UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO ELECTRÓNICO

PROYECTO TÉCNICO:

“DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE”

AUTORES:

CINTHYA GEMA ALAVA RIVAS

DANIEL IVÁN PALADINES MONTIEL

DIRECTOR:

ING. PABLO FABIÁN ECHEVERRÍA ÁVILA, MSC.

GUAYAQUIL – ECUADOR

2020

SALESIAN POLYTECHNIC UNIVERSITY

GUAYAQUIL HEADQUARTERS

ELECTRONIC ENGINEERING CAREERS

THECNICAL PROJECT PRIOR TO OBTAINING THE TITLE:

ELECTRONIC ENGINEERING

THEME:

“DESIGN AND IMPLEMENTATION OF A SOFTWARE DEFINED NETWORK (SDN) DIDACTIC MODULE FOR UNIVERSITY PRACTICES WITH OPENFLOW PROTOCOL THROUGH OPEN-SOURCE HARDWARE”

AUTORS:

CINTHYA GEMA ALAVA RIVAS

DANIEL IVÁN PALADINES MONTIEL

TUTOR:

ING. PABLO FABIÁN ECHEVERRÍA ÁVILA, MSC.

GUAYAQUIL – ECUADOR

2020

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo **ALAVA RIVAS CINTHYA GEMA** con documento de identificación # 0921895843, concedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud que soy autora del proyecto titulado “DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE.”, mismo que he desarrollado para obtener el título de Ingeniería Electrónica, quedando la Universidad autorizada para ejercer completamente los derechos concedidos anteriormente.

Guayaquil, Septiembre del 2020

Alava Rivas Cinthya Gema

C.I. 0921895843

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo **PALADINES MONTIEL DANIEL IVÁN** con documento de identificación # 0952545705, concedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud que soy autora del proyecto titulado “DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE.”, mismo que he desarrollado para obtener el título de Ingeniería Electrónica, quedando la Universidad autorizada para ejercer completamente los derechos concedidos anteriormente.

Guayaquil, Septiembre del 2020

Paladines Montiel Daniel Iván

C.I. 0952545705

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR

Por medio de la presente, Alava Rivas Cinthya Gema y Paladines Montiel Daniel Iván, declaramos que el proyecto técnico de titulación: “DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE.”, que se presentó ante el Consejo de la Carrera de Ingeniería Electrónica, es cedido en su totalidad, los derechos de propiedad intelectual que nos corresponden de este proyecto técnico, enunciado forma gratuita a favor de la Universidad Politécnica Salesiana.

Guayaquil, Septiembre del 2020

Alava Rivas Cinthya Gema	Paladines Montiel Daniel Iván
C.I. 0921895843	C.I. 0952545705

CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN

Yo declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación **"DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE."** con resolución de aprobación de Consejo de Carrera N° RESOLUCION 1061 realizado por los estudiantes **ALAVA RIVAS CINTHYA GEMA** con cédula de identidad N° 0921895843, **PALADINES MONTIEL DANIEL IVÁN** con cédula de identidad N° 0952545705, obteniendo un producto que cumple con los objetivos del diseño de aprobación, informe final y demás requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Guayaquil, Septiembre 18 del 2020

Ing. Pablo Fabián Echeverría Ávila, MSc.
Docente
C.I. 0916893357

DEDICATORIA

Dedico este proyecto de titulación a Dios por permitirme llegar a este momento de mi formación profesional.

Pero sobre todo dedico este proyecto de manera especial a mis maravillosos padres: Alvaro Alava Medina y Cintia Rivas Sellán, las personas que me han llenado de amor y fortaleza pero que también me han ensañado valores y disciplina. De ustedes aprendí que no hay mejor herencia que la educación. Les dedico este trabajo por todo su esfuerzo, sacrificio y dedicación como padres.

A las personas especiales que me han acompañado a lo largo de esta etapa. A mis hermanos: Fernanda, Sophia y Alvaro Alava por inspirarme e impulsarme a seguir mis metas, a mis amigos y amigas quienes fueron un gran apoyo emocional en el transcurso de este trabajo y a mi pareja Carlos Ordóñez que se preocupó por mí en cada momento y siempre fue mi ayuda idónea.

Alava Rivas Cinthya Gema

DEDICATORIA

Quiero dedicar este proyecto de titulación primero a Dios porque Él es sobre todas las cosas, por haberme permitido llegar hasta donde estoy en este momento, por mantener vivos a mis padres para que vean esta etapa de mi vida, a mis padres Paladines Jiménez Alejandro Iván y Montiel Calderón Esther Aracelly por incentivarme constantemente para salir adelante y tener un mejor futuro.

A mis hermanos por estar ahí como guías mayores que son y seguirán siendo un gran apoyo para mi vida, a mis amigos, a mis compañeros y a todos los que estuvieron para mí en todo momento.

Paladines Montiel Daniel Iván

AGRADECIMIENTO

Agradezco a Dios por guiarme y bendecirme a lo largo de mi existencia, por hacerme fuerte en los momentos difíciles y por permitirme vivir sonriendo cada día, a mi papá Alvaro Alava, le doy gracias por siempre ser mi guía y apoyo, por darme los mejores y más sabios consejos en el momento exacto, a mi mamá Cintia Rivas, gracias por su paciencia y entera confianza, por siempre estar en cada momento y levantar mis ánimos cuando es difícil.

Agradezco también a mi alma mater la Universidad Politécnica Salesiana por recibirme en sus aulas para formarme como una profesional y a todos los que forman parte de la carrera de Ingeniería Electrónica, sitio que llegó a convertirse en mi segundo hogar, a mi tutor el Ing. Pablo Echeverría por su trabajo, paciencia y su excelente dirección en el desarrollo de este proyecto de titulación y a todos los docentes que han contribuido en mi enseñanza y amor por la carrera en el transcurso de esta en especial a: PhD. Francisco Novillo, Ing. Mónica Miranda, Ing. Víctor Larco, Ing. Diego Freire, Ing. Orlando Barcia, Ing. Luis Córdova.

Alava Rivas Cinthya Gema

AGRADECIMIENTO

Quiero agradecer a Dios por siempre guiarme, por darme sabiduría para seguir adelante profesional y estudiantilmente, por darme el perdón y la oportunidad de seguir viviendo cada día y siempre perdonando mis errores y pecados, a mis padres: Paladines Jiménez Alejandro Iván y Montiel Calderón Esther Aracelly por darme desde que era pequeño una gran educación y su apoyo que no se puede comparar.

También quiero agradecer a Universidad Politécnica Salesiana por abrirme sus puertas a la carrera de Ingeniería Electrónica y poder ser un profesional salido de ella, a cada docente que estuvo en cada materia otorgándome sus conocimientos para que, a todos en general debido a que todos ellos han aportado mucho en mi vida estudiantil y profesional.

Paladines Montiel Daniel Iván

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Glosario

Calidad de Servicio: (QoS) es un conjunto de necesidades de servicio que la red debe desempeñar para afirmar un nivel de servicio conveniente para la transmisión de datos.

Clase de Servicio: También conocido como CoS, es un conjunto de propiedades que narra las funciones y las particularidades de los recursos de almacenamiento [1].

OpenFlow: Es el protocolo más utilizado en la interfaz SouthBound. Posee grandes ventajas ya que cuenta con un plano de reenvío flexible y configurable. [2].

SDN: Busca perfeccionar los recursos de la red y acoplar rápidamente las redes a las alterables necesidades empresariales, aplicaciones y tráfico [3].

***Router*:** Dispositivo de red encargado de llevar por la ruta adecuada el tráfico [4].

Switch: Dispositivo que sirve para conectar varios elementos dentro de una red [5].

WiFi: Es el vínculo de unión entre una red de datos fija y una cadena de dispositivos que trabajan de modo inalámbrico [6].

TCP/IP: Sistema de protocolos que hacen posibles servicios entre ordenadores que no pertenecen a la misma red [7].

IPv4: IPv4 es la cuarta versión del protocolo de internet que fue adaptado y ahora se utiliza ampliamente en la comunicación de datos a través de diferentes tipos de redes [8].

IPv6: Nueva versión del protocolo de internet está destinada a sustituir al estándar IPv4, la misma cuenta con un límite de direcciones de red, lo cual impide el crecimiento de la red [9].

Zodiac FX: Conmutador compatible con OpenFlow más pequeño y asequible del mundo [10].

Ryu: Controlador de red abierto (SDN) definido por software diseñado para aumentar la agilidad de la red al facilitar la administración y adaptación de la forma en que se maneja el tráfico [11].

OpenDaylight*:* Proyecto de código abierto (Open Source) el cual tiene como objetivo acelerar y acrecentar la difusión de la innovación en el diseño e implementación de un estándar abierto y transparente de redes definidas por software, es decir, Software Defined Networking (SDN) [12]**.**

Faucet: Controlador OpenFlow compacto de código abierto, que permite a los operadores de red ejecutar sus redes de la misma manera que lo hacen en clústeres de servidores [13].

Mininet: Simulador de red que puede crear redes de hosts, *switches* controladores y links virtuales [14].

Floodlight: Controlador OpenFlow de clase empresarial con licencia de Apache y basado en Java [15].

Netconf: Protocolo *IETF* pensado como una evolución de *SNMP*. Usa *SSH*, *SOAP*, o *TLS* como transporte. En este curso utilizaremos únicamente SSH [16].

SoftNet: Red similar a la *WSN*.

Soft switch: Engloba los procesos y elementos informáticos que controlan las sesiones, el medio (voz, video o mensajes) y los servicios [17].

Open vSwitch: Plataforma de conmutador virtual de código abierto [18].

OpenStack: Plataforma *cloud computing de* software libre [19].

Cloud computing: Es un grupo de principios y enfoques que permite suministrar infraestructura informática, servicios, plataformas y aplicaciones (provenientes de la nube) a los usuarios, según las requieran y por medio de una red [20].

AutoCAD: Programa de dibujo por computadora CAD para diseños de 2 y 3 dimensiones.

Solidworks: Software tipo CAD que permite la creación de una forma intuitiva y rápida de modelos sólidos en 3D.

VirtualBox: Programa de gran utilidad para todos aquellos que necesiten utilizar un sistema operativo puntualmente pero no quieran crear una partición en su equipo y por consiguiente tener que instalar dos sistemas operativos en el mismo ordenador [21].

WinSCP: Es un cliente SFTP gráfico el cual usa SSH.

Wireshark: Herramienta gráfica empleada para el análisis de redes.

Grafana: Herramienta hecha en software libre, específicamente con licencia Apache 2.0, ideada por Torkel Ödegaard. [22]

Gauge: Marco de automatización de pruebas gratuito y de código abierto. [23]

Prometheus: Es un sistema de series de tiempo y monitoreo de código abierto de próxima generación que recopila métricas de los agentes que se ejecutan en los hosts de destino. [24]

Keystone: Servicio de OpenStack que proporciona autenticación de clientes API. [25]

Flow Maker: Aplicación perteneciente a Northbound Networks que permite crear y eliminar flujos desde la interfaz del controlador HP VAN SDN.

OpenFlow Manager: Aplicación desarrollada para ejecutarse sobre ODL (OpenDayLight) para representar topologías de OpenFlow (OF), programar rutas OF y recopilar estadísticas OF. [26]

Términos utilizados

USB: Universal Serial Bus. Es un puerto que permite la conexión de diferentes periféricos.

SNMP: Simple Network Management Protocol o protocolo simple de administración de red [27].

IETF: Internet Engineering Task Force.

SSH: Secure Shell.

SOAP: Formato de mensaje *XML* utilizado en interacciones de servicios web.

XML: Es un subconjunto de *SGML* (Estándar Generalised Mark-up Language), simplificado y adaptado a Internet.

SGML: Standard Generalized Markup Language.

TLS: Transport Layer Security o seguridad de la capa de transporte.

SSL: Es el acrónimo de Secure Sockets Layer o capa de sockets seguros.

WSN: Wireless Sensor Network o redes de sensores inalámbricos.

NAT: Network Address Translation o traducción de direcciones de red.

VLAN: Virtual Local Area Network o red de área local virtual.

GRE: Generic Routing Encapsulation o encapsulación de enrutamiento genérico.

API: Application Programming Interface o interfaz de programación de aplicaciones.

MPLS: Multiprotocol Label Switching o conmutación de etiquetas multiprotocolo***.***

OSPF: Open Shortest Path First o protocolo abierto de los enlaces.

IS-IS: Intermidiate System to Intermidiate System o sistema intermedio a sistema intermedio.

BGP – LS: Border Gateway Protocol – Link State o puerta de enlace de frontera – estado de enlace.

PCEP: Path Computation Element Communication Protocol o protocolo de elemento de cálculo de ruta.

LAN: Local Area Network o red de área local.

WLAN: Wireless Local Area Network o red de área local inalámbrica.

UTP: Unshielded Twisted Pair o cable de par trenzado sin blindaje.

STP: Shielded Twisted Pait o cable de par trenzado blindado.

RSTP: Rapid Spanning Tree Protocol o protocolo de árbol de expansión rápida.

SCTP: Stream Control Transmission Protocol o protocolo de transmisión de control de corriente.

REST: Transferencia de estado representacional.

JSON: JavaScript Object Notation o notación de objetos de JavaScript.

ONF: Open Networking Foundation.

OVA: Objeto virtual de aprendizaje.

OFM: OpenFlow Manager.

RAM: Random Access Memory.

SFTP: SSH File Transfer Protocol.

PC: Personal Computer o computadora personal.

IDS: Sistema de detección de intrusos.

JDK: Java Development Kit.

ID: Identification o identificación.

Resumen

En este trabajo se demuestra el diseño e implementación de un módulo sobre redes definidas por software (SDN) el cual funcionará con el protocolo OpenFlow mediante el uso de hardware libre para realizar prácticas en la Universidad Politécnica Salesiana, sede Guayaquil.

Uno de los más grandes beneficios de implementar SDN es su flexibilidad, ya que el controlador SDN toma las funciones más complejas como por ejemplo monitorear en tiempo real el comportamiento de la red o manejar la inteligencia de la red, los dispositivos de red, entre otros. Esto elimina la necesidad de que los dispositivos de red entiendan como ejecutar el flujo de distribución de datos basados en diversos protocolos de comunicación de diversos proveedores, y como resultado termina otorgando a los administradores diversos beneficios como lo son una gran flexibilidad al configurar, administrar, asegurar, y optimizar los dispositivos de red. Como resultado de esto es mucho más fácil variar o modificar la aplicación ya que anteriormente se tenía reconfigurar cada dispositivo de red manualmente por separado.

La principal causa de la implementación de una red definida por software es el creciente proceso de la era digital donde el software se ha convertido en un elemento tan importante como el hardware, teniendo en cuenta que las organizaciones están en constante aumento en el ancho de banda de su red.

El nivel de exigencia cuando nos referimos a redes ha incrementado mucho y también la necesidad de poseer una infraestructura que pueda soportar requerimientos

de alto nivel en un futuro tal como internet de las cosas, realidad virtual, inteligencia artificial entre otros.

Empresas y data centers han empezado a implementar el uso de SDN en sus redes corporativas debido a que el uso de esta tecnología permite tener una red automatizada que ofrece eficiencia, seguridad y ahorro de costos operativos.

