OFMatch.OFPFW_NW_TOS|
                          OFMatch.OFPFW_TP_DST|
                          OFMatch.OFPFW_TP_SRC);
      fm1.setMatch(match1);

      OFAction action1=new OFActionOutput((short)1);

      fm1.setActions(Collections.singletonList((OFAction)action1)
      );
            sw.getOutputStream().write(fm1);

      //////////////IP//////////////////

      OFFlowMod fm2=new OFFlowMod();
      fm2.setCommand(OFFlowMod.OFPFC_ADD);
      fm2.setBufferId(-1);
      fm2.setIdleTimeout((short)40);
      fm2.setHardTimeout((short)40);
      fm2.setPriority((short)20);

      OFMatch match2 =OFMatch.load(pi.getPacketData(),
      pi.getInPort());

      match2.setDataLayerType(Ethernet.TYPE_IPv4);
      match2.setInputPort((short)2);
      match2.setNetworkDestination(IPv4.toIPv4Address(ip2));
      match2.setWildcards(OFMatch.OFPFW_DL_DST|
                          OFMatch.OFPFW_DL_SRC|
                          OFMatch.OFPFW_DL_VLAN|
                          OFMatch.OFPFW_DL_VLAN_PCP|
                          OFMatch.OFPFW_NW_PROTO|
                          OFMatch.OFPFW_NW_SRC_ALL|
                          OFMatch.OFPFW_NW_TOS|
                          OFMatch.OFPFW_TP_DST|
                          OFMatch.OFPFW_TP_SRC);
      fm2.setMatch(match2);
```

```
        OFAction action2=new OFActionOutput((short)1);

        fm2.setActions(Collections.singletonList((OFAction)action2)
        );
        sw.getOutputStream().write(fm2);

//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo

        OFFlowMod fm3=new OFFlowMod();
        fm3.setCommand(OFFlowMod.OFPFC_ADD);
        fm3.setBufferId(-1);
        fm3.setIdleTimeout((short)30);
        fm3.setHardTimeout((short)30);
        fm3.setPriority((short)20);

        OFMatch match3 =OFMatch.load(pi.getPacketData(),
        pi.getInPort());

        match3.setDataLayerType(Ethernet.TYPE_IPv4);
        match3.setInputPort((short)1);
        match3.setNetworkSource(IPv4.toIPv4Address(ip2));
        match3.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_DST_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm3.setMatch(match3);

        OFAction action3=new OFActionOutput((short)2);
        fm3.setActions(Collections.singletonList((OFAction)action3)
        );
        sw.getOutputStream().write(fm3);
        }
}

    @Override
    public void addedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub
     }

    @Override
    public void removedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub
    }
}
```

ANEXO 2: Módulo filtrado por Mac desarrollado en Beacon

```
package net.beaconcontroller.filtromac;

import java.io.IOException;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.ConcurrentSkipListSet;

import net.beaconcontroller.packet.Ethernet;
import net.beaconcontroller.packet.IPv4;
import net.beaconcontroller.core.IBeaconProvider;
import net.beaconcontroller.core.IOFMessageListener;
import net.beaconcontroller.core.IOFSwitchListener;
import net.beaconcontroller.core.IOFSwitch;

import org.openflow.protocol.OFFlowMod;
import org.openflow.protocol.OFMatch;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFPacketIn;
import org.openflow.protocol.OFPacketOut;
import org.openflow.protocol.OFPort;
import org.openflow.protocol.action.OFAction;
import org.openflow.protocol.action.OFActionOutput;
import org.openflow.protocol.OFType;
import org.openflow.util.HexString;
import org.openflow.util.U16;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class FiltroMac implements IOFMessageListener,
IOFSwitchListener {

protected static Logger logger =
LoggerFactory.getLogger(FiltroMac.class);

protected IBeaconProvider beaconProvider;
protected Set<Integer> macAddresses = new

ConcurrentSkipListSet<Integer>();

protected Map<IOFSwitch, Map<Long,Short>> macTables = new
HashMap<IOFSwitch, Map<Long,Short>>();


public IBeaconProvider getBeaconProvider() {
        return beaconProvider;
}

public void setBeaconProvider(IBeaconProvider beaconProvider) {
        this.beaconProvider = beaconProvider;
}
```

```
public void startUp() {
    beaconProvider.addOFMessageListener(OFType.PACKET_IN, this);
}

public void shutDown() {
        beaconProvider.removeOFMessageListener(OFType.PACKET_IN,
this);
    }

public String getName() {
        return "filtromac";
}


public Command receive(IOFSwitch sw, OFMessage msg) throws
IOException {

        initMACTable(sw);
        OFPacketIn pi = (OFPacketIn) msg;
        FiltroMac(sw,pi);

        return Command.CONTINUE;
    }

private void initMACTable(IOFSwitch sw) {
Map<Long,Short> macTable = macTables.get(sw);

        if (macTable == null) {
            macTable = new HashMap<Long,Short>();
            macTables.put(sw, macTable);
          }
    }
public void FiltroMac(IOFSwitch sw, OFPacketIn pi) throws
IOException
    {

      Map<Long,Short> macTable =macTables.get(sw);

      //Crear el Match

      OFMatch match = OFMatch.load(pi.getPacketData(),
      pi.getInPort());

      //Recuperar la MAC de origen

      String MAC_ORIGEN
      =HexString.toHexString(match.getDataLayerSource());
      String M1="00:1c:c0:de:08:31";
      String M2="00:1c:c0:de:08:2d";
      String M3="00:1c:c0:55:2e:b4";

      //Condición que evalúa que host tendrán conexión

      if ((M1.equals( MAC_ORIGEN )||M2.equals( MAC_ORIGEN
      )||M3.equals( MAC_ORIGEN )))
      {

      //aprender el puerto de llegada de la MAC origen del
      //paquete
```

```
macTable.put(Ethernet.toLong(match.getDataLayerSource()),
pi.getInPort());

//Recuperar el puerto previamente aprendido para la
//dirección MAC destino del paquete

Short
puerto_salida=macTable.get(Ethernet.toLong(match.getDataLay
erDestination()));

   if (puerto_salida==null)
   {

      //Se debe inundar de paquetes a los host para
      //conocer la mac y el puerto destino y
      //crear el objeto packet-out  para transferir
      //paquetes enviados  por el controlador
      //desde el puerto especificado

       OFPacketOut po=new OFPacketOut();

      //Crear una acción de salida para realizar una
      //inundación de paquetes

      OFAction action = new
      OFActionOutput(OFPort.OFPP_FLOOD.getValue());

      //establecer las acciones en el paquet out

      po.setActions(Collections.singletonList(action));
      //Establecer el puerto de entrada del paquete
      po.setInPort(pi.getInPort());

      //Hacer referencia al paquete almacenado en el switch
      por medio de su //ID
       po.setBufferId(pi.getBufferId());

       if (pi.getBufferId()==OFPacketOut.BUFFER_ID_NONE)
       {
            //Si el paquete no fue almacenado en el switch,
            //debemos copiar los datos del paquete desde
            //el paquete de entrada PacketIn
            //tranferir los datos que contiene el paquete
            //de entrada al paquete de salida

            po.setPacketData((pi.getPacketData()));
       }
                  // enviar el packet-out a el switch
                 sw.getOutputStream().write(po);
      }
      else
      {
            //Si el puerto destino ya se conoce enviar el
            //flujo hacia él para lo cual se debe crear el
            //objeto FlowMod que permite modificar el
            //flujo existente en la tabla de flujo del
            //switch
```

```
                OFFlowMod fm = new OFFlowMod();
                fm.setBufferId(pi.getBufferId());

                // Anadir una entrada de flujo

                fm.setCommand(OFFlowMod.OFPFC_ADD);

                // Establecer el tiempo de expiracion de
                inactividad =idle_timeout=15 segundos

                fm.setIdleTimeout((short) 15);

                // Realizar el proceso del matching para
                obtener las coincidencias

                fm.setMatch(match);