Está más que claro que las empresas son cada vez más dependientes de la tecnología y las redes actuales ya no pueden permitirse tener tiempos de ociosidad. Un fallo en un sistema informático ocasiona contrariedades considerables, como lo son las pérdidas, o como mínimo, la insatisfacción de las personas que se benefician o están ligados a un servicio que usa este tipo de tecnología o redes.

Abstract

This article evidences the design and implementation of a module on Software Defined Networks (SDN), which will work with the OpenFlow protocol using open-source hardware to carry out practices at the “Universidad Politécnica Salesiana”, Guayaquil.

One of the greatest benefits of implementing SDN is its flexibility, since the SDN controller takes the most complex functions such as monitoring in real time the behavior of the network or managing the intelligence of the network, network devices, among others. This eliminates the need for network devices to understand how to execute the data distribution flow based on various communication protocols from different providers, and as a result end up giving administrators various benefits such as great flexibility in configuring, managing, secure, and optimize network devices.

As a result of this, it is much easier to vary or modify the application since previously; the administrator had to manually reconfigure each network device separately.

The main cause of the implementation of a software-defined network is the growing process of the digital era where software has become as important an element as hardware, taking into account that organizations are constantly increasing the bandwidth of their network.

The level of demand when we refer to networks has increased a lot and the need to have an infrastructure that can support high-level requirements in the future such as the internet of things, virtual reality, and artificial intelligence among others.

Companies and data centers have begun to implement the use of SDN in their corporate networks because the use of this technology allows having an automated network that offers efficiency, security and savings in operating costs.

It is more than clear that companies are increasingly dependent on technology and current networks can no longer afford idle times. A failure in a computer system causes considerable setbacks, such as losses, or at least, the dissatisfaction of people who benefit or are linked to a service that uses this type of technology or networks.

Introducción

El presente proyecto de titulación “DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DIDÁCTICO DE RED DEFINIDA POR SOFTWARE (SDN) PARA PRÁCTICAS UNIVERSITARIAS CON PROTOCOLO OPENFLOW MEDIANTE HARDWARE LIBRE”, está basado en un conjunto de *Raspberries* que se usan como servidores y hosts que están interconectados por medio del mini *switch Zodiac FX* para así desarrollar las prácticas que pueden hacer que la red sea configurable.

El objetivo principal es desarrollar un módulo SDN para que sea utilizado en el laboratorio de telecomunicaciones, ya que este tema es muy actual en el ámbito tecnológico y de las comunicaciones, por lo tanto, sería satisfactorio implementar las prácticas en seminarios profesionales y materias como Redes I, Redes II, entre otras.

Se realizan diferentes prácticas en las que una Raspberry tiene instalado un controlador que actúa como servidor entre las *Raspberries* que actúan como Host para ejecutar diferentes servicios y así definir cuál controlador es mucho más versátil que otro.

En el capítulo 2 se muestran los conceptos utilizados en base a la SDN, como también la historia y evolución de ésta. En el capítulo 3 se visualiza la fabricación y construcción del módulo y del Stack que será usado para las prácticas con los diferentes controladores, denotando y acentuando una serie de diferencias entre cada uno de ellos y así seleccionar el controlador más estable.

1 Problema

Al momento en el laboratorio de Telecomunicaciones de la Universidad Politécnica Salesiana, sede Guayaquil, no existe un módulo para realizar prácticas enfocadas a las redes definidas por software, por lo tanto, se considera la creación y desarrollo de éste para que los estudiantes puedan profundizar sus conocimientos en las materias afines.

1.1 Antecedentes

Teniendo en cuenta la problemática, se tiene la necesidad de diseñar e implementar el módulo didáctico que es de gran utilidad para los estudiantes y docentes de la carrera de ingeniería electrónica y de telecomunicaciones de la Universidad Politécnica Salesiana sede Guayaquil.

Se debe indicar que el ahorro de costos operativos es un gran beneficio para las empresas que implementen *SDN* ya que se trabaja sobre aquella infraestructura. Sin embargo, el uso de estos dispositivos de red de última generación involucra altos costos de inversión, razón por la cual implementar *SDN* en un ámbito estudiantil demanda mucha inversión para el centro educativo.

Encontrar una solución a este inconveniente fue nuestra mayor motivación para realizar este proyecto de titulación, que los estudiantes puedan comprender la separación del plano de control y el plano de datos al programar y utilizar un controlador

SDN centralizado para que puedan desarrollar sus conocimientos sobre protocolos de red que sean programables mediante el uso de hardware libre tales como *Raspberry Pi* y la integración de la tarjeta *Zodiac FX.*

El diseño e implementación de este módulo didáctico de red definida por software busca involucrar tanto a estudiantes como docentes en la adquisición de conocimientos teóricos y prácticos de las redes definidas por software.

1.2 Importancia y alcance

La importancia y el alcance del presente proyecto es el diseño e implementación de un módulo didáctico de red definida por software (*SDN*) aplicando el protocolo *OpenFlow* mediante hardware libre para la creación y ejecución de aplicaciones *SDN* para que sea posible cambiar el comportamiento de una red de manera dinámica.

El cambio en la tecnología es constante, motivo por el cual, debemos seguir fomentando e impulsando la innovación entre los estudiantes de la Universidad Politécnica Salesiana.

Además, se busca beneficiar a los estudiantes de ingeniería electrónica y telecomunicaciones que cursan las asignaturas de redes de computadoras I y II, entre otras, cuya finalidad es complementar y reforzar los conocimientos adquiridos

teóricamente en clases para así incentivar a realizar proyectos basados en redes definidas por software.

1.3 Delimitación

1.3.1 Delimitación temporal

El proyecto de titulación se diseña y desarrolla en el periodo lectivo 2019 - 2020.

1.3.2 Delimitación espacial

El proyecto se desarrolla en el laboratorio de telecomunicaciones de la carrera Ingeniería Electrónica en la Universidad Politécnica Salesiana, sede Guayaquil, que se encuentra ubicada en la Av. Domingo Comín y callejón Chambers.

1.3.3 Delimitación académica

A través del diseño y desarrollo de este proyecto de titulación se aplican conocimientos técnicos adquiridos durante los cursos regulares y seminarios profesionales dictados en el trascurso de la carrera en las materias como red de

computadoras I y II, redes de comunicaciones, taller de comunicaciones y seminarios como redes *LAN-WAN*, programación usando *Python*. También se utilizaron herramientas como software de simulación y adquisición de datos en la red.

1.4 Explicación del problema

¿Las *SDN* serán necesarias en el aprendizaje de los estudiantes de las carreras de telecomunicaciones o afines? Debido al auge que actualmente tienen las *SDN* en el campo de las telecomunicaciones, se considera que debería de haber módulos de éstas para el aprendizaje sustancial de las *SDN* para estar más allegados a estas.

1.5 Justificación del proyecto

Actualmente en el laboratorio de telecomunicaciones de la Universidad Politécnica Salesiana (UPS), sede de Guayaquil, se realizan prácticas orientadas a complementar los conocimientos teóricos, sin embargo, se deben incorporar nuevas prácticas de acuerdo con las necesidades del medio. Esto es de vital ayuda para los estudiantes, ya que con esto se puede solidificar, perfeccionar y tener más eficacia en la formación de futuros Ingenieros familiarizándolos con el tema de las telecomunicaciones.

1.6 Objetivos

1.6.1 Objetivo general

Diseñar un módulo didáctico de red definida por software (*SDN*) para prácticas universitarias en el laboratorio de telecomunicaciones utilizando protocolo *OpenFlow* mediante el uso de hardware libre.

1.6.2 Objetivos específicos

- Evaluar el comportamiento de la *SDN* al utilizar las alternativas de controladores como *Ryu*, *Floodlight*, *OpenDaylight* y *Faucet*.

- Utilizar el software *Mininet* como emulador de una red definida por software.

- Validar la funcionalidad y flexibilidad de la red *SDN* mediante la ejecución de prácticas.

- Elaborar manual de prácticas guiadas de la *SDN*, para ser utilizada por docentes y estudiantes del laboratorio de telecomunicaciones de la Universidad Politécnica Salesiana, Sede Guayaquil.

2 Marco teórico referencial

2.1 SDN [28] [29]

La red definida por software (*SDN*) es una pauta para la virtualización de red que tiene como objetivo perfeccionar los recursos de la red y ajustar rápidamente las redes a las variables necesidades empresariales, aplicaciones, tráfico, etc. Funciona alejando el plano de control de la red y el plano de datos, elaborando así una infraestructura programable por software que es diferente de los dispositivos físicos.

Cuando un paquete llega a un conmutador en una red convencional, las reglas predeterminadas al firmware propietario del conmutador le dicen al conmutador a dónde enviar el paquete. El conmutador envía cada paquete al mismo destino por la semejante trayectoria y trata a todos los paquetes de la misma manera.

Durante ya muchos años se ha innovado en tecnologías de redes informáticas que han dado soporte a la aparición y diseño de modernas aplicaciones que piden nuevas exigencias. Las redes actuales son cada vez más grandes y los servicios de comunicaciones que estas exhiben han tomado protagonismo alrededor del mundo, el envío de información generado por los usuarios se realiza a través de las incontables redes que existen y que crecen aceleradamente, debido a que actualmente son muchos los dispositivos aptos de conectarse a una red. Esto ha paralizado la gestión, administración y configuración en las redes tradicionales, las *SDN* ofrecen una nueva idea de la red donde el plano de control es independiente del plano de datos. Una de

las principales doctrinas de la *SDN* es suministrar una configuración más eficaz de modo que se pueda tener una red que presente un mejor rendimiento y flexibilidad para que ayude a solucionar los nuevos problemas y exigencias que solicitan las tecnologías hoy en día. Por eso, grupos de investigación junto con las grandes compañías del mercado tecnológico han perfeccionado la idea de renovar el diseño actual de las redes de datos, naciendo así el concepto de las redes definidas por software. En la **Fig. 1** se observa la principal diferencia entre una red tradicional y una red definida por software.

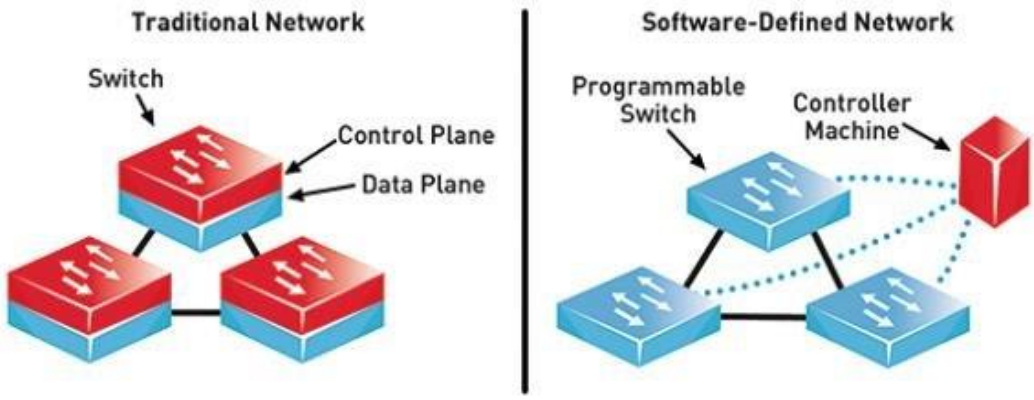

Fig. 1 **Red Tradicional vs. SDN [29]**

Después de que se retomara el concepto de *SDN*, se elaboraron e implementaron un conjunto de controladores personalizados; grandes empresas pertenecientes a la *ONF* (Open Networking Foundation) desarrollaron sus propios controladores y los expusieron al mercado global. Por otra parte, el público en general también ha desarrollado controladores de uso libre cuyo propósito es el mismo que los controladores propietarios. En la **TABLA I** se observan los diferentes tipos de

controladores SDN que existen en la actualidad junto con las versiones de *OpenFlow* que admiten:

TABLA I

Controladores SDN [30]

Nombre	Lenguaje	Versión de OpenFlow	Libre	Interfaz gráfica
Beacon	Java	1.0	Si	No
Floodlight	Java	1.0 – 1.4	Si	Si
HPE VAN SDN Controller	Java, Python	1.0 – 1.3	No	Si
OpenDaylight (ODL)	Java	1.0 – 1.3	Si	Si
Ryu SDN Framework	Python	1.3	Si	No
NOX	C++	1.0	Si	No
POX	Python	1.0	Si	No
Cisco Open SDN Controller	Java, Python	1.0 – 1.3	No	Si
Juniper Contrail Controller	Java, Python	1.0 – 1.3	No	Si

2.1.1 Nacimiento de las SDN [31]

El nacimiento de las *SDN* no tiene un punto de partida exacto en el tiempo, pero se podría tomar como referencia la aparición de *SOFTNET* en los años 80, ésta fue una red similar a las Wireless Sensor Networks (*WSN*) cuya innovación consistía en el campo de datos de los paquetes incluía comandos que eran ejecutados por los nodos a medida que estos los recibían. De esta forma, la red podía ser modificada en tiempo real y de forma dinámica. Este fue un intento de realizar una red que se auto organice,

no hubo desarrollo posterior, pero sirvió de base para el desarrollo de otros proyectos. Uno de los primeros proyectos más notables fue “GeoPlex” de la empresa AT&T, que consistió en un software que asistía a una aplicación para comunicarse con otras aplicaciones, redes, hardware y sistemas operativos.

Lo que AT&T quería era un “*soft switch*” que permitiera reconfigurar los *switches* dentro de la red para poder cargar nuevos servicios desde un sistema de soporte de operaciones.

Las “active networks” tenían una arquitectura que consistía en llevar en los paquetes, pequeños programas que podían ser ejecutados por los nodos que estos iban atravesando. Esto permitía que los elementos de red como *switches* y *routers*, procesen los paquetes de datos y no se limiten únicamente a reenviar los paquetes de un puerto a otro de forma pasiva.

La separación del plano de control del plano de datos, que empezó a desarrollarse a inicios del año 2001, surgió de la problemática del aumento del volumen de tráfico y la necesidad de redes de mayor confianza. Los *switches* y *routers* convencionales, tenían una estrecha relación entre el plano de datos y el plano de control, esto complicaba la tarea de configuración y control del comportamiento de enrutamiento.

A mediados del año 2007, varios grupos de investigación desarrollaron sistemas basados en este enfoque, lo que eventualmente desembocó en la aparición de *OpenFlow*. Gracias a la adopción de *OpenFlow* por parte de varias empresas, se permitió a los programadores controlar algunos de los parámetros de reenvío mediante

la apertura de sus *API*, la versión inicial de *OpenFlow* se estableció en los equipos a través de una simple actualización de firmware.

La virtualización de la red presenta la abstracción de una red que esta desacoplada del equipo físico. Esto permite tener múltiples redes virtuales ejecutándose en una misma infraestructura de forma compartida. Esta idea de virtualización de redes que viene desarrollándose desde mediados de los años 90, donde se tiene abstracción de una red física en términos de una red lógica, fue la base para que en las *SDN* se separe el plano de control con el plano de datos.

En el año 2011 se fundó la *"Open Networking Foundation"* (*ONF*) con el fin de promover y promocionar las *SDN* y OpenFlow. En la **Fig. 2** se muestra la cronología de desarrollo de la SDN.