                // Establecer la acción de salida

                OFAction action = new
                OFActionOutput(puerto_salida);
                fm.setActions(Collections.singletonList((OFActi
                on)action));

                // Escribir la entrada de flujo en el switch

                sw.getOutputStream().write(fm);

                if (pi.getBufferId() ==
                OFPacketOut.BUFFER_ID_NONE)
                {
                      //Si el paquete no fue almacenado en el
                      //switch primero se debe enviar un packet
                      //out posteriormente se envía el objeto
                      //Flowmod

                      OFPacketOut po = new OFPacketOut();

                      //Definir la acción determinada al puerto
                      //de salida

                      action=new OFActionOutput(puerto_salida);
                      po.setActions(Collections.singletonList(a
                      ction));
                      po.setBufferId(OFPacketOut.BUFFER_ID_NONE
                      );

                      po.setInPort(pi.getInPort());
                      po.setPacketData(pi.getPacketData());
                      sw.getOutputStream().write(po);
            }
      }
   }
}

 @Override
 public void addedSwitch(IOFSwitch sw) {
     // TODO Auto-generated method stub
```

```
    }

    @Override
    public void removedSwitch(IOFSwitch sw) {
        // TODO Auto-generated method stub

    }
}
```

ANEXO 3: Módulo filtrado por Mac a través de Rest Api desarrollado en Floodlight.

Código del archivo FiltroMac

```
package net.floodlightcontroller.filtromac;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.openflow.protocol.OFFlowMod;
import org.openflow.protocol.OFMatch;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFPacketIn;
import org.openflow.protocol.OFPacketOut;
import org.openflow.protocol.OFPort;
import org.openflow.protocol.OFType;
import org.openflow.protocol.action.OFAction;
import org.openflow.protocol.action.OFActionOutput;
import org.openflow.util.HexString;
import org.openflow.util.U16;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import net.floodlightcontroller.core.FloodlightContext;
import net.floodlightcontroller.core.IFloodlightProviderService;
import net.floodlightcontroller.core.IOFMessageListener;
import net.floodlightcontroller.core.IOFSwitch;
import net.floodlightcontroller.core.IListener.Command;
import
net.floodlightcontroller.core.module.FloodlightModuleContext;
import
net.floodlightcontroller.core.module.FloodlightModuleException;
import net.floodlightcontroller.core.module.IFloodlightModule;
import net.floodlightcontroller.core.module.IFloodlightService;
import net.floodlightcontroller.packet.Ethernet;
import net.floodlightcontroller.restserver.IRestApiService;

public class Filtromac implements IFloodlightModule,
IOFMessageListener, IFiltromacService {

	protected IFloodlightProviderService floodlightProvider;
	protected IRestApiService restApi;
	private ArrayList<String> buffer = new ArrayList<String>();
	protected Map<IOFSwitch, Map<Long,Short>> macTables = new
	HashMap<IOFSwitch, Map<Long,Short>>();
	protected static Logger log =
	LoggerFactory.getLogger(Filtromac.class);
	@Override
	public String getName() {
		// TODO Auto-generated method stub
```

```
            return "Filtromac";
      }


      @Override
      public boolean isCallbackOrderingPrereq(OFType type, String
      name) {
            // TODO Auto-generated method stub
            return false;
      }

      @Override
      public boolean isCallbackOrderingPostreq(OFType type,
      String name) {
            // TODO Auto-generated method stub
            return false;
      }

      private void initMACTable(IOFSwitch sw) {
      Map<Long,Short> macTable = macTables.get(sw);

        if (macTable == null) {
            macTable = new HashMap<Long,Short>();
            macTables.put(sw, macTable);
          }
    }

      @Override
public Command receive(IOFSwitch sw, OFMessage msg,
FloodlightContext cntx) {
      initMACTable(sw);
      Map<Long,Short> macTable =macTables.get(sw);
      OFPacketIn pi = (OFPacketIn) msg;
      OFMatch match = new OFMatch();
      match.loadFromPacket(pi.getPacketData(),pi.getInPort());

      if(buffer.contains(HexString.toHexString(match.getDataLayer
      Source())))
      {
      macTable.put(Ethernet.toLong(match.getDataLayerSource()),
      pi.getInPort());
      Short
      puerto_salida=macTable.get(Ethernet.toLong(match.getDataLay
      erDestination()));

            if (puerto_salida==null)
            {
                  OFPacketOut po = (OFPacketOut)
                  floodlightProvider.getOFMessageFactory().getMes
                  sage(OFType.PACKET_OUT);
                  po.setBufferId(pi.getBufferId());
                  po.setInPort(pi.getInPort());
                  // ESTABLECER LAS ACCIONES
                  OFActionOutput action = new
                  OFActionOutput().setPort(OFPort.OFPP_FLOOD.getV
                  alue());

                  po.setActions(Collections.singletonList((OFActi
                  on)action));
```

```
po.setActionsLength((short)
OFActionOutput.MINIMUM_LENGTH);

// EVALUAR SI EL SWITCH SOPORTA BUFFER
if (pi.getBufferId() ==
OFPacketOut.BUFFER_ID_NONE) {
      byte[] packetData = pi.getPacketData();

      po.setLength(U16.t(OFPacketOut.MINIMUM_LE
      NGTH + po.getActionsLength() +
      packetData.length));
      po.setPacketData(packetData);
 }
else
{

      po.setLength(U16.t(OFPacketOut.MINIMUM_LE
      NGTH + po.getActionsLength()));
 }

 try {
       sw.write(po, cntx);
  } catch (IOException e) {
        log.error("Error al escribir el mensaje
        packet-out", e);
        }
   }
else
{
OFFlowMod fm = new OFFlowMod();
fm.setCommand(OFFlowMod.OFPFC_ADD);
fm.setBufferId(pi.getBufferId());
fm.setIdleTimeout((short) 15);
fm.setMatch(match);

List<OFAction> actions = new
ArrayList<OFAction>();
OFAction action=new
OFActionOutput().setPort(puerto_salida);


fm.setLength((short)(OFFlowMod.MINIMUM_LENGTH+O
FActionOutput.MINIMUM_LENGTH));
actions.add(action);
fm.setActions(actions);

try {
      sw.write(fm, cntx);
      } catch (IOException e) {
      log.error("Error al escribir la   entrada
      de flujo", e);
  }

if(pi.getBufferId()==OFPacketOut.BUFFER_ID_NONE
)
{
      OFPacketOut po = (OFPacketOut)
      floodlightProvider.getOFMessageFactory().
      getMessage(OFType.PACKET_OUT);
```

```
                    OFActionOutput action1 = new
                    OFActionOutput().setPort(puerto_salida);

                    po.setActions(Collections.singletonList((
                    OFAction)action1));
                    po.setActionsLength((short)
                    OFActionOutput.MINIMUM_LENGTH);
                  po.setBufferId(OFPacketOut.BUFFER_ID_NONE);
                    po.setInPort(pi.getInPort());
                    byte[] packetData = pi.getPacketData();

                    po.setLength(U16.t(OFPacketOut.MINIMUM_LE
                    NGTH + po.getActionsLength() +
                    packetData.length));
                    po.setPacketData(packetData);

                    try {
                          sw.write(po, cntx);
                          } catch (IOException e) {
                          log.error("Error al escribir la
                          entrada de flujo", e);
                          }

                    }
              }
        }
              return Command.CONTINUE;
}

@Override
public Collection<Class<? extends IFloodlightService>>
getModuleServices() {
      Collection<Class<? extends IFloodlightService>> l =
      new ArrayList<Class<? extends IFloodlightService>>();
      l.add(IFiltromacService.class);
      return l;
}

@Override
public Map<Class<? extends IFloodlightService>,
IFloodlightService> getServiceImpls() {
      Map<Class<? extends IFloodlightService>,
      IFloodlightService> m = new HashMap<Class<? extends
      IFloodlightService>,IFloodlightService>();
      m.put(IFiltromacService.class,this);
      return m;
}