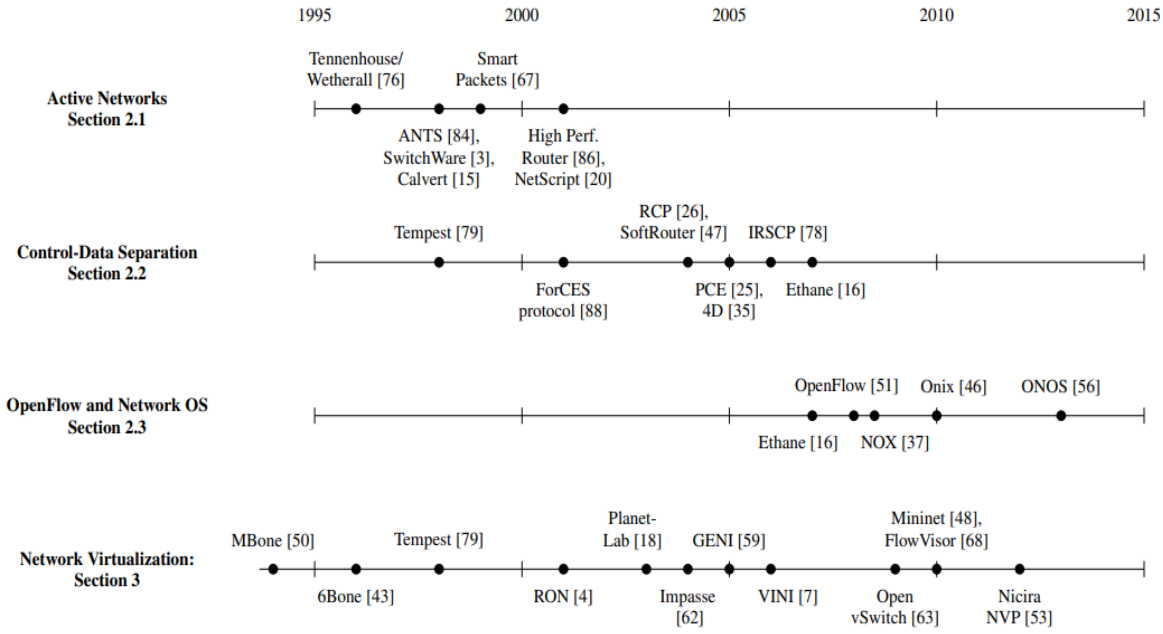

Fig. 2 **Cronología de desarrollo de las SDN [31]**

2.1.2 Definición de SDN [32]

Es una nueva arquitectura de red en la que se separa el plano de control del plano de datos para conseguir redes más fáciles de administrar, que sean flexibles y automatizables. En resumen, con las redes definidas por software se virtualiza la red, haciéndola independiente de la infraestructura física. La tecnología de las *SDN* desacopla el plano de datos (información de usuario y de aplicaciones que viaja en la red), del plano de control (información generada por los dispositivos de red que permite la comunicación de los mismos), de esta forma se logra disminuir la complejidad y costo de los equipos de red. La **Fig. 3** muestra un diagrama conceptual de la estructura de la *SDN*.

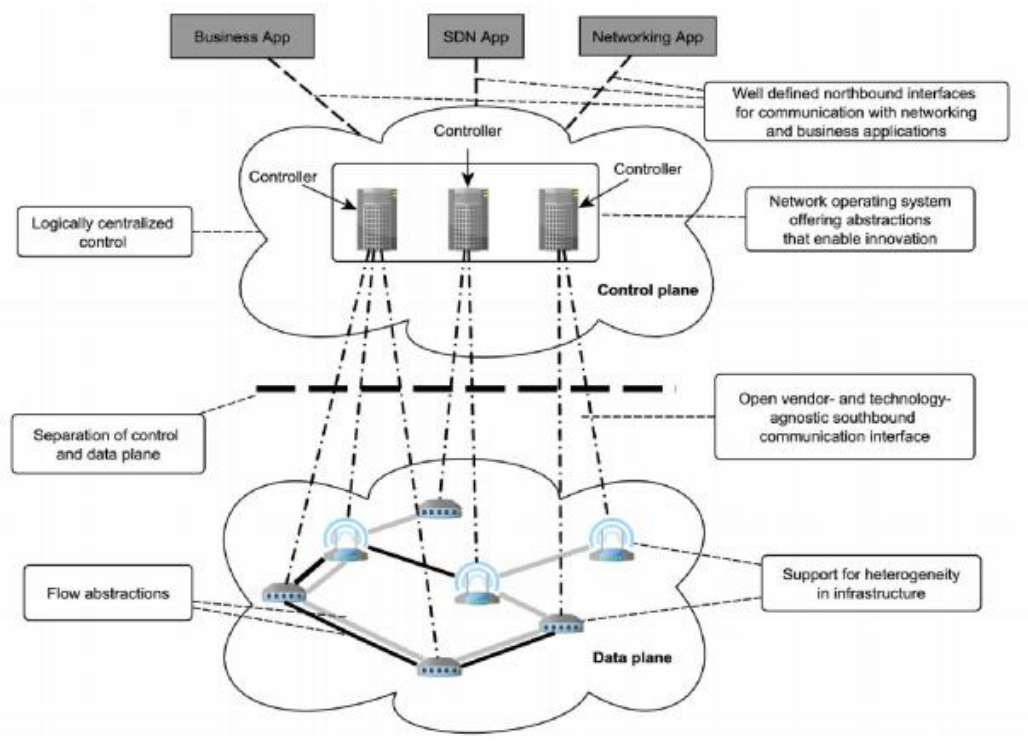

Fig. 3 **Estructura de la SDN [31]**

2.1.3 Ventajas [33] [34]

Las redes definidas por software se basan en la virtualización e introducen una capa inteligente de separación de la configuración de las conexiones físicas y el hardware. De esta manera, las *SDN* ofrecen control programático sobre dispositivos de red tanto físicos como virtuales que pueden responder dinámicamente a los cambios en las condiciones de la red utilizando *OpenFlow* o algún otro protocolo. Por lo tanto, en su forma más simple, la virtualización, permite que el software se pueda ejecutar de forma independiente del hardware.

La virtualización ha hecho posible la computación en la nube, la cual permite a los centros de datos proveer de forma dinámica los recuerdos de tecnologías de información exactamente donde se los necesita y sobre la marcha. Para mantenerse al día con la complejidad y la velocidad de este procesamiento, la red debe adaptarse siendo cada vez más sensible y capaz de responder automáticamente.

Las redes actuales tienen serias limitaciones y métodos antiguos que simplemente ya no funcionan. Como la virtualización, la nube y la movilidad crean entornos más complejos, las redes deben ser capaces de adaptarse en términos de seguridad, escalabilidad y capacidad de administración. Cuando un paquete llega a un *switch* en una red tradicional, las configuraciones realizadas al firmware propietario del *switch* le dicen a éste donde debe transferir el paquete. Existen *switches* inteligentes diseñados con circuitos integrados de aplicación específica que son lo suficientemente sofisticados para reconocer diferentes tipos de paquetes y dar a los mismos un tratamiento

específico según las configuraciones realizadas, pero estos *switches* pueden llegar a ser muy costosos dependiendo de las características de los mismos.

En una *SDN*, el administrador de la red puede configurar el comportamiento del tráfico desde una consola de control centralizada sin tener que realizar configuración alguna en los *switches* individuales. El administrador puede modificar cualquier regla en los *switches* en cualquier momento que se requiera.

El propósito de la arquitectura de una *SDN* es permitir a los ingenieros y administradores de red una respuesta rápida a los nuevos requerimientos de aplicaciones dentro de la red. Con las *SDN* se virtualiza la red haciéndola independiente de la infraestructura física y creando una red lógica con el fin de cumplir los requerimientos de rendimiento, escalabilidad y agilidad solicitados por la red. Algunas de las ventajas más notables que poseen las *SDN* son las siguientes:

- **Directamente programable:** El control de la red es directamente programable ya que se encuentra desacoplado de la función de envío de datos.

- **Administración centralizada:** La inteligencia de la red está basada en controladores realizados en software, que mantienen una vista global de toda la red y que aparecen para las aplicaciones como un solo *switch* central.

- **Programación configurada:** Las *SDN* permiten a los administradores de red configurar, administrar, asegurar y optimizar los recursos de red de forma rápida

y dinámica a través de programas automatizados que pueden ser escritos por los mismos administradores ya que estos no dependen de software propietario.

- **Basado en estándares abiertos y vendedores neutrales:** Debido a que se implementa mediante estándares abiertos, *SDN* simplifica el diseño que la red y su operación ya que las instrucciones están provistas por un controlados en lugar de protocolos específicos provistos por los vendedores.

- **Mejora de la seguridad:** La seguridad es ahora administrada por el controlador, lo que facilita el control de la seguridad en los equipos de red.

- **Facilidad de innovación:** Las redes *SDN* son más flexibles y configurables lo que facilita la experimentación con nuevas aplicaciones, configuraciones, etc.

2.1.4 Arquitectura de la SDN [35] [36] [37] [38]

De forma general se puede considerar la arquitectura *SDN* está compuesta de 3 capas que han sido especificadas por la *ONF*:

- **Capa de infraestructura:** Se encuentra formada por los dispositivos de red encargados de realizar la conmutación y en enrutamiento de los paquetes. Proporciona un acceso libre que se da a través del protocolo *OpenFlow*.

- **Capa de control:** La función de control de la red se encuentra centralizada y permite a los desarrolladores de aplicaciones utilizar recursos de red abstrayéndolos de la topología de la misma. Su principal componente es el controlador *SDN* encargado del control de la configuración de los nodos para la correcta distribución del tráfico. La arquitectura describe varias funciones internas al controlador y los elementos de red, pero solo específica el comportamiento de aquellos aspectos necesarios para la interoperabilidad.

- **Capa aplicación:** Consiste en las aplicaciones ejecutadas por los usuarios finales que utilizan la red a través de los servicios de *SDN* y que utilizan las *API* de la capa de control "dirección norte" (NorthBound), estas pueden ser *XML*, *REST*, *JSON*, etc. Permite a las aplicaciones y servicios simplificar y automatizar las tareas de provisión y configuración y administrar nuevos servicios en la red, ofreciendo tanto a proveedores de servicios como a clientes nuevas vías de ingreso, innovación y diferenciación.

En la **Fig. 4** se muestran las capas de la arquitectura de la SDN:

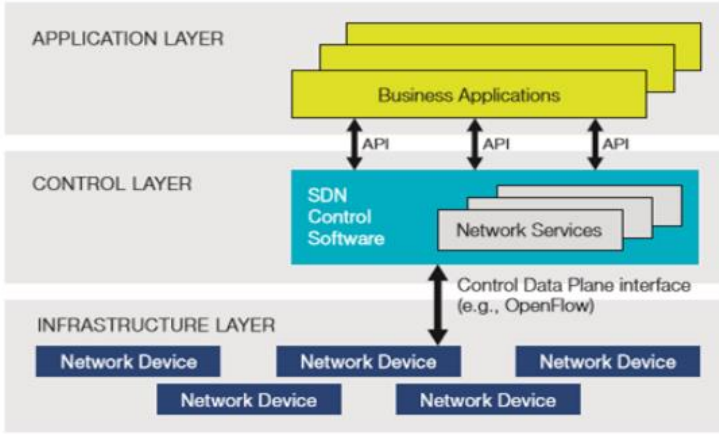

Fig. 4 **Capas de la arquitectura SDN [38]**

2.2 Mininet [39] [40] [41]

Se puede interactuar fácilmente con la red utilizando *Mininet CLI* y *API*, personalizarlo y compartirlo con otros o implementarlo en hardware real, *Mininet* es útil para el desarrollo, enseñanza e investigación. También es una excelente manera de desarrollar, compartir y experimentar con *OpenFlow* y los sistemas de redes definidas por software. *Mininet* está activamente desarrollado y admitido, y se publica bajo una licencia *BSD* de código abierto permisiva incluso se le recomienda a la comunidad que contribuya con código, informes/correcciones de errores, documentación y cualquier otra cosa que pueda mejorar el sistema.

Las *SDN* son un nuevo modelo en redes de datos que está siendo foco de artículo y exploración en los últimos años, debido a las disímiles ventajas que tiene con respecto al funcionamiento de las redes convencionales. Las redes definidas por software separan el plano de control de un dispositivo de red, del plano de datos, permitiendo controlar, monitorizar y tramitar una red desde un nodo centralizado o controlador, lo cual promete facilitar la gestión de red e incluir creación a través de su sistematización.

Teniendo en cuenta que las *SDN* son un tema de investigación actual y consecuentes de la importancia que tienen las herramientas que permiten suponer o emular diferentes dispositivos de red para probar las nuevas soluciones propuestas por los investigadores, este artículo presenta en primera instancia una conceptualización general de las *SDN*, luego describe las diferentes herramientas de simulación y emulación más utilizadas y finalmente se enfoca en el estudio de una herramienta de emulación denominada Mininet, mostrando modos de instalación, creación de

topologías de red, controladores y principales funcionalidades, revelando su versatilidad en el prototipado con este novedoso paradigma.

Mininet proporciona un banco de pruebas virtual y un entorno de desarrollo para redes definidas por software (*SDN*). Mininet permite el desarrollo de *SDN* en cualquier computadora portátil o *PC*, y los diseños de *SDN* pueden moverse sin problemas entre Mininet (lo que permite un desarrollo económico y optimizado) y el hardware real que se ejecuta al ritmo de la línea en implementaciones en vivo. *Mininet* habilita:

- Prototipado rápido de redes definidas por software.
- Pruebas de topología complejas sin la necesidad de conectar una red física.
- Múltiples desarrolladores concurrentes para trabajar independientemente en la misma topología.

Las redes de Mininet ejecutan código real, incluidas las aplicaciones de red estándar de Unix / Linux, así como el núcleo de Linux real y la pila de red. *Mininet* proporciona una *API* de *Python* extensible para la creación y experimentación de redes. Se lanzó bajo una licencia *BSD open source* permisiva y está activamente desarrollado y respaldado por la comunidad de entusiastas de redes y *SDN*.

2.3 Zodiac FX [10] [42]

La *Zodiac FX* es el conmutador o *switch* compatible con *OpenFlow* más pequeño y barato en el mundo. Al proporcionar muchas de las características de un conmutador de grado empresarial pero lo suficientemente pequeño como para caber en la palma de su mano, es perfecto para investigación y educación.

Imagine la posibilidad de tener un control completo sobre su red doméstica. Como dar prioridad a las aplicaciones sensibles al ancho de banda, como la transmisión de medios o juegos en línea. ¿Qué sucede con la supervisión del uso de la red o el acceso de invitados a amigos y familiares o incluso su uso para proteger su red de intrusos? Hasta ahora, el poder de las redes definidas por software (*SDN*) solo estaba disponible para los administradores de grandes redes corporativas como Google y Facebook. A pesar de que existen numerosos controladores *SDN* gratuitos o de código abierto en internet, lo único que faltaba era un conmutador *OpenFlow* pequeño y asequible.

La *Zodiac FX* proporciona muchas de las características de un conmutador *OpenFlow* que cuesta miles de dólares, pero es lo suficientemente pequeño como para caber en la palma de su mano. Algunas de esas características sorprendentes incluyen:

- 4 x 10/100 puertos fast ethernet con magnetismo integrado.
- Interfaz de línea de comandos accesible a través del puerto serie virtual *USB.*
- Procesador Atmel ATSAM4E Cortex M4

- Soporte para OpenFlow 1.0, 1.3 y 1.4
- Tabla de flujo de software de 512 entradas
- Búfer de cuadros de 64KB con almacenamiento y reenvío sin bloqueo
- Compatibilidad con *VLAN* 802.1q para 64 grupos desde 4096 ID
- Autenticación 802.1x por puerto
- Protocolo de árbol de expansión rápida 802.1w (RSTP)
- 16 ACL por puerto
- Soporte de marco jumbo de 2KB
- Priorización de *QoS* / *CoS* con inserción de etiqueta 802.1q
- Auto MDIX con detección de X-over
- Por enlace de puerto y LEDs de actividad.
- Cabezal de expansión SPI de alta velocidad
- Alimentado por *USB*
- Tamaño ultra pequeño de tan solo 10 cm x 8 cm.

En la **Fig. 5** se muestran los componentes que incluye la tarjeta *Zodiac FX*:

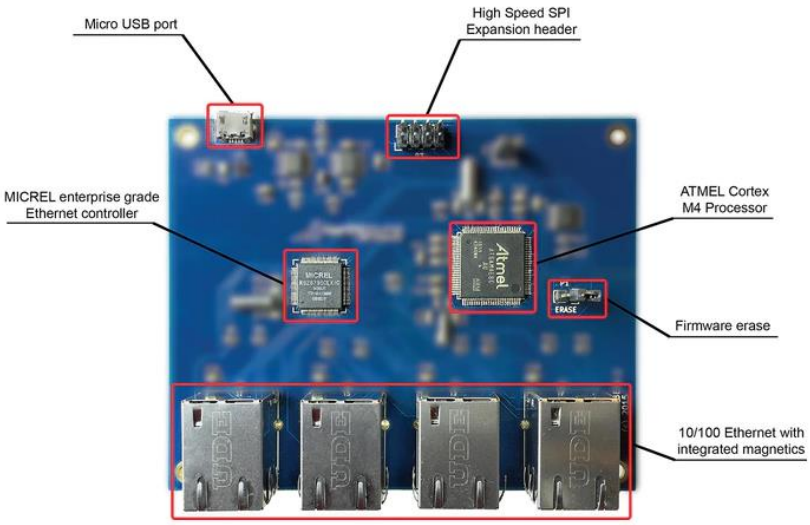

***Fig. 5* Zodiac FX [42]**

2.4 Floodlight [43]

Floodlight es un controlador *OpenFlow* de variedad empresarial con licencia de Apache y basado en Java. Es apoyado por una gran comunidad de desarrolladores que incluye varios ingenieros de big *switch* networks. Está diseñado para funcionar con el creciente número de conmutadores, enrutadores, conmutadores virtuales y puntos de acceso compatibles con el estándar *OpenFlow*.

OpenFlow es uno de los primeros y más ampliamente utilizados estándares *SDN*; define el protocolo de comunicaciones abierto en un entorno *SDN* que permite al controlador *SDN* (cerebro de la red) hablar con el plano de reenvío (conmutadores,

enrutadores, etc.) para realizar cambios en la red. El controlador *SDN* es responsable de mantener todas las reglas de la red y de proporcionar las instrucciones necesarias a la infraestructura subyacente sobre cómo debe manejarse el tráfico. Esto permite a las empresas adaptarse mejor a sus necesidades cambiantes y tener un mejor control sobre sus redes. El controlador *Floodlight* fue originalmente ofrecido por big *switch* como parte del Proyecto *OpenDaylight*, pero en junio de 2013, big *switch* se alejó en medio de lo que parece ser un conflicto de intereses con Cisco. Si bien el controlador de *Floodlight* aún es de código abierto, no está involucrado con el proyecto *OpenDaylight*.

2.4.1 Beneficios del controlador Floodlight [43]

Puede ser favorable para los desarrolladores porque les brinda la capacidad de adecuar fácilmente el software y desarrollar aplicaciones, y está escrito en Java. Se incluyen las interfaces del programa de aplicación de transferencia de estado representativo (*API REST*) que facilitan la programación de la interfaz con el producto, y el sitio web de Floodlight ofrece ejemplos de codificación que ayudan a los desarrolladores a construir el producto. El controlador de *Floodlight* es compatible con *openstack*, un conjunto de herramientas de software que ayudan a construir y administrar plataformas de computación en la nube para nubes públicas y privadas.

2.4.2 ¿Cómo funciona Floodlight controller en entornos SDN? [43]

En la **Fig. 6** se grafica la funcionalidad del controlador Floodlight:

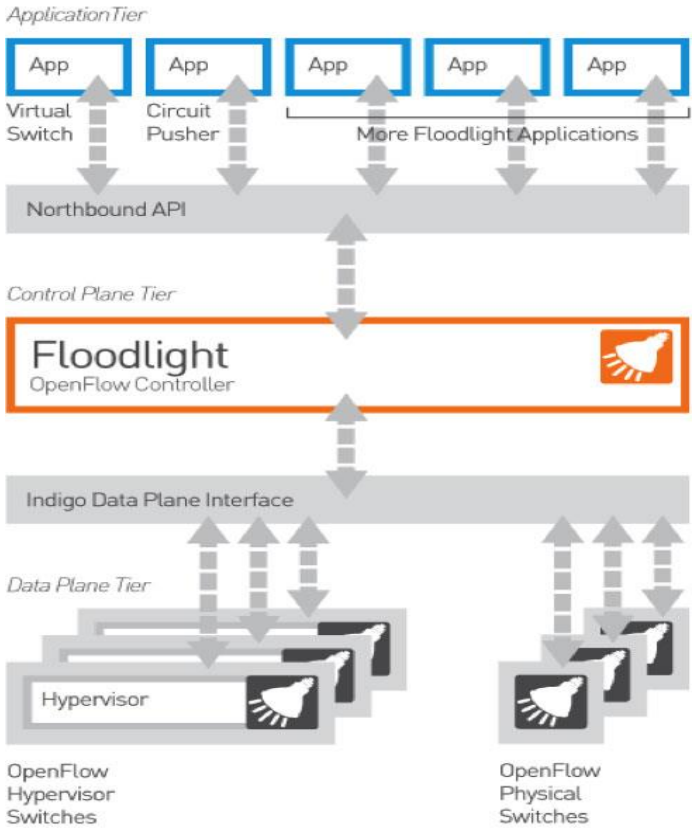

***Fig. 6* Funcionalidad de Floodlight [43]**

2.5 Raspberry Pi [44]

Los modelos de *Raspberry Pi* más utilizados (b, b+, 2b, 3b, 4) incorporan interfaz ethernet. El modelo 3B es uno de los más comercializados ya que incorpora además una interfaz *WiFi,* esto le da una gran ventaja ante sus antecesores y la mayoría de los usuarios de otros modelos utilizan un pincho *USB WiFi* (dongle *WiFi*) para actualizar el sistema y conectarse a la red.

Raspberry Pi, es un ordenador de tamaño de tarjeta de crédito que se conecta a su televisor y un teclado. Es una placa que tolera varios componentes precisos en un

ordenador común. Es un pequeño ordenador capaz, que puede ser utilizado por muchas de las cosas que su *PC* de escritorio hace, como hojas de cálculo, procesadores de texto y juegos. También reproduce vídeo de alta definición», apuntan en la página web del producto. En la **Fig. 7** se muestra un cuadro comparativo con los diversos modelos, excluyendo a la actual Raspberry Pi 4:






Raspberry Pi	Modelo A	Modelo A+	Modelo B	Modelo B+	RPi V2 modelo B	RPi 3 modelo B
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	700MHz ARM1176JFZ-S	700MHz ARM1176JFZ-S	700MHz ARM1176JFZ-S	700MHz ARM1176JFZ-S	900MHz Quad-core ARM Cortex-A7	1.2Ghz Quad Cortex A53
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	250Mhz VideoCore IV	400Mhz VideoCore IV
RAM	256Mb	512Mb	512Mb	512Mb	1Gb	1Gb
USB	1	1	2	4	4	4
Video	RCA, HDMI	Jack, HDMI	RCA, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI
Boot	Memoria SD	Memoria microSD	Memoria SD	Memoria microSD	Memoria microSD	Memoria microSD
Wireless	No tiene	No tiene	No tiene	No tiene	No tiene	802.11n / Bluetooth 4.1
Red Ethernet	No tiene	No tiene	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100
Alimentación	5V / 2Amp	5V / 2Amp	5V / 2Amp	5V / 2Amp	5V / 2Amp	5V / 2,5Amp
GPIO	26 pines GPIO	40 pines GPIO	26 pines GPIO	40 pines GPIO	40 pines GPIO	40 pines GPIO
Tamaño	85,6 x 53,98 mm	65 x 56 mm	85,6 x 53,98 mm	85 x 56 x 17 mm	85 x 56 x 17 mm	85 x 56 x 17 mm

***Fig. 7* Modelos de Raspberry Pi [44]**

2.5.1 ¿Qué es Raspberry Pi y para qué sirve? [44]

Este proyecto fue ideado en 2006 pero no fue lanzado al mercado hasta febrero de 2012. Fue desarrollado por un grupo de la Universidad de Cambridge y su misión es fomentar la enseñanza de las ciencias de la computación los niños.

La placa tiene varios puertos y entradas, dos *USB*, uno de Ethernet y salida *HDMI* (Interfaz Multimedia de Alta Definición, por sus siglas en inglés). Estos puertos permiten

conectar el miniordenador a otros dispositivos o periféricos como teclados, ratones y pantallas, etc.

También contiene un procesador ARM (Advanced RISC Machine, arquitectura de procesadores ya sean en 32 bits o 64 bits) que corre a 700 MHz, un procesador gráfico VideoCore IV y hasta 1 GB de memoria RAM, estos datos técnicos dependen del modelo y varían entre ellos. Se pueden instalar sistemas operativos libres por medio de una tarjeta micro SD. Es un mini ordenador muy funcional y práctico que debido a su tamaño puede funcionar para muchos otros propósitos, aunque hay que tener algunas ideas sobre programación o de computación.

2.6 Ryu controller [45]

Ryu es un controlador de red definida por software de código abierto (*SDN*) diseñado para aumentar la agilidad de la red al facilitar la administración y adaptación de la forma en que se maneja el tráfico. Por lo general, el controlador *SDN* es el cerebro y la base del entorno *SDN*, que comunica información a los conmutadores y enrutadores con *API* (interfaces de programa de aplicación) hacia una dirección A, y hasta a las aplicaciones y la lógica empresarial con *API* a una dirección B. El controlador *Ryu* es compatible con *NTT* y también se implementa en los centros de datos en la nube de *NTT*. El controlador *Ryu* proporciona componentes de software, *API*s bien definidas, que facilitan a los desarrolladores la creación de nuevas aplicaciones de control y gestión de red.

Ryu es compatible con varios protocolos para administrar dispositivos de red, como *OpenFlow*, netconf, OF-config, etc. Considerando a *OpenFlow*, *Ryu* es totalmente compatible con 1.0, 1.2, 1.3, 1.4, 1.5 y Nicira Extensions. Todo el código está disponible gratuitamente bajo la licencia Apache 2.0. Como dato curioso, *Ryu* significa "flujo" en japonés.

2.6.1 ¿Cómo se ajusta un controlador Ryu en entornos SDN? [45]

En la **Fig. 8** se muestra el ajuste del controlador SDN en los entornos SDN.

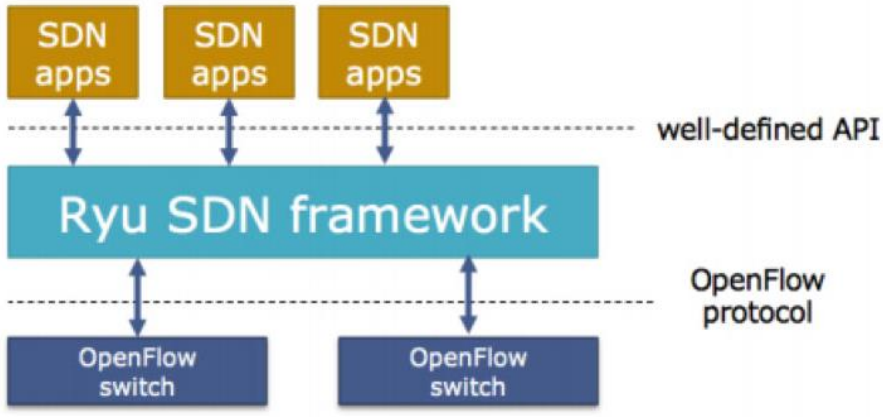

Fig. 8 **Ajuste de un controlador Ryu en entornos SDN [45]**

El código fuente de *Ryu* controller está alojado en GitHub y es administrado y mantenido por la comunidad abierta de *Ryu*. *OpenStack* ejecuta una colaboración abierta centrada en el desarrollo de un sistema operativo en la nube que puede controlar los recursos de cómputo, almacenamiento y redes de una organización, admite

implementaciones de *Ryu* como el Controlador de red. Está escrito en su totalidad en *Python*, todo el código de *Ryu* está disponible bajo la licencia Apache 2.0 y está abierto para que lo use cualquier persona. El controlador *Ryu* es compatible con los protocolos de administración de red *NETCONF* y OF-config, así como con *OpenFlow*, que es uno de los primeros y más ampliamente implementados estándares de comunicaciones *SDN*. También soporta extensiones de Nicira. El controlador *Ryu* puede usar *OpenFlow*, u otros protocolos, para interactuar con el plano de reenvío (conmutadores y enrutadores) para modificar la forma en que la red manejará los flujos de tráfico. Se ha probado y certificado para funcionar con varios conmutadores *OpenFlow*, incluidos open vSwitch y las ofertas de Centec, Hewlett Packard, IBM y NEC.

2.6.2 Librerías de Ryu [46]

Ryu cuenta con una impresionante colección de librerías que van desde soporte a múltiples protocolos SouthBound hacía varias operaciones para el procesamiento de la red. Con lo que respecta a protocolos SouthBound, *Ryu* tiene soporte para OF – Config (OpenFlow management and configuration protocol). *OVSDB* (open vswitch database management protocol), *NETCONF* (Protocolo de Configuración de Red por sus siglas en inglés), *XFLOW* (netflow and sflow) y otros protocolos de terceros. Netflow es soportado por Cisco y algunas otras marcas y es específico para una red *IP*.

2.6.3 Administradores y procesos del núcleo [46]

El administrador *Ryu* es el ejecutable principal. Cuando se ejecuta se aloja en una dirección *IP* específica y un puerto específico, después de eso, cualquier conmutador que soporte *OpenFlow* puede conectarse al administrador de *Ryu*. El administrador de aplicaciones es el componente fundamental para todas las aplicaciones de *Ryu*.