@Override
public Collection<Class<? extends IFloodlightService>>
getModuleDependencies() {
      Collection<Class<? extends IFloodlightService>> l =
      new ArrayList<Class<? extends IFloodlightService>>();
      l.add(IFloodlightProviderService.class);
      l.add(IRestApiService.class);
      return l;
}

@Override
```

```
public void init(FloodlightModuleContext context)throws
FloodlightModuleException {
      floodlightProvider =
      context.getServiceImpl(IFloodlightProviderService.cla
      ss);
      restApi =
      context.getServiceImpl(IRestApiService.class);
}

@Override
public void startUp(FloodlightModuleContext context)throws
FloodlightModuleException {
      restApi.addRestletRoutable(new
      FiltromacWebRoutable());
floodlightProvider.addOFMessageListener(OFType.PACKET_IN,
this);
}

//-------------------------------------------------------
// METODOS PARA VISUALIZAR, GUARDAR, Y ELIMINAR LAS
DIRECCIONES MAC
//-------------------------------------------------------

@Override
public String getBuffer() {
      String ret;
      String key;

      // "{\"key1\":\"value1\",\"key2\":\"value2\"}"
      ret = "{";
      for (int i=0; i < buffer.size(); i++) {
            key = buffer.get(i);
            ret += "\"" + key + "\""+ ":";
            ret += "\"\"";
            if (i < (buffer.size() - 1))
                  ret += ",";
      }
      ret += "}";

      return ret;
}

@Override
public void showBuffer() {
      for (String entry: buffer) {
            System.out.println(entry);
      }

}

@Override
public void setBuffer(String key) {
      buffer.add(key);
}

@Override
public void removeEntry(String key) {
      for (int i=0; i < buffer.size(); i++) {
            if ((buffer.get(i)).equals(key)) {
```

```
                        buffer.remove(i);
                        return;
                  }
            }
      }

      @Override
      public void removeAllEntries() {
            buffer.clear();
      }
}
```

Código del archivo FiltromacResource

```
package net.floodlightcontroller.filtromac;

import java.io.IOException;

import org.restlet.resource.Delete;
import org.restlet.resource.Get;
import org.restlet.resource.Post;
import org.restlet.resource.ServerResource;

import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.core.JsonToken;
import com.fasterxml.jackson.databind.MappingJsonFactory;

public class FiltromacResource extends ServerResource {

      // Obtener la solicitud y devolver los datos
      @Get("json")
      public Object handleReqeust() {
            IFiltromacService filtro =

      (IFiltromacService)getContext().getAttributes().

      get(IFiltromacService.class.getCanonicalName());

            return filtro.getBuffer();
      }

      // anadir datos
      @Post
      public String handlePost(String str) {
            String key,value;
            IFiltromacService filtro =

      (IFiltromacService)getContext().getAttributes().

      get(IFiltromacService.class.getCanonicalName());

            // Comparar la cadena string recivida de una manera
            serializada y guardarla en el buffer
            MappingJsonFactory f = new MappingJsonFactory();
            JsonParser jp;
```

```
		try {
			// obtener el JSon
			jp = f.createJsonParser(str);

			// salta la primera llaves '{'
			jp.nextToken();


			while (jp.nextToken() != JsonToken.END_OBJECT){

				// obtener el key
				key = jp.getCurrentName();

				// obtener el value
				jp.nextToken();

				// guardar el key en el buffer
				filtro.setBuffer(key);
			}
		} catch (IOException e) {
			System.err.println(e);
		}

		return "{\"add\":\"success\"}";
	}

	// Borrar los datos
	@Delete
	public String handleDelete(String str) {
		String identifier, key;
		IFiltromacService filtro =

	(IFiltromacService)getContext().getAttributes().

	get(IFiltromacService.class.getCanonicalName());


		MappingJsonFactory f = new MappingJsonFactory();
		JsonParser jp;

		try {
			// obtener el JSon
			jp = f.createJsonParser(str);

			// saltar la primera llaves '{'
			jp.nextToken();


			while (jp.nextToken() != JsonToken.END_OBJECT){
				//obtener el identificador
				identifier = jp.getCurrentName();

				// obtener el key
				jp.nextToken();
				key = jp.getText();

				// remover la key del buffer
				filtro.removeEntry(key);
```

```
                }
            } catch (IOException e) {
                System.err.println(e);
            }

            return "{\"delete\":\"success\"}";
        }
}
```

Código del archivo FiltomacWebRoutable

```
package net.floodlightcontroller.filtromac;

import org.restlet.Context;
import org.restlet.Restlet;
import org.restlet.routing.Router;

import net.floodlightcontroller.restserver.RestletRoutable;

public class FiltromacWebRoutable implements RestletRoutable {

      @Override
      public Restlet getRestlet(Context context) {
            Router router = new Router(context);
            //media la palabra además se adicionara una mac a la
            lista
            router.attach("/addmac", FiltromacResource.class);
            //mediante la palabra crear se borra cada mac que
            queramos eliminar de la lista
            router.attach("/clear",FiltromacClearResource.class);
            return router;
      }

      @Override
      public String basePath() {
            // Las palabras /wm/filtro serán la base para las
            peticiones curl
            return "/wm/filtro";
      }
}
```

Código del archivo IFloodlightService

```
package net.floodlightcontroller.filtromac;

import net.floodlightcontroller.core.module.IFloodlightService;

public interface IFiltromacService extends IFloodlightService {

	//función para visualizar las direcciones mac
	public String getBuffer();
	//función para añadir las direcciones mac
	public void setBuffer(String key);
	//función para remover una dirección mac
	public void removeEntry(String key);
	//función para borrar todas las direcciones mac
	public void removeAllEntries();
	public void showBuffer();
}
```

Código del archivo FiltromacClearResource

```
package net.floodlightcontroller.filtromac;

import org.restlet.resource.Get;
import org.restlet.resource.ServerResource;

public class FiltromacClearResource extends ServerResource {

	// get request and remove all entries in a buffer
	@Get("json")
	public Object handleRequest() {
		IFiltromacService filtro =

	(IFiltromacService)getContext().getAttributes().

	get(IFiltromacService.class.getCanonicalName());

		filtro.removeAllEntries();
		return "{\"delete all\":\"success\"}";
	}
}
```

ANEXO 4: Módulo para inserción de reglas de flujo desarrollado en Floodlight.

```
package net.floodlightcontroller.modulo;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Collections;
import java.util.List;
import java.util.Map;

import net.floodlightcontroller.core.FloodlightContext;
import net.floodlightcontroller.core.IFloodlightProviderService;
import net.floodlightcontroller.core.IOFMessageListener;
import net.floodlightcontroller.core.IOFSwitch;
import
net.floodlightcontroller.core.module.FloodlightModuleContext;
import
net.floodlightcontroller.core.module.FloodlightModuleException;
import net.floodlightcontroller.core.module.IFloodlightModule;
import net.floodlightcontroller.core.module.IFloodlightService;
import net.floodlightcontroller.counter.ICounterStoreService;
import net.floodlightcontroller.learningswitch.LearningSwitch;
import net.floodlightcontroller.packet.Ethernet;
import net.floodlightcontroller.packet.IPv4;

import org.openflow.protocol.OFFlowMod;
import org.openflow.protocol.OFMatch;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFPacketIn;
import org.openflow.protocol.OFPacketOut;
import org.openflow.protocol.OFPort;
import org.openflow.protocol.OFStatisticsRequest;
import org.openflow.protocol.OFType;
import org.openflow.protocol.action.OFAction;
import org.openflow.protocol.action.OFActionOutput;
import org.openflow.protocol.statistics.OFFlowStatisticsRequest;
import org.openflow.util.U16;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class Modulo implements IFloodlightModule,
IOFMessageListener {
protected IFloodlightProviderService floodlightProvider;
protected static Logger log =
LoggerFactory.getLogger(Modulo.class);
public void setFloodlightProvider(IFloodlightProviderService
floodlightProvider) {
        this.floodlightProvider = floodlightProvider;
    }

    @Override
    public String getName() {
        return Modulo.class.getPackage().getName();
    }
```

```
public Command receive(IOFSwitch sw, OFMessage msg,
FloodlightContext cntx) {
        OFPacketIn pi = (OFPacketIn) msg;
        reglas(sw,pi,cntx);
        return Command.CONTINUE;
    }
private void reglas (IOFSwitch sw, OFPacketIn pi,
FloodlightContext cntx) {

    //INGRESE DATOS
    /**SWITCH1*/
    //DPID
    long id1=5894009871857728L;
    //long id1=1;
    //IP HOST1
    String ip1="192.168.111.13";
    //IP HOST2
    String ip2="192.168.111.14";
    //MAC H1
    //String mac1="00:1c:c0:de:08:31";
    //MAC H2
    //String mac2="c";

    /**SWITCH2*/
    //DPID
    long id2=5629651061111680L;
    //long id2=2;
    //IP HOST3
    String ip3="192.168.111.15";
    //MAC H3
    //String mac3="00:1c:c0:de:08:2d";

    /**SWITCH3*/
    //DPID
    long id3=3;
    //IP HOST4
    String ip4="192.168.111.16";
    //MAC H4
    //String mac4="00:1c:c0:55:2e:b4";

if(sw.getId()==id1)
    {
        //////////////ARP//////////////////
        OFFlowMod fm=new OFFlowMod();
        fm.setCommand(OFFlowMod.OFPFC_ADD);
        fm.setBufferId(-1);
        fm.setIdleTimeout((short)30);
        fm.setHardTimeout((short)30);
        fm.setPriority((short)20);
        OFMatch match = new OFMatch();
        match.loadFromPacket(pi.getPacketData(),pi.getInPort());

        match.setDataLayerType(Ethernet.TYPE_ARP);
        match.setInputPort((short)3);
        match.setNetworkDestination(IPv4.toIPv4Address(ip3));
        match.setWildcards(OFMatch.OFPFW_DL_DST|
                           OFMatch.OFPFW_DL_SRC|
                           OFMatch.OFPFW_DL_VLAN|
                           OFMatch.OFPFW_DL_VLAN_PCP|
```

```
                        OFMatch.OFPFW_NW_PROTO|
                        OFMatch.OFPFW_NW_SRC_ALL|
                        OFMatch.OFPFW_NW_TOS|
                        OFMatch.OFPFW_TP_DST|
                        OFMatch.OFPFW_TP_SRC);
        fm.setMatch(match);

        List<OFAction> action = new ArrayList<OFAction>();
        OFAction actions=new OFActionOutput((short) 1);

        fm.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionOu
        tput.MINIMUM_LENGTH));
        action.add(actions);
        fm.setActions(action);

        try {
                sw.write(fm, null);
            } catch (IOException e) {
                e.printStackTrace();
            }
//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo
        OFFlowMod fm1=new OFFlowMod();
        fm1.setCommand(OFFlowMod.OFPFC_ADD);
        fm1.setBufferId(-1);
        fm1.setIdleTimeout((short)30);
        fm1.setHardTimeout((short)30);
        fm1.setPriority((short)20);

        OFMatch match1 = new OFMatch();

        match1.loadFromPacket(pi.getPacketData(),pi.getInPort())
        ;

        match1.setDataLayerType(Ethernet.TYPE_ARP);
        match1.setInputPort((short)1);
        match1.setNetworkSource(IPv4.toIPv4Address(ip3));
        match1.setWildcards(OFMatch.OFPFW_DL_DST|
                        OFMatch.OFPFW_DL_SRC|
                        OFMatch.OFPFW_DL_VLAN|
                        OFMatch.OFPFW_DL_VLAN_PCP|
                        OFMatch.OFPFW_NW_PROTO|
                        OFMatch.OFPFW_NW_DST_ALL|
                        OFMatch.OFPFW_NW_TOS|
                        OFMatch.OFPFW_TP_DST|
                        OFMatch.OFPFW_TP_SRC);
        fm1.setMatch(match1);
        List<OFAction> action1 = new ArrayList<OFAction>();
        OFAction actions1=new OFActionOutput((short) 3);

        fm1.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
        utput.MINIMUM_LENGTH));
        action1.add(actions1);
        fm1.setActions(action1);

        try {
                sw.write(fm1, null);
            } catch (IOException e) {
                e.printStackTrace();
            }
```

```
        //////////////IP//////////////////

        OFFlowMod fm2=new OFFlowMod();
        fm2.setCommand(OFFlowMod.OFPFC_ADD);
        fm2.setBufferId(-1);
        fm2.setIdleTimeout((short)40);
        fm2.setHardTimeout((short)40);
        fm2.setPriority((short)20);

        OFMatch match2 = new OFMatch();

        match2.loadFromPacket(pi.getPacketData(),pi.getInPort())
        ;

        match2.setDataLayerType(Ethernet.TYPE_IPv4);
        match2.setInputPort((short)3);
        match2.setNetworkDestination(IPv4.toIPv4Address(ip3));
        match2.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm2.setMatch(match2);

        List<OFAction> action2 = new ArrayList<OFAction>();
        OFAction actions2=new OFActionOutput((short) 1);

        fm2.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
        utput.MINIMUM_LENGTH));
        action2.add(actions2);
        fm2.setActions(action2);

        try {
                  sw.write(fm2, null);
            } catch (IOException e) {
                  e.printStackTrace();
            }

    //ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo
        OFFlowMod fm3=new OFFlowMod();
        fm3.setCommand(OFFlowMod.OFPFC_ADD);
        fm3.setBufferId(-1);
        fm3.setIdleTimeout((short)30);
        fm3.setHardTimeout((short)30);
        fm3.setPriority((short)20);

        OFMatch match3 = new OFMatch();

        match3.loadFromPacket(pi.getPacketData(),pi.getInPort())
        ;

        match3.setDataLayerType(Ethernet.TYPE_IPv4);
        match3.setInputPort((short)1);
        match3.setNetworkSource(IPv4.toIPv4Address(ip3));
        match3.setWildcards(OFMatch.OFPFW_DL_DST|
```

```
                    OFMatch.OFPFW_DL_SRC|
                    OFMatch.OFPFW_DL_VLAN|
                    OFMatch.OFPFW_DL_VLAN_PCP|
                    OFMatch.OFPFW_NW_PROTO|
                    OFMatch.OFPFW_NW_DST_ALL|
                    OFMatch.OFPFW_NW_TOS|
                    OFMatch.OFPFW_TP_DST|
                    OFMatch.OFPFW_TP_SRC);
fm3.setMatch(match3);

List<OFAction> action3 = new ArrayList<OFAction>();
OFAction actions3=new OFActionOutput((short) 3);

fm3.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
utput.MINIMUM_LENGTH));
action3.add(actions3);
fm3.setActions(action3);

try {
          sw.write(fm3, null);
    } catch (IOException e) {
          e.printStackTrace();
    }

//-------------REGLAS PARA EL HOST 4-----------------

//////////////ARP//////////////////
OFFlowMod fm4=new OFFlowMod();
fm4.setCommand(OFFlowMod.OFPFC_ADD);
fm4.setBufferId(-1);
fm4.setIdleTimeout((short)30);
fm4.setHardTimeout((short)30);
fm4.setPriority((short)20);

OFMatch match4 = new OFMatch();

match4.loadFromPacket(pi.getPacketData(),pi.getInPort())
;

match4.setDataLayerType(Ethernet.TYPE_ARP);
match4.setInputPort((short)4);
match4.setNetworkDestination(IPv4.toIPv4Address(ip4));
match4.setWildcards(OFMatch.OFPFW_DL_DST|
                    OFMatch.OFPFW_DL_SRC|
                    OFMatch.OFPFW_DL_VLAN|
                    OFMatch.OFPFW_DL_VLAN_PCP|
                    OFMatch.OFPFW_NW_PROTO|
                    OFMatch.OFPFW_NW_SRC_ALL|
                    OFMatch.OFPFW_NW_TOS|
                    OFMatch.OFPFW_TP_DST|
                    OFMatch.OFPFW_TP_SRC);
fm4.setMatch(match4);