2.6.4 NorthBound de Ryu [46]

En la capa de las *API*, *Ryu* incluye un plug-in de OpenStack-Neutron que soporta configuraciones para *VLAN*. Ryu también es compatible con una interfaz *REST* para sus operaciones con *OpenFlow*. Además, con el uso de *WSGI* (Web Server Gateway Interface), se puede introducir fácilmente nuevas *API* REST en la aplicación.

2.6.5 Aplicaciones Ryu [46]

Ryu se distribuye con múltiples aplicaciones como: simple_switch, *router*, insolation, firewall, *GRE* (Generic Routing Encapsulation) Tunnel, *VLAN*, etc. Las aplicaciones Ryu con entidades de un único subproceso, las cuales implementan varias funcionalidades. Las aplicaciones *Ryu* envían mensajes asincrónicos entre sí. El funcionamiento de la arquitectura de una aplicación *Ryu* se muestra en la **Fig 9.**

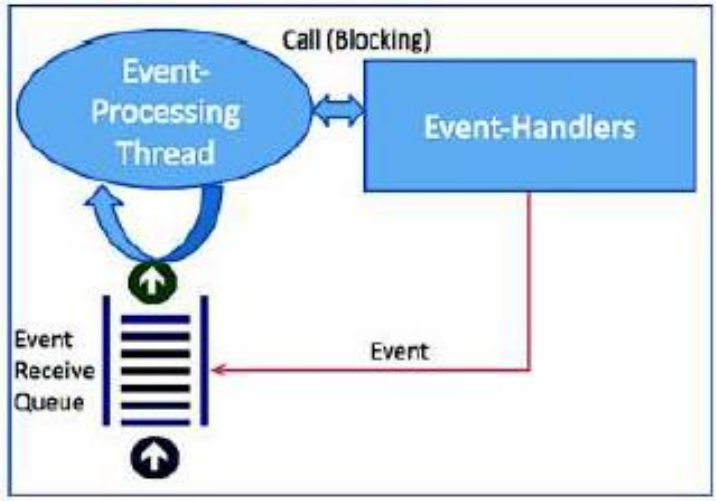

Fig. 9 **Funcionamiento de la arquitectura de Ryu [46]**

Todas las aplicaciones de *Ryu* tienen una cola de recepción para los eventos que utilizan principalmente el método *FIFO* (First In First Out) para preservar el orden de los acontecimientos, también, cada aplicación incluye un hilo de procesamiento de eventos de la cola. El hilo del bucle principal escoge eventos de la fila de recepción y llama al controlador de eventos correspondiente. Por lo tanto, el controlador de eventos se llama dentro del contexto del hilo de procesamiento de eventos que funciona a manera de bloqueo, es decir, cuando un controlador de eventos tiene el control, no se procesarán más eventos de la aplicación *Ryu* hasta que el control sea devuelto. Las aplicaciones de *Ryu* pueden ser ejecutadas y configuradas pasando el archivo de configuración del gestor de *Ryu* (*Ryu* manager) como se observa en la **Fig. 10**.

```
ryu-manager [--flagfile <path to configuration file>]
[generic/application specific options...]
```

Fig. 10 **Ejecución de aplicación a través de Ryu manager [46]**

2.6.6 Estructura de código en Ryu [47]

El código principal del controlador está organizado en la carpeta /*Ryu*. Se denota las funcionalidades de los componentes clave para familiarizarse con ellos:

- **app/:** Contiene un conjunto de aplicaciones que se ejecutan cobre el controlador.

- **base/:** Contiene la clase base para aplicaciones *Ryu*. La clase RyuApp en app_manager.py se hereda cuando se crea una nueva aplicación.

- **controller/:** Contiene el conjunto de archivos necesarios para manejar las funciones *OpenFlow* (paquetes de los conmutadores, generación de flujos, gestión de eventos de red, obtención de estadísticas, etc.).

- **lib/:** Contiene un conjunto de librerías de paquetes para analizar diferentes cabeceras del protocolo y una librería de OF-CONFIG. Además, incluye analizadores para Netflow y Sflow.

- **ofproto/:** Contiene la información específica del protocolo *OpenFlow* y de analizadores relacionados para soportar diferentes versiones de OF – Protocol (1.0, 1.2, 1.3, 1.4).

- **topology/:** Contiene código que realiza el descubrimiento de la topología relacionada con conmutadores *OpenFlow* y maneja información asociada (ejemplo: puertos, enlace). Internamente utiliza el protocolo *LLDP* (Link Layer Discovery Protocol).

2.7 Protocolo OpenFlow [48] [49] [50]

Es la primera interfaz de comunicaciones estándar que está definida entre los planos de control y los planos de datos en la arquitectura de la *SDN*. *OpenFlow* admite el acceso directo y manejo de dispositivos, ya sean reales o virtuales ubicados en el plano de datos, tales como conmutadores o enrutadores. Las tecnologías *SDN* basadas en el protocolo *OpenFlow* aprueban a dinamizar aplicaciones de forma que la red se adecúe a las necesidades cambiantes del entorno y así, empequeñecer la complejidad de la operatividad y administrabilidad. Las ventajas de usar *OpenFlow* son:

- **Programabilidad en el plano de gestión:** Se encarga de la innovación y variación de aplicaciones, también de apresurar la creación de nuevas características e introducción de servicios de la red.

- **Inteligencia centralizada en el plano de control**: Facilita el abastecimiento del hardware y software a un único controlador maestro y remoto, afinando el rendimiento y la gestión de las políticas de red.

- **Abstracción en el plano de datos:** Desajusta el hardware y el software de los dispositivos de red, en otras palabras, la configuración física de la lógica.

Desde que apareció *OpenFlow* en 2008, se han elaborado diversas versiones con grandes diferencias entre estas. Por lo tanto, la primera versión de *OpenFlow*, denominado OF10, denota todos los procesos para desvincular y comunicar los planos

de datos y control, e integrar la interoperabilidad de redes SDN/OpenFlow con las redes convencionales. Esta versión usa protocolos como Ethernet, *IPv4* y *TCP/UDP* para clasificar y reenviar la información en el plano de datos. La segunda versión, también llamada OF11, soporta el balance de carga, la comunicación con múltiples controladores y usa protocolos adicionales como *SCTP* (Stream Control Transmission Protocol) y *MPLS* (Multiprotocol Label Switching) para la clasificación y el reenvío de información. Las versiones OF12 y OF13 agregan la capacidad de *QoS* y DiffServ (Servicios Diferenciados) y también soporta metadatos del protocolo IPv6. La versión OF14 agrega escalabilidad a las tablas de flujo de datos, configuración simultánea de múltiples conmutadores y soporta *PBB* (Provider Backbone Bridge). La versión OF15 agrega la posibilidad de poseer información programada, en otras palabras, que se reenvíe cerca de los tiempos de reloj del conmutador para fortalecer la sincronización entre los dispositivos del plano de datos. En esta ocasión se usa la versión *OpenFlow* 1.3 (OF13) ya que es la versión que soporta de manera estable a *Mininet*.

OpenFlow fue inicialmente propuesto por la Universidad de Stanford y actualmente ha sido estandarizado por la *ONF* (Open Networking Foundation por sus siglas en ingles). Se procede a evidenciar las ventajas de este protocolo. Una de las ventajas de utilizar *OpenFlow* es la flexibilidad que ofrece al programar de forma soberana el tratamiento de cada uno de los flujos en la red. Dicho de otra forma, *OpenFlow* determina como un flujo puede ser definido en la red, cuáles son las acciones que se pueden realizar para cada uno de los paquetes que pertenecen al flujo, es decir, la manera en que la información será transformada en los elementos de la red.

Por lo que, una entrada en la tabla de flujo de datos de un *switch OpenFlow* se forma por la unión de la definición de un flujo y un conjunto de acciones explícitas. Un flujo se forma por medio del establecimiento de los valores de uno o más campos de la cabecera del paquete que se procese por el dispositivo *OpenFlow*. Estos campos forman un grupo de doce elementos que combinan características de los protocolos de capa de enlace de datos, red y transporte de acuerdo con la arquitectura *TCP/IP*. Entonces, se muestran los componentes de una entrada en la **Fig. 11**.

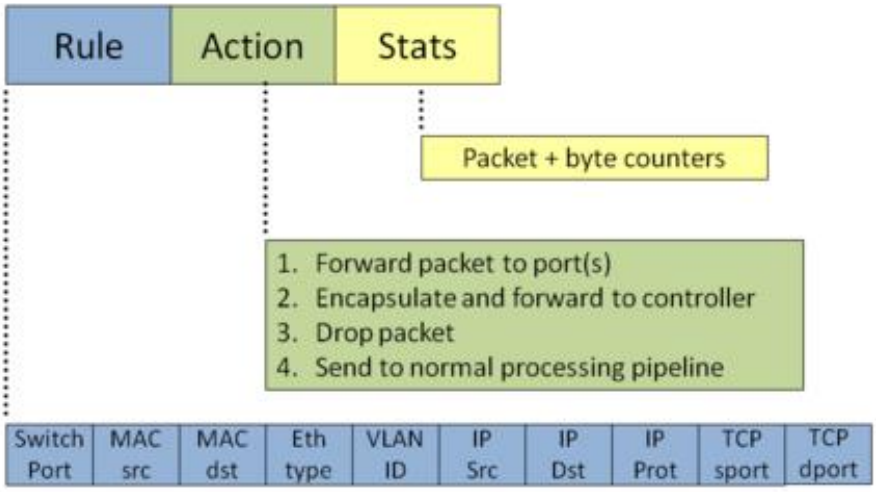

Fig. 11 **Componentes de una entrada de flujo OpenFlow [2]**

Esto quiere decir que al definir un flujo se pueden ver que las reglas de enrutamiento de un paquete no se limitan a la dirección *IP* o a la dirección *MAC* (Media Access Control) de los paquetes. El protocolo *OpenFlow* es muy amplio, de modo que los elementos formados por los campos del paquete pueden contener valores que sean exactos o inexactos que responden a cualquier valor que presente el paquete en

comparación con el campo. El enrutamiento puede estar cedido por otras características del paquete, como los puertos de origen y protocolo de transporte de destino.

OpenFlow se fundamenta en tablas de flujo de datos construidas en las *TCAM* (Ternary Content Addresable Memory), que están incorporadas en *switches* y *routers* siendo diferentes para cada fabricante. Sin embargo, cuentan con funciones comunes que pueden ser manejadas mediante el protocolo *OpenFlow*.

2.7.1 Controlador OpenFlow [2]

El controlador *OpenFlow* tiene casi las mismas características que cualquier otro controlador para *SDN*. Consiste en un control centralizado ya sea física o lógicamente que ejecuta aplicaciones de control sobre la configuración de la red y gestiona el flujo en los conmutadores. Evidentemente, es necesario que el controlador implemente el protocolo *OpenFlow*, para que, por medio de mensajes, pueda comunicarse con los elementos de conmutación (*switches* con *OpenFlow* habilitado) mediante el envío de comandos a la red. El controlador actúa como un sistema operativo de la red, actuando sobre el plano de control, como una interfaz entre las aplicaciones de control de la red y la red misma.

2.7.2 Mensajes OpenFlow [51] [52]

Los mensajes enviados por el controlador pueden necesitar o no una respuesta del *switch* y se catalogan de acuerdo a los siguientes tipos:

Mensajes controladores – *switch*: Estos mensajes son generados en el controlador y el *switch* debe necesariamente responder a los mismos. En este tipo se encuentran los siguientes mensajes:

- **Mensajes de características (features):** Generalmente después de establecer el canal entre el *switch* y el controlador OpenFlow, el controlador envía un mensaje al *switch* para identificarlo y conocer sus capacidades y características básicas.

- **Mensaje de configuraciones (Configuration):** El controlador es capaz de establecer y consultar los parámetros de configuración del *switch*. El *switch* responde a la consulta enviada por el controlador.

- **Mensajes de modificaciones de estado (Modify-State):** Son mensajes enviados por el controlador para gestionar el estado de los *switches*. Su principal objetivo es añadir o remover entradas en las tablas de flujo de datos de los conmutadores, o añadir y eliminar reglas.

- **Mensajes de lectura de estado (Read-State):** Son los mensajes enviados por el controlador para leer el estado y recopilar estadísticas.

- **Mensajes packet-out (Send-Packet):** Estos mensajes son utilizados por el controlador para enviar paquetes por un puerto específico del *switch* y para que transmita paquetes recibidos a través de mensajes Packet-in. Los mensajes Packet-out deben de contener un paquete completo o un identificador de buffer que referencia el paquete almacenado en el *switch*. El mensaje también debe contener una lista de reglas a ser aplicadas en un orden específico, una lista vacía descartará el paquete.

- **Mensajes barrier:** Son mensajes utilizados por el controlador para asegurar que los mensajes de órdenes se han cumplido, o para recibir notificaciones de las operaciones realizadas.

- **Mensajes role-request:** Estos mensajes son usados por el controlador para establecer el rol de su canal *OpenFlow* o para consultar el rol actual.

- **Mensajes asynchronous-configuration:** Este mensaje es usado por el controlador para establecer un filtro adicional en los mensajes asincrónicos que quieres recibir en su canal *OpenFlow* o para consultar este filtro. Este mensaje es de utilidad cuando el *switch* está conectado a múltiples controladores y es comúnmente enviado en el establecimiento del canal *OpenFlow*.

Mensajes asincrónicos: Con mensajes que se envían desde el *switch* hacia el controlador cuando ha llegado un paquete, cuando ha ocurrido un error o cuando ha existido un cambio de estado. En este tipo de mensajes se encuentran los siguientes:

- **Mensaje packet-in:** Es un mensaje que el *switch* envía al controlador cuando ha llegado un paquete y este no tiene ninguna coincidencia en la tabla de flujo de datos. El controlador procesa el paquete y responde con un mensaje Packet-out.

- **Mensaje de eliminación de flujo (Flow-Removed):** Este paquete se envía cuando el tiempo de inactividad de un flujo ha expirado. Puede ser enviado por el controlador o por el *switch*.

- **Mensaje de estado de puerto (Port Status):** Este paquete se envía del *switch* al controlador cuando se ha producido un cambio en el puerto.

- **Mensaje de error (Error):** El *switch* notifica al controlador si existen errores o fallos a través de estos paquetes.

Mensajes simétricos: Estos mensajes pueden ser enviados por el controlador o por el *switch* sin que exista una solicitud previa. En este tipo se encuentran los siguientes mensajes:

- **Mensajes de saludo (Hello):** Estos mensajes son intercambiados entre el *switch* y el controlador al momento del establecimiento de la conexión.

- **Mensajes de eco (Echo):** Este mensaje se utiliza para verificar si existe conexión entre el controlador y el *switch*, además también puede ser utilizado para medir la latencia y el ancho de banda de la conexión.

- **Mensajes de experimentación (Experimenter):** Estos mensajes permiten dar una funcionalidad extra al *switch* OpenFlow, serán usados en próximas versiones del protocolo.

2.7.3 Funcionamiento de OpenFlow [53]

En una *SDN* el control se provee a través del protocolo *OpenFlow* y en controlador *OpenFlow*. Los flujos pueden ser definidos de la forma elegida por el controlador, por ejemplo, todos los paquetes enviados por la máquina con *IP* “A” a la máquina con IP “B”, o incluso todos los paquetes que pertenecen a la *VLAN* (Virtual Local Area Network o Red de Área Local Virtual) “C”. Las acciones son definidas a partir de flujos e incluyen:

- Reenviar los paquetes pertenecientes a un flujo o puerto.

- Enviar paquetes al controlador.

- Descartar paquetes como media de seguridad.

- Modificar los campos de cabecera de los paquetes.

- Encaminar los paquetes para procesamiento normal en el dispositivo.

El último literal de la lista mencionada anteriormente permite que el tráfico experimental no entorpezca el tráfico de producción de la red. Con este desviamiento de tráfico es posible tener equipos mezclados que procesan el tráfico enviado de acuerdo con los protocolos y características de los equipos integrados, y al mismo tiempo, de forma espaciada tener tráfico basado en la tabla de flujos de datos del *switch*. Además de las acciones, la arquitectura *OpenFlow* también tiene contadores para el control de cada flujo. Cada flujo tiene contadores para el control de: número de paquetes, número de bytes en el flujo y la duración del flujo. Estos contadores son implementados para cada entrada en la tabla de flujo y pueden ser supervisados y accedidos por el controlador a través del protocolo *OpenFlow*.

Después de formar una entrada en la tabla de flujo de datos, cada paquete entrante a un *switch OpenFlow* se compara con cada una de las entradas de esta tabla y si se encuentra una coincidencia, se considera que el paquete pertenece a ese flujo y a su vez se procede con la acción señalada predeterminadamente. Si no se encontrase una coincidencia, el paquete se envía por completo al controlador para ser procesado, o se puede eliminar de acuerdo a alguna política preestablecida. Alternativamente, solo la cabecera del paquete se envía al controlador y se mantiene el resto almacenado en la memoria del *switch*. Si el paquete se reenvía al controlador, el controlador revisará el paquete y dependiendo del análisis de la aplicación, instaurará una nueva entrada de flujo. En **la Fig. 12** se muestra un diagrama de flujo del proceso realizado por el *switch*.

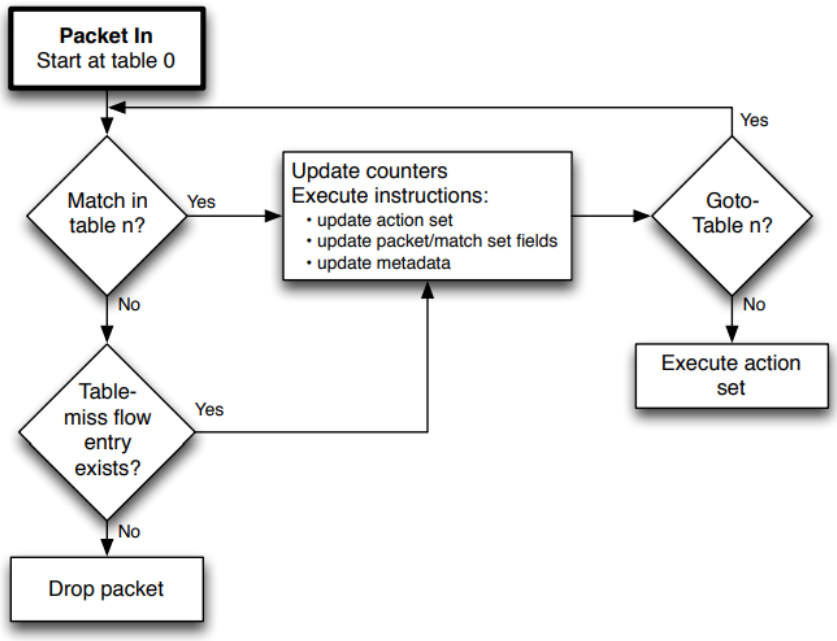

Fig. 12 **Entrada de un paquete a un switch OpenFlow [53]**

Por lo general, los paquetes que alcanzan al controlador siempre pertenecen al primer paquete de un nuevo flujo y, dependiendo de la aplicación, el controlador puede elegir instalar una nueva regla en el *switch* para que todos los paquetes de determinado flujo sean enviados al controlador o sean tratados de forma individual.

2.8 OpenDaylight OpenFlow manager [54]:

Un controlador se ubica entre la aplicación y la red e interactúa con los elementos de la red (por ejemplo, conmutadores) en dirección sur manejando una variedad de diferentes protocolos. En el norte, llamándolo así, presenta una abstracción de la red

utilizando en la práctica *API REST* comunes. El vehículo controlador para esta aplicación es *ODL*. OpenFlow Manager (*OFM*) es una aplicación que aprovecha esta innovación para administrar la red *OpenFlow*. En la **Fig. 13** se observa su arquitectura.

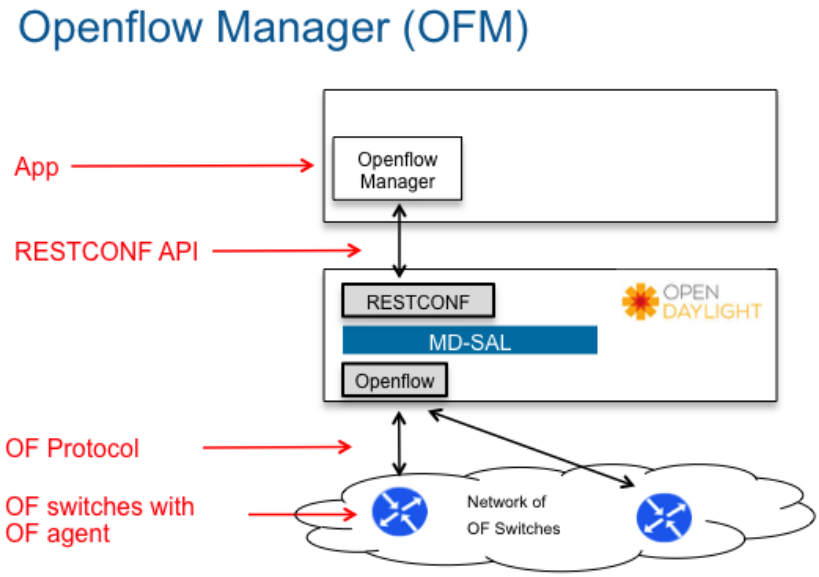

Fig. 13 **Arquitectura de OFM [54]**

De abajo hacia arriba, se muestra una red de conmutadores *OpenFlow* que ejerce un paradigma de reenvío "match / action" para establecer operaciones de conmutación de flujo por medio de la red. Los conmutadores *OpenFlow* aceptan un agente OpenFlow y manejan *MPLS* (Multiprotocol Label Switching o Conmutación de Etiquetas Multiprotocolo) para rotular paquetes de conmutación en la red, y *OSPF* (Open Shortest Path First o Protocolo Abierto de los Enlaces) o *IS-IS* (Intermidiate System to Intermidiate System o Sistema Intermedio a Sistema Intermedio) para conservar y distribuir la base de datos de topología (estado de enlace) entre los otros enrutadores

de la red. Uno de los enrutadores es un *BGP-LS* (Border Gateway Protocol – Link State o Puerta de Enlace de Frontera – Estado de Enlace) y transporta una copia de la base de datos de topología al controlador ODL. Los enrutadores también ejecutarán *PCEP* (significa protocolo de elemento de cálculo de ruta) utilizado por el controlador ODL para indicar a un enrutador de origen que configure una ruta de ingeniería de tráfico *MPLS* a un enrutador de destino. Estos son detalles muy específicos de la red y del protocolo, que francamente el usuario final puede no conocer o importar. Aquí es donde entran en juego *OpenDaylight* y las aplicaciones. En la **Fig 14** muestra la pantalla inicial de *OFM* una vez se hayan cargado los dispositivos conectados a la red con protocolo *OpenFlow*:

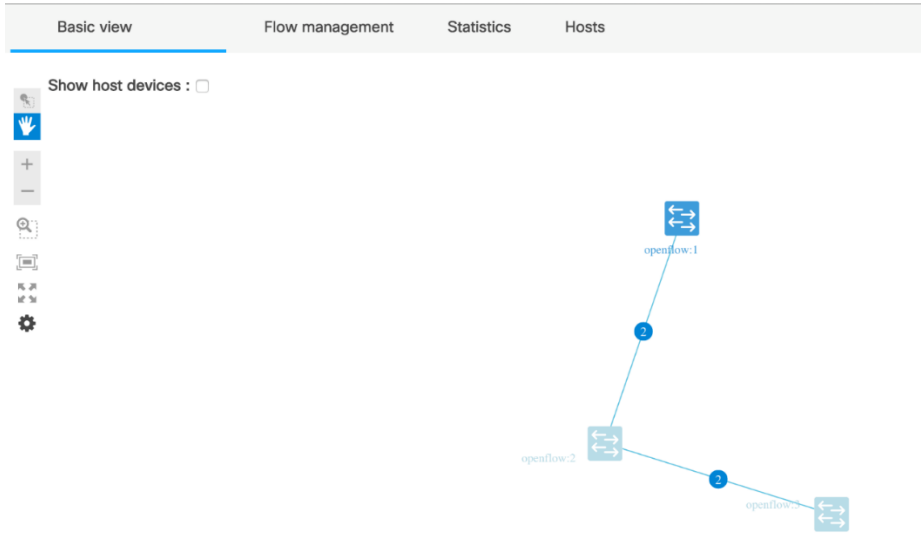

***Fig. 14* Pantalla inicial de OFM [54]**

Las subfunciones básicas de *OFM* son:

- **La pestaña vista básica:** Es la pestaña predeterminada, esta muestra la topología de los dispositivos habilitados para OpenFlow en su red y los hosts que están conectados a ellos.

- **La pestaña de gestión de flujo:** Determina la cantidad de flujos para cada dispositivo habilitado para OpenFlow en su red, también se lo usa para observar una lista de los flujos dispuestos actualmente, para agregar, para modificar y eliminar los flujos.

- **La pestaña estadística:** Facilita las estadísticas para los flujos configurados en la red como para los puertos de los dispositivos correspondientes.

- **La pestaña Hosts:** Provee la información resumida para los dispositivos que juegan el papel de host que están habilitados para *OpenFlow* que dispone *OFM*.

2.9 Flow manager [54]:

Flow manager es una aplicación de controlador *Ryu* que le permite al usuario el control manual de las tablas de flujo de datos en una red *OpenFlow*. Se pueden crear, modificar o eliminar flujos directamente desde la aplicación web. El usuario también puede monitorear los interruptores *OpenFlow* y divisar las estadísticas. Flow Manager es excelente para aprender *OpenFlow* en un entorno de laboratorio, o en conjunto con otras aplicaciones para acordar el comportamiento de los flujos de red en un ambiente de producción.

Para acceder a la interfaz gráfica web de la aplicación Flow manager habría que digitar en la barra URL lo siguiente: **http://localhost:8080/home/index.html.** Antes de ingresar a la aplicación se debe de ejecutar cualquier aplicación del controlador Ryu junto con la aplicación Flow manager, como por ejemplo junto con la aplicación simple_switch. Las características de Flow manager son las siguientes:

- Agregar, modificar o eliminar entradas de flujo en las tablas de flujo.
- Agregar, modificar o eliminar un grupo de tablas y medidores.
- Hacer una copia de seguridad, restauración de la unidad local.
- Ver tablas, grupos de tablas y medidores.
- Ver estadísticas del *switch*.
- Presenciar la topología de red.

A continuación, en la **Fig. 15** se muestra la página de inicio de la interfaz gráfica web de Flow manager:

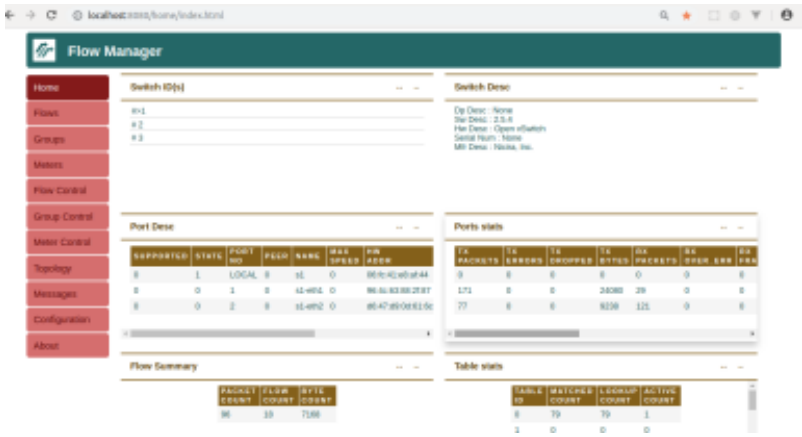

Fig. 15 **Página de inicio de Flow manager [55]**

2.10 Tarjeta de memoria micro SD [56] [57]

Se usaron tarjetas con una capacidad de 16 GB para no tener futuros problemas de almacenamiento, ya que el *SO* (Sistema Operativo) al momento de instalarlo ocupa una cantidad aproximada de 4 GB, aparte de esto, se tienen que descargas los elementos necesarios para las prácticas a realizar.

Las tarjetas micro SD son una opción almacenamiento que pertenece al estándar de la “SD Association”. Hay varios formatos de tarjetas, entre los que se enfatizan SD (usados en las cámaras digitales), mini SD y micro SD. El estándar que necesario en este proyecto de titulación es el micro SD, ya que el sistema operativo de la *Raspberry Pi 3 Model B* se lo instala en este tipo de tarjetas. Estas tarjetas de memoria se dividen en diferentes clases, las cuales se pueden observar en la **TABLA II**:

TABLA II

Clases de Tarjetas Micro SD [57]

Clases de Tarjetas	Velocidad de Escritura
Class 2	2 MB/s
Class 4	4 MB/s
Class 6	6 MB/s
Class 8	8 MB/s
Class 10	10 MB/s

También hay diversas velocidades y utilidades para una tarjeta micro SD dependiendo de su clase, como se observa en la **TABLA III**:

TABLA III

Velocidades de Tarjetas SD [56]

Categoría/Class o Clase	Velocidad mínima de transmisión	Nivel de velocidad	Recomendación de uso
UHS 3	30 MB/s	Máxima	Grabación de vídeo 4K y 2K
UHS 1/ Class10	10 MB/s	Alta	Grabación de vídeo HD, full HD y fotos
Class 6	6 MB/s	Media	Grabación de vídeo HD, full HD
Class 4	4 MB/s	Baja	Grabación de vídeo HD
Class 2	2 MB/s	Baja	Grabación de vídeo HQ

Al momento de comprar una tarjeta micro SD lo principal a tener en cuenta es la capacidad y la clase de ésta, para la implementación del proyecto se utilizaron tarjetas micro SD de 16 GB de capacidad de clase.

2.11 Cable RJ-45 o Ethernet [58]

Es el componente más importante de cualquier red, sea de kilómetros de distancia como de una simple red en la oficina o el hogar, es el cable Ethernet, y es el tipo de cable utilizado comúnmente para interconectar todos los dispositivos que conforman una *LAN* (Local Area Network o red de área local), se incluyen impresoras, discos externos, *routers*, escáneres, *switches* y por supuesto las propias computadoras.