List<OFAction> action4 = new ArrayList<OFAction>();
OFAction actions4=new OFActionOutput((short) 2);

fm4.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
utput.MINIMUM_LENGTH));
action4.add(actions4);
```

```
        fm4.setActions(action4);
        try {
                  sw.write(fm4, null);
              } catch (IOException e) {
                  e.printStackTrace();
              }
//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo
        OFFlowMod fm5=new OFFlowMod();
        fm5.setCommand(OFFlowMod.OFPFC_ADD);
        fm5.setBufferId(-1);
        fm5.setIdleTimeout((short)30);
        fm5.setHardTimeout((short)30);
        fm5.setPriority((short)20);

        OFMatch match5 = new OFMatch();

        match5.loadFromPacket(pi.getPacketData(),pi.getInPort())
        ;

        match5.setDataLayerType(Ethernet.TYPE_ARP);
        match5.setInputPort((short)2);
        match5.setNetworkDestination(IPv4.toIPv4Address(ip2));
        match5.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm5.setMatch(match5);

        List<OFAction> action5 = new ArrayList<OFAction>();
        OFAction actions5=new OFActionOutput((short) 4);

        fm5.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
        utput.MINIMUM_LENGTH));
        action5.add(actions5);
        fm5.setActions(action5);

        try {
                  sw.write(fm5, null);
              } catch (IOException e) {
                  e.printStackTrace();
              }

        //////////////IP//////////////////

        OFFlowMod fm6=new OFFlowMod();
        fm6.setCommand(OFFlowMod.OFPFC_ADD);
        fm6.setBufferId(-1);
        fm6.setIdleTimeout((short)40);
        fm6.setHardTimeout((short)40);
        fm6.setPriority((short)20);

        OFMatch match6 = new OFMatch();
```

```
        match6.loadFromPacket(pi.getPacketData(),pi.getInPort())
        ;

        match6.setDataLayerType(Ethernet.TYPE_IPv4);
        match6.setInputPort((short)4);
        match6.setNetworkDestination(IPv4.toIPv4Address(ip4));
        System.out.println(match6);
        match6.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
        fm6.setMatch(match6);

        List<OFAction> action6 = new ArrayList<OFAction>();
        OFAction actions6=new OFActionOutput((short) 2);

        fm6.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
        utput.MINIMUM_LENGTH));
        action6.add(actions6);
        fm6.setActions(action6);

        try {
                  sw.write(fm6, null);
            } catch (IOException e) {
                  e.printStackTrace();
            }

//ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo
        OFFlowMod fm7=new OFFlowMod();
        fm7.setCommand(OFFlowMod.OFPFC_ADD);
        fm7.setBufferId(-1);
        fm7.setIdleTimeout((short)30);
        fm7.setHardTimeout((short)30);
        fm7.setPriority((short)20);

        OFMatch match7 = new OFMatch();

        match7.loadFromPacket(pi.getPacketData(),pi.getInPort())
        ;

        match7.setDataLayerType(Ethernet.TYPE_IPv4);
        match7.setInputPort((short)2);
        match7.setNetworkDestination(IPv4.toIPv4Address(ip2));
        System.out.println(match7);
        match7.setWildcards(OFMatch.OFPFW_DL_DST|
                            OFMatch.OFPFW_DL_SRC|
                            OFMatch.OFPFW_DL_VLAN|
                            OFMatch.OFPFW_DL_VLAN_PCP|
                            OFMatch.OFPFW_NW_PROTO|
                            OFMatch.OFPFW_NW_SRC_ALL|
                            OFMatch.OFPFW_NW_TOS|
                            OFMatch.OFPFW_TP_DST|
                            OFMatch.OFPFW_TP_SRC);
```

```
        fm7.setMatch(match7);

        List<OFAction> action7 = new ArrayList<OFAction>();
        OFAction actions7=new OFActionOutput((short) 4);

        fm7.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
        utput.MINIMUM_LENGTH));
        action7.add(actions7);
        fm7.setActions(action7);

        try {
                sw.write(fm7, null);
            } catch (IOException e) {
                e.printStackTrace();
            }
    }

    if(sw.getId()==id2)
    {
            OFFlowMod fm=new OFFlowMod();
            fm.setCommand(OFFlowMod.OFPFC_ADD);
            fm.setBufferId(-1);
            fm.setIdleTimeout((short)30);
            fm.setHardTimeout((short)30);
            fm.setPriority((short)20);

            OFMatch match = new OFMatch();

            match.loadFromPacket(pi.getPacketData(),pi.getInPort(
            ));

            match.setDataLayerType(Ethernet.TYPE_ARP);
            match.setInputPort((short)2);
            match.setNetworkSource(IPv4.toIPv4Address(ip1));
            match.setWildcards(OFMatch.OFPFW_DL_DST|
                               OFMatch.OFPFW_DL_SRC|
                               OFMatch.OFPFW_DL_VLAN|
                               OFMatch.OFPFW_DL_VLAN_PCP|
                               OFMatch.OFPFW_NW_PROTO|
                               OFMatch.OFPFW_NW_DST_ALL|
                               OFMatch.OFPFW_NW_TOS|
                               OFMatch.OFPFW_TP_DST|
                               OFMatch.OFPFW_TP_SRC);
            fm.setMatch(match);
            List<OFAction> action = new ArrayList<OFAction>();
            OFAction actions=new OFActionOutput((short) 1);

            fm.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActio
            nOutput.MINIMUM_LENGTH));
            action.add(actions);
            fm.setActions(action);

            try {
                  sw.write(fm, null);
            } catch (IOException e) {
                  e.printStackTrace();
            }

//ooooooooooooooooo--FLUJO DE REGRESO--oooooooooooooooooooooo
```

```
    OFFlowMod fm1=new OFFlowMod();
    fm1.setCommand(OFFlowMod.OFPFC_ADD);
    fm1.setBufferId(-1);
    fm1.setIdleTimeout((short)30);
    fm1.setHardTimeout((short)30);
    fm1.setPriority((short)20);

    OFMatch match1 = new OFMatch();

    match1.loadFromPacket(pi.getPacketData(),pi.getInPort
    ());

    match1.setDataLayerType(Ethernet.TYPE_ARP);
    match1.setInputPort((short)1);

    match1.setNetworkDestination(IPv4.toIPv4Address(ip1))
    ;
    match1.setWildcards(OFMatch.OFPFW_DL_DST|
                        OFMatch.OFPFW_DL_SRC|
                        OFMatch.OFPFW_DL_VLAN|
                        OFMatch.OFPFW_DL_VLAN_PCP|
                        OFMatch.OFPFW_NW_PROTO|
                        OFMatch.OFPFW_NW_SRC_ALL|
                        OFMatch.OFPFW_NW_TOS|
                        OFMatch.OFPFW_TP_DST|
                        OFMatch.OFPFW_TP_SRC);
    fm1.setMatch(match1);

    List<OFAction> action1 = new ArrayList<OFAction>();
    OFAction actions1=new OFActionOutput((short) 2);

    fm1.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActi
    onOutput.MINIMUM_LENGTH));
    action1.add(actions1);
    fm1.setActions(action1);

    try {
          sw.write(fm1, null);
    } catch (IOException e) {
          e.printStackTrace();
    }

//////////////IP//////////////////

    OFFlowMod fm2=new OFFlowMod();
    fm2.setCommand(OFFlowMod.OFPFC_ADD);
    fm2.setBufferId(-1);
    fm2.setIdleTimeout((short)30);
    fm2.setHardTimeout((short)30);
    fm2.setPriority((short)20);

    OFMatch match2 = new OFMatch();

    match2.loadFromPacket(pi.getPacketData(),pi.getInPort
    ());

    match2.setDataLayerType(Ethernet.TYPE_IPv4);
    match2.setInputPort((short)2);
```

```
            match2.setNetworkSource(IPv4.toIPv4Address(ip1));
            match2.setWildcards(OFMatch.OFPFW_DL_DST|
                                OFMatch.OFPFW_DL_SRC|
                                OFMatch.OFPFW_DL_VLAN|
                                OFMatch.OFPFW_DL_VLAN_PCP|
                                OFMatch.OFPFW_NW_PROTO|
                                OFMatch.OFPFW_NW_DST_ALL|
                                OFMatch.OFPFW_NW_TOS|
                                OFMatch.OFPFW_TP_DST|
                                OFMatch.OFPFW_TP_SRC);
            fm2.setMatch(match2);

            List<OFAction> action2 = new ArrayList<OFAction>();
            OFAction actions2=new OFActionOutput((short) 1);

            fm2.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActi
            onOutput.MINIMUM_LENGTH));
            action2.add(actions2);
            fm2.setActions(action2);

            try {
                  sw.write(fm2, null);
            } catch (IOException e) {
                  e.printStackTrace();
            }

      //ooooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo
            OFFlowMod fm3=new OFFlowMod();
            fm3.setCommand(OFFlowMod.OFPFC_ADD);
            fm3.setBufferId(-1);
            fm3.setIdleTimeout((short)30);
            fm3.setHardTimeout((short)30);
            fm3.setPriority((short)20);

            OFMatch match3 = new OFMatch();

            match3.loadFromPacket(pi.getPacketData(),pi.getInPort
            ());

            match3.setDataLayerType(Ethernet.TYPE_IPv4);
            match3.setInputPort((short)1);

            match3.setNetworkDestination(IPv4.toIPv4Address(ip1))
            ;
            match3.setWildcards(OFMatch.OFPFW_DL_DST|
                                OFMatch.OFPFW_DL_SRC|
                                OFMatch.OFPFW_DL_VLAN|
                                OFMatch.OFPFW_DL_VLAN_PCP|
                                OFMatch.OFPFW_NW_PROTO|
                                OFMatch.OFPFW_NW_SRC_ALL|
                                OFMatch.OFPFW_NW_TOS|
                                OFMatch.OFPFW_TP_DST|
                                OFMatch.OFPFW_TP_SRC);
            fm3.setMatch(match3);

            List<OFAction> action3 = new ArrayList<OFAction>();
            OFAction actions3=new OFActionOutput((short) 2);
```

```
            fm3.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActi
            onOutput.MINIMUM_LENGTH));
            action3.add(actions3);
            fm3.setActions(action3);

            try {
                  sw.write(fm3, null);
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
      if(sw.getId()==id3)
      {
      //////////////ARP//////////////////
      OFFlowMod fm=new OFFlowMod();
      fm.setCommand(OFFlowMod.OFPFC_ADD);
      fm.setBufferId(-1);
      fm.setIdleTimeout((short)30);
      fm.setHardTimeout((short)30);
      fm.setPriority((short)20);

      OFMatch match = new OFMatch();
      match.loadFromPacket(pi.getPacketData(),pi.getInPort());

      match.setDataLayerType(Ethernet.TYPE_ARP);
      match.setInputPort((short)1);
      match.setNetworkSource(IPv4.toIPv4Address(ip2));
      match.setWildcards(OFMatch.OFPFW_DL_DST|
                         OFMatch.OFPFW_DL_SRC|
                         OFMatch.OFPFW_DL_VLAN|
                         OFMatch.OFPFW_DL_VLAN_PCP|
                         OFMatch.OFPFW_NW_PROTO|
                         OFMatch.OFPFW_NW_DST_ALL|
                         OFMatch.OFPFW_NW_TOS|
                         OFMatch.OFPFW_TP_DST|
                         OFMatch.OFPFW_TP_SRC);
      fm.setMatch(match);

      List<OFAction> action = new ArrayList<OFAction>();
      OFAction actions=new OFActionOutput((short) 2);

       fm.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionOu
       tput.MINIMUM_LENGTH));
      action.add(actions);
      fm.setActions(action);

      try {
                  sw.write(fm, null);
            } catch (IOException e) {
                  e.printStackTrace();
            }

//ooooooooooooooooo--FLUJO DE REGRESO--oooooooooooooooooooooo
      OFFlowMod fm1=new OFFlowMod();
      fm1.setCommand(OFFlowMod.OFPFC_ADD);
      fm1.setBufferId(-1);
      fm1.setIdleTimeout((short)30);
      fm1.setHardTimeout((short)30);
```

```
fm1.setPriority((short)20);

OFMatch match1 = new OFMatch();

match1.loadFromPacket(pi.getPacketData(),pi.getInPort())
;

match1.setDataLayerType(Ethernet.TYPE_ARP);
match1.setInputPort((short)2);
match1.setNetworkDestination(IPv4.toIPv4Address(ip2));
match1.setWildcards(OFMatch.OFPFW_DL_DST|
                    OFMatch.OFPFW_DL_SRC|
                    OFMatch.OFPFW_DL_VLAN|
                    OFMatch.OFPFW_DL_VLAN_PCP|
                    OFMatch.OFPFW_NW_PROTO|
                    OFMatch.OFPFW_NW_SRC_ALL|
                    OFMatch.OFPFW_NW_TOS|
                    OFMatch.OFPFW_TP_DST|
                    OFMatch.OFPFW_TP_SRC);
fm1.setMatch(match1);

List<OFAction> action1 = new ArrayList<OFAction>();
OFAction actions1=new OFActionOutput((short) 1);

fm1.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
utput.MINIMUM_LENGTH));
action1.add(actions1);
fm1.setActions(action1);

try {
          sw.write(fm1, null);
    } catch (IOException e) {
          e.printStackTrace();
    }

//////////////IP//////////////////

OFFlowMod fm2=new OFFlowMod();
fm2.setCommand(OFFlowMod.OFPFC_ADD);
fm2.setBufferId(-1);
fm2.setIdleTimeout((short)40);
fm2.setHardTimeout((short)40);
fm2.setPriority((short)20);

OFMatch match2 = new OFMatch();

match2.loadFromPacket(pi.getPacketData(),pi.getInPort())
;

match2.setDataLayerType(Ethernet.TYPE_IPv4);
match2.setInputPort((short)2);
match2.setNetworkDestination(IPv4.toIPv4Address(ip2));
match2.setWildcards(OFMatch.OFPFW_DL_DST|
                    OFMatch.OFPFW_DL_SRC|
                    OFMatch.OFPFW_DL_VLAN|
                    OFMatch.OFPFW_DL_VLAN_PCP|
                    OFMatch.OFPFW_NW_PROTO|
                    OFMatch.OFPFW_NW_SRC_ALL|
                    OFMatch.OFPFW_NW_TOS|
```

```
                                OFMatch.OFPFW_TP_DST|
                                OFMatch.OFPFW_TP_SRC);
            fm2.setMatch(match2);

            List<OFAction> action2 = new ArrayList<OFAction>();
            OFAction actions2=new OFActionOutput((short) 1);

            fm2.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
            utput.MINIMUM_LENGTH));
            action2.add(actions2);
            fm2.setActions(action2);

            try {
                      sw.write(fm2, null);
                  } catch (IOException e) {
                      e.printStackTrace();
                  }
        //oooooooooooooooo--FLUJO DE REGRESO--ooooooooooooooooooooo
            OFFlowMod fm3=new OFFlowMod();
            fm3.setCommand(OFFlowMod.OFPFC_ADD);
            fm3.setBufferId(-1);
            fm3.setIdleTimeout((short)30);
            fm3.setHardTimeout((short)30);
            fm3.setPriority((short)20);