Este cable es el encargado de transportar todos los datos que usualmente circulan por una red, siendo el más utilizado en las instalaciones estándar el denominado como “Cruzado”. Si bien en la actualidad las redes inalámbricas están cada vez más extendidas, lo cierto es que las redes mediante cables ofrecen ventajas que las redes *WLAN* (Wireless Local Area Net o red de área local inalámbrica) por el momento no pueden alcanzar, como largas distancias y estar libres de interferencias. En la **Fig. 16** se visualizan los pares trenzados del cable RJ-45:

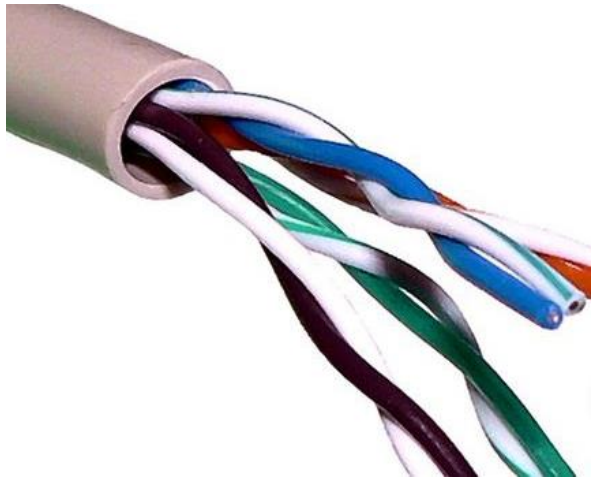

***Fig. 16* Cable RJ-45 [58]**

En la actualidad, un cable Ethernet de red estándar es un cable *UTP* 8 Categoría 5 el cual contiene 4 pares de hilos, recubiertos con una malla metálica cuya intención es la de resguardar los datos de todo tipo de interferencias eléctricas.

2.12 Tipos de cables ethernet [58]

Actualmente en el mercado existen varios tipos de cables de red, *UTP* (Unshielded Twisted Pair o cable de par trenzado sin blindaje), *STP* (Shielded Twisted Pair o cable de par trenzado blindado), de fibra óptica y coaxiales, siendo el cable *UTP* en el más utilizado en las implementaciones comunes, debido esencialmente a su flexibilidad, bajo coste y facilidad de implementación.

Este tipo de cables, específicamente los *UTP*, se dividen en categorías, siendo las mismas convenientes a un determinado y específico uso. Básicamente son siete las categorías de cables *UTP* que podemos encontrar, las cuales son:

- **Cat 1:** Esta se compone de dos pares de hilos y es utilizado únicamente para las instalaciones telefónicas.

- **Cat 2:** Utilizado únicamente en las instalaciones telefónicas, una diferencia al cable cat 1, el cable cat 2 es idóneo para transferir datos a una tasa máxima de 4 Mbps.

- **Cat 3:** Esta es mayormente conocido como "*Ethernet 10BaseT*", ofrece una tasa de transferencia máxima de 10 Mbps. Hasta la aparición de los cables *UTP* Categoría 5 era el más usado para las conexiones de red, tanto en las empresas como en el hogar.

- **Cat 4:** También llamada "Ethernet 10baseT/TokenRing", es capaz de dar una velocidad de transferencia máxima de hasta 20 Mbps.

- **Cat 5:** Es el más manejado en la actualidad. Conocido como "*Ethernet 100baseT/10BaseT*" puede alcanzar hasta una tasa de transferencia máxima de 100 Mbps. En la actualidad es estándar para la implementación de redes tanto de corporaciones como hogareñas, como de pequeñas empresas.

- **Cat 6:** Este tipo de cable *UTP* ha sido desarrollado con la intención de obtener grandes velocidades de transferencia a distancias mucho más largas, por ello puede alcanzar tasas de transferencia de hasta 1 Gbps a una distancia de 100 metros.

- **Cat 7:** Elaborado para que pueda llegar a una velocidad de 10 Gbps de transferencia a distancias considerables, hasta 100 metros, el cable *UTP* categoría 7 también es muy resistente a las interferencias eléctricas. Está dotado de un aislamiento muy eficaz para que el ruido eléctrico no sea un problema. Cabe recalcar que es compatible con las anteriores categorías.

3 Implementación del módulo

En esta sección se describe la elaboración y construcción del módulo para las prácticas efectuadas, así como también la fabricación del case o bases en forma de torre para los componentes utilizados. También se describen las configuraciones que se realizaron en la Zodiac FX y en los controladores SDN utilizados. Se procede a describir la configuración, el funcionamiento y la integración a este proyecto. En la **Fig. 17** y **Fig. 18** se visualizan los esquemas del módulo:

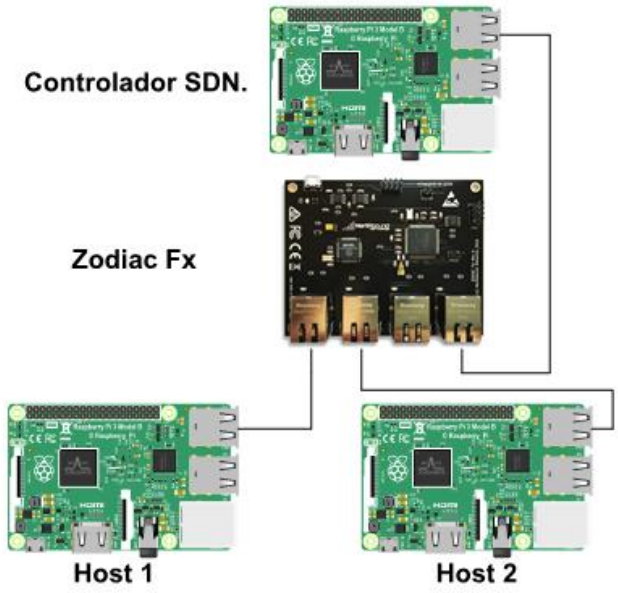

Fig. 17 **Esquema gráfico del módulo [Fuente propia]**

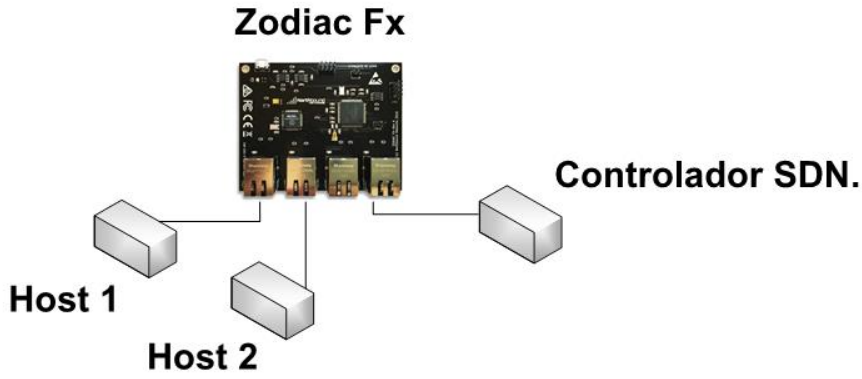

Fig. 18 **Diagrama esquemático del módulo [Fuente propia]**

3.1 Componentes del módulo:

El módulo de red *SDN* está compuesto de las siguientes piezas:

- *Raspberry Pi 3 Model B* x3.
- Tarjetas de memoria del 16 GB.
- *Zodiac FX*
- Cables de red RJ-45

El módulo está conformado por una *Raspberry Pi* que cumple el rol de servidor o controlador, 2 Raspberries que actúan como hosts y ambas realizan intercambios de información o datos. Las Raspberries están conectadas a la tarjeta *Zodiac FX*, la cual

es un mini *switch* que permite la comunicación entre dichos componentes. La Raspberry, en la que se instale el controlador, debe de ser conectada al puerto # 4 de la *Zodiac FX* debido a que ese puerto es de tipo OpenFlow además en aquel puerto se encuentra el plano de control y el plano de datos se encuentra en los puertos 1,2 y 3. Las Raspberries están conectadas al mini switch mediante cables RJ-45. En cada una de las Raspberries se debe de instalar un Sistema Operativo, ya sea Ubuntu, Raspbian, etc. El SO (*System Operative*) más recomendable es Raspbian debido a su sencillez, ligereza, bajas probabilidades de corrupciones para archivos del sistema y estabilidad al momento de ejecutar los programas necesarios. Este SO debe de ser instalado sobre una tarjeta microSD, ya sea de 8 GB o de 16 GB, se recomienda de 16 GB por todos los paquetes que se deben de descargar. para efectuar las prácticas y evitar congelamientos debido a la incorrecta elección de capacidad de almacenamiento.

3.2 Stack del módulo

El case o soporte del módulo fue hecho mediante la aplicación AutoCAD y Solidworks para poder hacer una impresión en tres dimensiones de los diseños de cada parte de los soportes para así formar un Stack, donde en el soporte inferior se ubica la tarjeta *Zodiac FX* para realizar las conexiones desde la parte baja del Stack hacia arriba; en los otros tres pisos restantes se ubican las Raspberries. En la **Fig. 19** se visualiza el diseño de un piso para el stack del módulo.

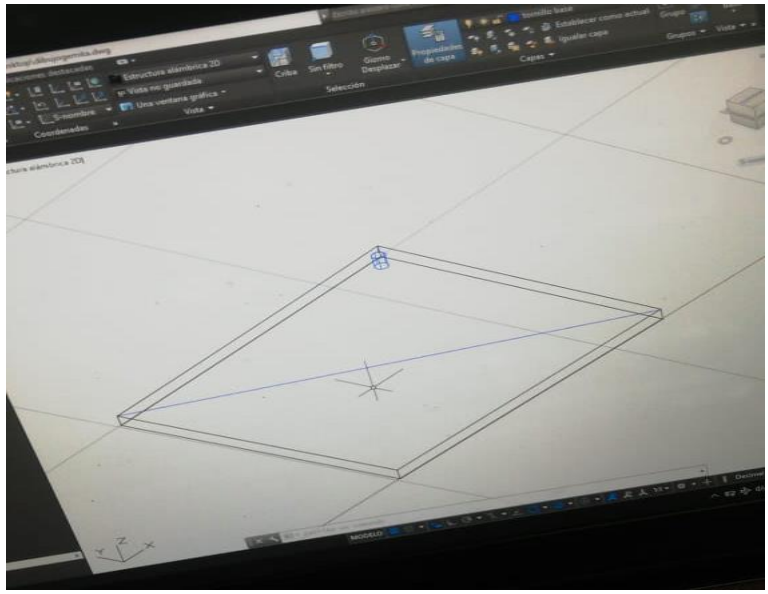

Fig. 19 **Diseño del Stack [Fuente propia]**

El diseño es elaborado como una torre de cuatro pisos o plantas, tres para las Raspberries (dos como hosts y uno como servidor) y una para la tarjeta *Zodiac FX* que hará la conexión y comunicación entre los dispositivos. Debido a la diferencia de tamaño que hay entre las Raspberries y la tarjeta *Zodiac FX*, se tuvo que nivelar de acuerdo a la posición de los tornillos que harán de soporte en el stack para cada dispositivo.

Al principio de la elaboración de la estructura se pensó en hacer otro tipo de case para este proyecto de titulación, pero se llegó a la conclusión de que en este caso es lo más factible debido al número de artefactos que habría en cada módulo. Se lo podrá usar con o sin la pantalla LCD (Liquid Cristal Display) ya que el uso de este dispositivo sería opcional para cada módulo, teniendo así facilidad a la hora de conectar los periféricos a las Raspberries debido a que el case no es cerrado, en las siguientes imágenes se observa la fabricación y el diseño. En la **Fig. 20** se detalla la elaboración de los soportes.

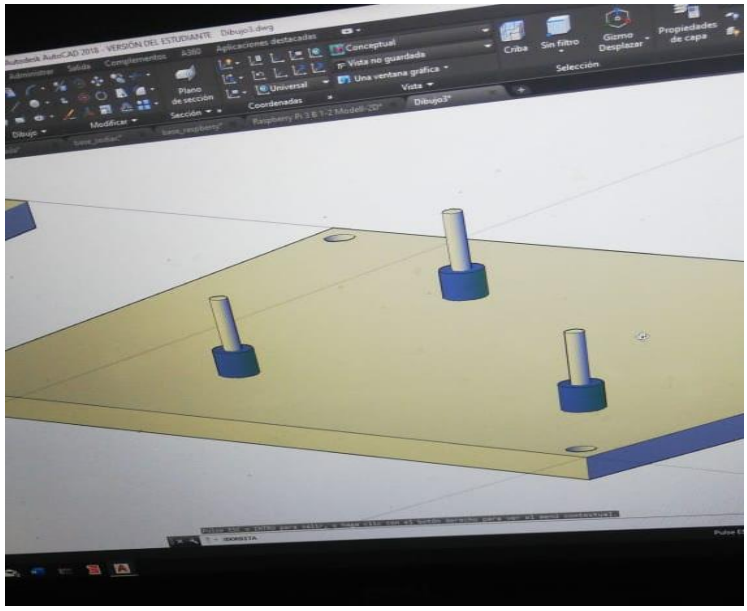

Fig. 20 **Soporte de una planta [Fuente propia]**

Una vez realizado el esquema, se imprime la primera planta del stack para así comprobar que no haya inconvenientes al momento de colocar los componentes y, también, confirmar que los soportes se ajusten a los dispositivos, cada soporte debe de ser atornillado para evitar la desestabilización de éstos y así evitar futuros daños en los dispositivos. Cuando no se encuentren problemas, se prosigue con la impresión de las demás plantas y cada una de las plantas se las une con tornillos para stacks, se debe de verificar que hagan presión y que puedan soportar movimientos leves sin que se descoloquen del orificio asignado. En la **Fig. 21** se realiza la impresión 3D de la primera planta.

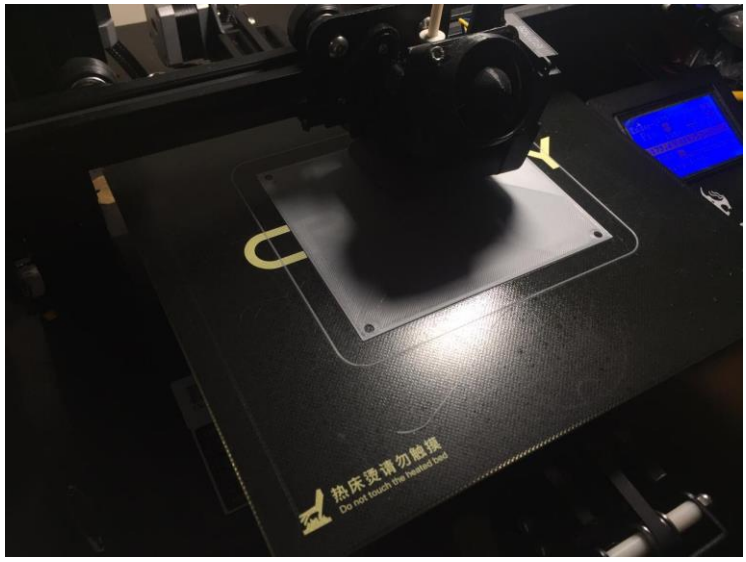

Fig. 21 **Impresión en 3D [Fuente propia]**

En la **Fig. 22** se visualizan los soportes donde se atornillan los componentes del módulo.