            OFMatch match3 = new OFMatch();

            match3.loadFromPacket(pi.getPacketData(),pi.getInPort())
            ;

            match3.setDataLayerType(Ethernet.TYPE_IPv4);
            match3.setInputPort((short)1);
            match3.setNetworkSource(IPv4.toIPv4Address(ip2));
            match3.setWildcards(OFMatch.OFPFW_DL_DST|
                                OFMatch.OFPFW_DL_SRC|
                                OFMatch.OFPFW_DL_VLAN|
                                OFMatch.OFPFW_DL_VLAN_PCP|
                                OFMatch.OFPFW_NW_PROTO|
                                OFMatch.OFPFW_NW_DST_ALL|
                                OFMatch.OFPFW_NW_TOS|
                                OFMatch.OFPFW_TP_DST|
                                OFMatch.OFPFW_TP_SRC);
            fm3.setMatch(match3);

            List<OFAction> action3 = new ArrayList<OFAction>();
            OFAction actions3=new OFActionOutput((short) 2);

            fm3.setLength((short)(OFFlowMod.MINIMUM_LENGTH+OFActionO
            utput.MINIMUM_LENGTH));
            action3.add(actions3);
            fm3.setActions(action3);

            try {
                      sw.write(fm3, null);
                  } catch (IOException e) {
                      e.printStackTrace();
                  }
      }
}
```

```
    @Override
      public boolean isCallbackOrderingPrereq(OFType type, String
      name) {
        return false;
      }

    @Override
      public boolean isCallbackOrderingPostreq(OFType type,
      String name) {
        return false;
      }

    // IFloodlightModule

    @Override
      public Collection<Class<? extends IFloodlightService>>
      getModuleServices() {
        // We don't provide any services, return null
        return null;
      }

    @Override
      public Map<Class<? extends IFloodlightService>,
      IFloodlightService>
            getServiceImpls() {
            return null;
      }

    @Override
      public Collection<Class<? extends IFloodlightService>>
            getModuleDependencies() {
            Collection<Class<? extends IFloodlightService>> l =
            new ArrayList<Class<? extends IFloodlightService>>();
            l.add(IFloodlightProviderService.class);
            return l;
      }

    @Override
      public void init(FloodlightModuleContext context)
      throws FloodlightModuleException {
            floodlightProvider =
            context.getServiceImpl(IFloodlightProviderService.cla
            ss);
    }

    @Override
    public void startUp(FloodlightModuleContext context) {

floodlightProvider.addOFMessageListener(OFType.PACKET_IN, this);
    }
}
```

ANEXO 5: Módulo balanceador de carga desarrollado en Floodlight

```
package net.floodlightcontroller.balanceocarga;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Map;

import org.openflow.protocol.OFFlowMod;
import org.openflow.protocol.OFMatch;
import org.openflow.protocol.OFMessage;
import org.openflow.protocol.OFPacketIn;
import org.openflow.protocol.OFPacketOut;
import org.openflow.protocol.OFPort;
import org.openflow.protocol.OFType;
import org.openflow.protocol.action.OFAction;
import
org.openflow.protocol.action.OFActionDataLayerDestination;
import org.openflow.protocol.action.OFActionDataLayerSource;
import
org.openflow.protocol.action.OFActionNetworkLayerDestination;
import org.openflow.protocol.action.OFActionNetworkLayerSource;
import org.openflow.protocol.action.OFActionOutput;
import org.openflow.util.U16;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import net.floodlightcontroller.core.FloodlightContext;
import net.floodlightcontroller.core.IFloodlightProviderService;
import net.floodlightcontroller.core.IOFMessageListener;
import net.floodlightcontroller.core.IOFSwitch;
import
net.floodlightcontroller.core.module.FloodlightModuleContext;
import
net.floodlightcontroller.core.module.FloodlightModuleException;
import net.floodlightcontroller.core.module.IFloodlightModule;
import net.floodlightcontroller.core.module.IFloodlightService;
import net.floodlightcontroller.packet.ARP;
import net.floodlightcontroller.packet.Ethernet;
import net.floodlightcontroller.packet.IPacket;
import net.floodlightcontroller.packet.IPv4;

public class LoadBalancer implements IOFMessageListener,
IFloodlightModule {

      protected IFloodlightProviderService floodlightProvider;
      protected static Logger logger;

      // Asignacion de una varia para la direccion IP y dirección
      // MAC para el balanceador de carga
      private final static int LOAD_BALANCER_IP =
      IPv4.toIPv4Address("192.168.111.254");
      private final static byte[] LOAD_BALANCER_MAC =
      Ethernet.toMACAddress("00:00:00:00:00:FE");
```

```
// Establecimiento de los tiempo de expiración de las
// reglas de flujo
private final static short IDLE_TIMEOUT = 60; // segundos
private final static short HARD_TIMEOUT = 0; // permanente

private static class Server
{
      private int ip;
      private byte[] mac;
      private short port;

      public Server(String ip, String mac, short port) {
            this.ip = IPv4.toIPv4Address(ip);
            this.mac = Ethernet.toMACAddress(mac);
            this.port = port;
      }

      public int getIP() {
            return this.ip;
      }

      public byte[] getMAC() {
            return this.mac;
      }

      public short getPort() {
            return this.port;
      }
}

final static Server[] SERVERS = {
new Server("192.168.111.13", "00:1c:c0:de:08:31",(short)1),
new Server("192.168.111.14", "00:1c:c0:de:0a:e1",(short)2)
};
private int lastServer = 0;

@Override
public String getName() {
      return LoadBalancer.class.getSimpleName();
}

@Override
public boolean isCallbackOrderingPrereq(OFType type, String
name) {
      // Auto-generated method stub
      return false;
}

@Override
public boolean isCallbackOrderingPostreq(OFType type,
String name) {
      // Auto-generated method stub
      return false;
}

@Override
public Collection<Class<? extends IFloodlightService>>
getModuleServices() {
      // Auto-generated method stub
```

```
      return null;
}

@Override
public Map<Class<? extends IFloodlightService>,
IFloodlightService> getServiceImpls() {
      // Auto-generated method stub
      return null;
}


@Override
public Collection<Class<? extends IFloodlightService>>
getModuleDependencies() {
Collection<Class<? extends IFloodlightService >>
floodlightService =
new ArrayList<Class<? extends IFloodlightService>>();

floodlightService.add(IFloodlightProviderService.class);
return floodlightService;
}


@Override
public void init(FloodlightModuleContext context)throws
FloodlightModuleException {
      floodlightProvider =
      context.getServiceImpl(IFloodlightProviderService.cla
      ss);
      logger = LoggerFactory.getLogger(LoadBalancer.class);
}


@Override
public void startUp(FloodlightModuleContext context) {
floodlightProvider.addOFMessageListener(OFType.PACKET_IN,
this);
}


@Override
public net.floodlightcontroller.core.IListener.Command
receive(IOFSwitch sw, OFMessage msg, FloodlightContext
cntx) {

      if (msg.getType() != OFType.PACKET_IN) {
          return Command.CONTINUE;
      }
            OFPacketIn pi = (OFPacketIn)msg;
            OFMatch match = new OFMatch();
            match.loadFromPacket(pi.getPacketData(),
            pi.getInPort());

      if (match.getDataLayerType() != Ethernet.TYPE_IPv4 &&
      match.getDataLayerType() != Ethernet.TYPE_ARP) {
      return Command.CONTINUE;
      }
```

```
        if (match.getNetworkDestination() !=
        LOAD_BALANCER_IP) {
        return Command.CONTINUE;
        }

        if (match.getDataLayerType() == Ethernet.TYPE_ARP) {
        logger.info("Solicitud ARP para el balanceador de
        carga recibida ");
              handleARPRequest(sw, pi, cntx);

        } else {

        logger.info("Paquete IPv4 distinado al balanceador de
        carga recibido");
        loadBalanceFlow(sw, pi, cntx);
        }

        return Command.STOP;
        }


private void pushPacket(IOFSwitch sw, OFPacketIn pi,
ArrayList<OFAction> actions, short actionsLength) {

        OFPacketOut po = (OFPacketOut)
        floodlightProvider.getOFMessageFactory().getMessage(O
        FType.PACKET_OUT);

        po.setInPort(pi.getInPort());
        po.setBufferId(pi.getBufferId());

        po.setActions(actions);
        po.setActionsLength(actionsLength);

    if (pi.getBufferId() == OFPacketOut.BUFFER_ID_NONE) {
        byte[] packetData = pi.getPacketData();
        po.setLength(U16.t(OFPacketOut.MINIMUM_LENGTH
        + po.getActionsLength() + packetData.length));
        po.setPacketData(packetData);
    } else {

        po.setLength(U16.t(OFPacketOut.MINIMUM_LENGTH
        + po.getActionsLength()));
    }

    try {
        sw.write(po, null);
    } catch (IOException e) {
        logger.error("error al escribir el mensaje packetOut:
        ", e);
    }
}

private void handleARPRequest(IOFSwitch sw, OFPacketIn pi,
FloodlightContext cntx) {

        logger.debug("Handle ARP request");
```

```
        Ethernet eth =
        IFloodlightProviderService.bcStore.get(cntx,IFloodlig
        htProviderService.CONTEXT_PI_PAYLOAD);
        System.out.println(eth);

        if (! (eth.getPayload() instanceof ARP))
        return;
        ARP arpRequest = (ARP) eth.getPayload();
        IPacket arpReply = new Ethernet()
        .setSourceMACAddress(LoadBalancer.LOAD_BALANCER_MAC)
        .setDestinationMACAddress(eth.getSourceMACAddress())
              .setEtherType(Ethernet.TYPE_ARP)
              .setPriorityCode(eth.getPriorityCode())
              .setPayload(new ARP()
                    .setHardwareType(ARP.HW_TYPE_ETHERNET)
                    .setProtocolType(ARP.PROTO_TYPE_IP)
                    .setHardwareAddressLength((byte) 6)
                    .setProtocolAddressLength((byte) 4)
                    .setOpCode(ARP.OP_REPLY)
                    .setSenderHardwareAddress(LoadBalancer.LO
                    AD_BALANCER_MAC)
                    .setSenderProtocolAddress(LoadBalancer.LO
                    AD_BALANCER_IP)
                    .setTargetHardwareAddress(arpRequest.getS
                    enderHardwareAddress())
              .setTargetProtocolAddress(arpRequest.getSenderP
              rotocolAddress()));

sendARPReply(arpReply, sw, OFPort.OFPP_NONE.getValue(),
pi.getInPort());
}


private void sendARPReply(IPacket packet, IOFSwitch sw,
short inPort, short outPort) {

        OFPacketOut po = (OFPacketOut)
        floodlightProvider.getOFMessageFactory().getMessage(O
        FType.PACKET_OUT);
        po.setBufferId(OFPacketOut.BUFFER_ID_NONE);
        po.setInPort(inPort);

        List<OFAction> actions = new ArrayList<OFAction>();
        actions.add(new
        OFActionOutput(outPort,(short)0xffff));
        po.setActions(actions);

        po.setActionsLength((short)OFActionOutput.MINIMUM_LEN
        GTH);

        byte[] packetData = packet.serialize();
        po.setPacketData(packetData);
        po.setLength((short) (OFPacketOut.MINIMUM_LENGTH +
        po.getActionsLength() + packetData.length));

        try {
              sw.write(po, null);
              sw.flush();
        } catch (IOException e) {
```

```
                logger.error("Error al escribir el mensaje
                packet out", e);
        }
}


private void loadBalanceFlow(IOFSwitch sw, OFPacketIn pi,
FloodlightContext cntx) {

        Server server = getNextServer();
        Ethernet eth =
        IFloodlightProviderService.bcStore.get(cntx,

        IFloodlightProviderService.CONTEXT_PI_PAYLOAD);

        OFFlowMod rule = new OFFlowMod();
        rule.setType(OFType.FLOW_MOD);
        rule.setCommand(OFFlowMod.OFPFC_ADD);
        OFMatch match = new OFMatch()
                .setDataLayerDestination(LOAD_BALANCER_MAC)
                .setDataLayerSource(eth.getSourceMACAddress())
                .setDataLayerType(Ethernet.TYPE_IPv4)
                .setNetworkDestination(LOAD_BALANCER_IP)
                .setNetworkSource(((IPv4)
                eth.getPayload()).getSourceAddress())
                .setInputPort(pi.getInPort());

        match.setWildcards(OFMatch.OFPFW_NW_PROTO);
        rule.setMatch(match);

        rule.setIdleTimeout(IDLE_TIMEOUT);
        rule.setHardTimeout(HARD_TIMEOUT);

        rule.setBufferId(OFPacketOut.BUFFER_ID_NONE);

        ArrayList<OFAction> actions = new
        ArrayList<OFAction>();

        OFAction rewriteMAC = new
        OFActionDataLayerDestination(server.getMAC());
        actions.add(rewriteMAC);

        OFAction rewriteIP = new
        OFActionNetworkLayerDestination(server.getIP());
        actions.add(rewriteIP);


        OFAction outputTo = new
        OFActionOutput(server.getPort());
        actions.add(outputTo);

        rule.setActions(actions);
        short actionsLength =
        short)(OFActionDataLayerDestination.MINIMUM_LENGTH
        + OFActionNetworkLayerDestination.MINIMUM_LENGTH
        + OFActionOutput.MINIMUM_LENGTH);
        rule.setLength((short) (OFFlowMod.MINIMUM_LENGTH +
        actionsLength));
```

```
logger.debug("Longuitude de las acciones="+
(rule.getLength() - OFFlowMod.MINIMUM_LENGTH));

logger.debug("Instalar la regla para la dirección de
avance del flujo " + rule);

try {
      sw.write(rule, null);
} catch (Exception e) {
      e.printStackTrace();
}

      OFFlowMod reverseRule = new OFFlowMod();
      reverseRule.setType(OFType.FLOW_MOD);
      reverseRule.setCommand(OFFlowMod.OFPFC_ADD);

      OFMatch reverseMatch = new OFMatch()
      .setDataLayerSource(server.getMAC())
      .setDataLayerDestination(match.getDataLayerSour
      ce())
      .setDataLayerType(Ethernet.TYPE_IPv4)
      .setNetworkSource(server.getIP())
      .setNetworkDestination(match.getNetworkSource()
      )
      .setInputPort(server.getPort());

reverseMatch.setWildcards(OFMatch.OFPFW_NW_PROTO);
reverseRule.setMatch(reverseMatch);

reverseRule.setIdleTimeout(IDLE_TIMEOUT);
reverseRule.setHardTimeout(HARD_TIMEOUT);

reverseRule.setBufferId(OFPacketOut.BUFFER_ID_NONE);

ArrayList<OFAction> reverseActions = new
ArrayList<OFAction>();

OFAction reverseRewriteMAC = new
OFActionDataLayerSource(LOAD_BALANCER_MAC);
reverseActions.add(reverseRewriteMAC);

OFAction reverseRewriteIP = new
OFActionNetworkLayerSource(LOAD_BALANCER_IP);
reverseActions.add(reverseRewriteIP);

OFAction reverseOutputTo = new
OFActionOutput(pi.getInPort());
reverseActions.add(reverseOutputTo);
reverseRule.setActions(reverseActions);
reverseRule.setLength((short)
(OFFlowMod.MINIMUM_LENGTH +
OFActionDataLayerSource.MINIMUM_LENGTH +
OFActionNetworkLayerSource.MINIMUM_LENGTH +
OFActionOutput.MINIMUM_LENGTH));

logger.debug("Instalar la regla para la dirección
contraria del flujo " + reverseRule);

try {
```

```
                sw.write(reverseRule, null);
            } catch (Exception e) {
                e.printStackTrace();
            }

            pushPacket(sw, pi, actions, actionsLength);
        }


        private Server getNextServer() {
            lastServer = (lastServer + 1) % SERVERS.length;
            return SERVERS[lastServer];
        }

}
```