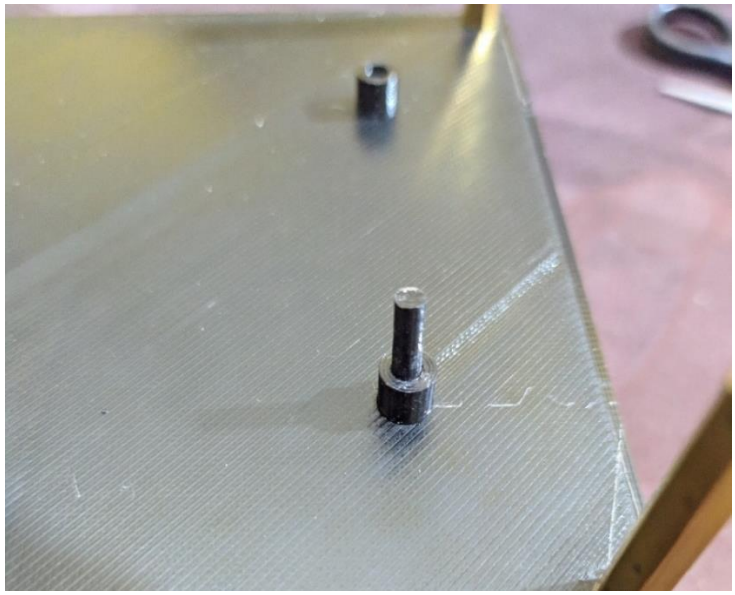

Fig. 22 **Ajuste del soporte [Fuente propia]**

En la **Fig. 23** se tiene una mejor vista de la primera planta del stack ya con los orificios asignados para atornillar el dispositivo.

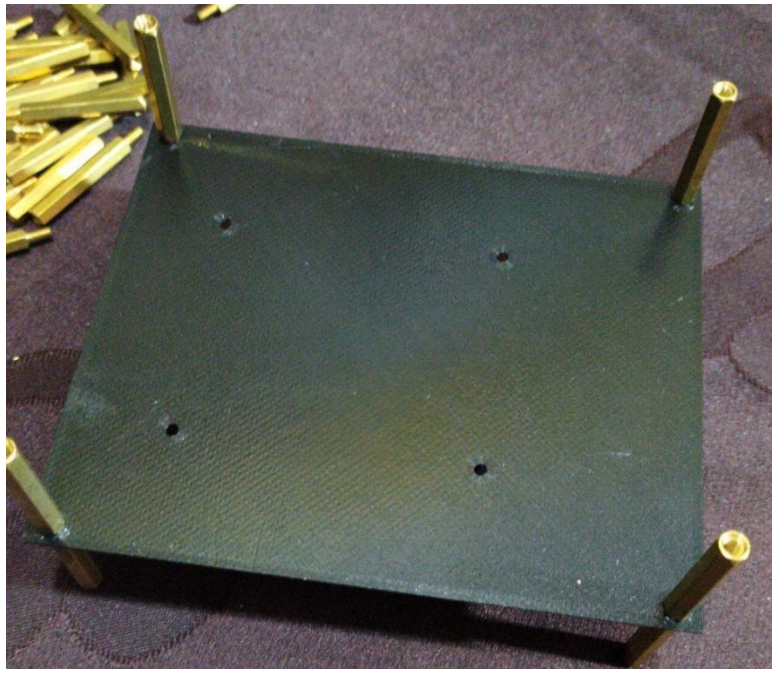

Fig. 23 **Primer soporte del case [Fuente propia]**

Ya con los orificios de los tornillos hechos, se puede atornillar el primer dispositivo, cabe recalcar que el tamaño de la *Raspberry Pi* con el tamaño de la *tarjeta Zodiac FX* no son iguales, por lo que se debe de tener cuidado al momento de realizar los cortes o perforaciones necesarias. En la **Fig. 24** se presenta una *Raspberry Pi* encajada en la planta armada.

Fig. 24 **Raspberry Pi colocada [Fuente propia]**

Una vez realizados los pasos mencionados anteriormente para cada planta del stack, a continuación, se debe armar y atornillar todos los elementos, como se visualiza en la **Fig. 25**. En el módulo se destaca la implementación de una pantalla LCD táctil de 5 pulgadas la cual se conecta a una *Raspberry Pi*, se sugiere que la raspberry pi que sea conectada a la pantalla LCD sea la que actúe como controlador o servidor en cada una de las prácticas, para evitar una frecuente manipulación de la pantalla y alargar el tiempo de vida útil de la pieza. La pantalla tiene su propio lápiz óptico para ser utilizada.

Fig. 25 **Módulo SDN [Fuente propia]**

4 Análisis y resultados de las prácticas

4.1 Análisis y resultados de la práctica #1

Los programas que se utilizan se encuentran en el portal de mega https://mega.nz/ y sus credenciales son (usuario: repositoriosdn@gmail.com clave: titulacion2020). Para el desarrollo de esta práctica se identificaron los dispositivos y programas que se utilizaron en la implementación del módulo (**Anexo A**).

Se estableció que para el desarrollo de cada una de las prácticas se realizaron diferentes configuraciones para cada controlador SDN y adicional a esto se descargaron los archivos correspondientes para el funcionamiento de cada controlador en su respectivo entorno.

Se descargaron programas de emulación virtual como Virtual Box debido a que en algunos casos las prácticas se desarrollarían en distintos escenarios y el entorno para cada controlador SDN no era compatible para ser ejecutado en una tarjeta con arquitectura ARM como la Raspberry Pi obteniendo como resultado la configuración requerida para el funcionamiento de cada controlador SDN en su respectivo entorno.

4.2 Análisis y resultados de la práctica #2

En esta práctica (**Anexo B**), se realizó un balanceador de cargas utilizando un controlador *SDN* que permitirá aprender cómo se dirigen las solicitudes *HTTP* a los respectivos servidores.

Un balanceo de cargas sirve para la distribución de tráfico en la red ya que las peticiones de tipo web se realizan desde un cliente hacia un servidor utilizando el protocolo *HTTP*. Una vez que se recibe la petición se inicia la conexión de cliente-servidor y se inicia el intercambio de paquetes entre estos.

Como resultado se realizó la configuración de un balanceador de cargas de tipo software que distribuirá la carga entre varios servidores al mismo tiempo, a su vez si un servidor deja de funcionar el tráfico que iba dirigido a ese será enviado hacia otro servidor que esté disponible y en funcionamiento, esta configuración se ajusta a la topología conformada por 6 *hosts*, 1 *switch* y un controlador remoto ese controlador remoto fue el controlador *SDN* de nombre *POX* configurado a través del puerto 6633. En la **Fig. 26** se muestra la topología del controlador cuando hace la función de balanceador de carga:

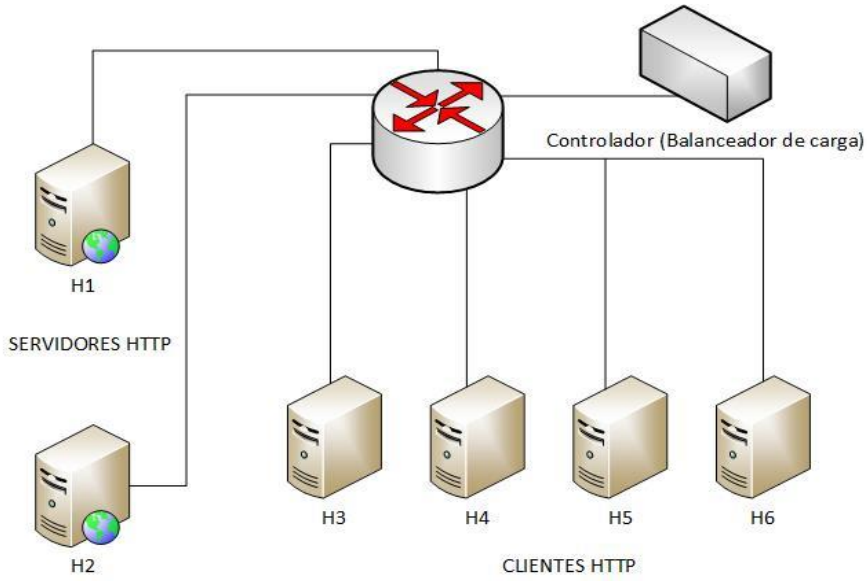

Fig. 26 **Topología del controlador POX como balanceador de carga [Fuente propia]**

En la **Fig. 27** se realiza el envío de tráfico de datos desde el host 3 hasta el controlador.

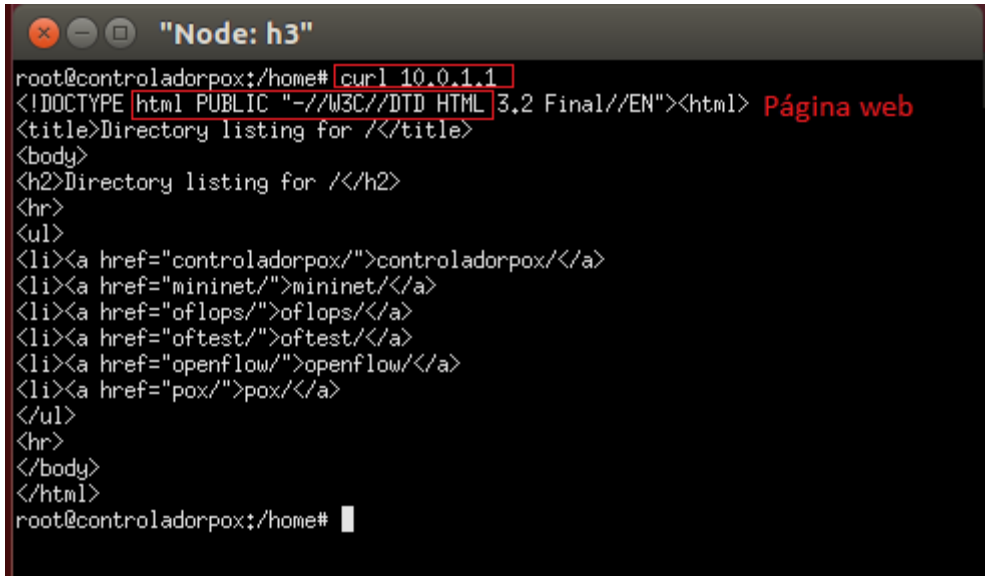

Fig. 27 **Envío de tráfico del host 3 [Fuente propia]**

En la **Fig. 28** se observa el redireccionamiento del tráfico al servidor *HTTP* 1 con dirección *IP* 10.0.0.1:

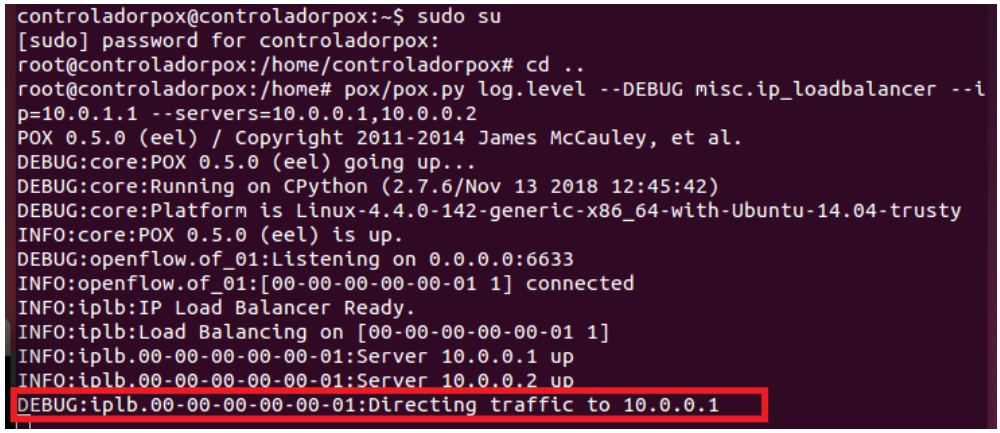

Fig. 28 **Redireccionamiento de tráfico [Fuente propia]**

En la **Fig. 29** se visualiza una solicitud realizada desde el host 3 al servidor *HTTP* 1:

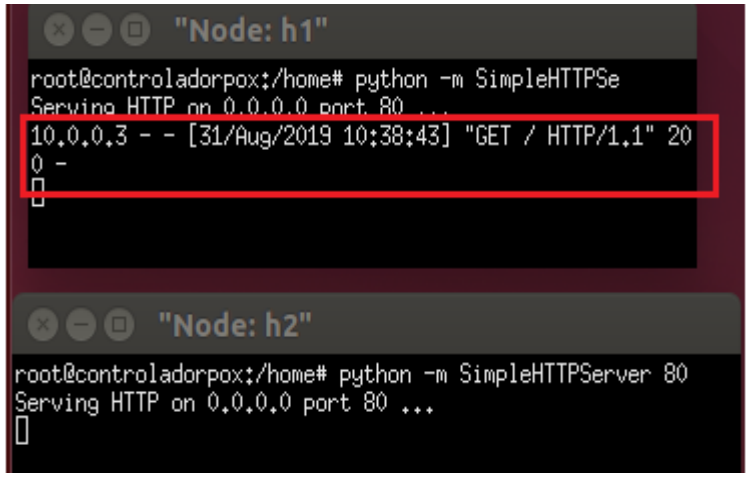

Fig. 29 **Solicitud del host [Fuente propia]**

En la **Fig. 30** se plasma el envío de tráfico del host 3 al controlador:

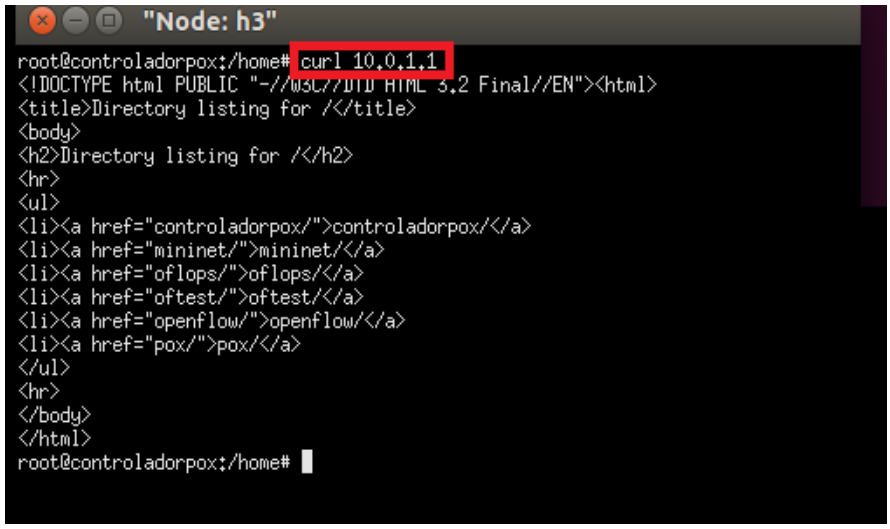

Fig. 30 **Envío de tráfico del host 3 [Fuente propia]**

En la **Fig. 31** se representa el redireccionamiento del tráfico hacia el servidor *HTTP* elegido:

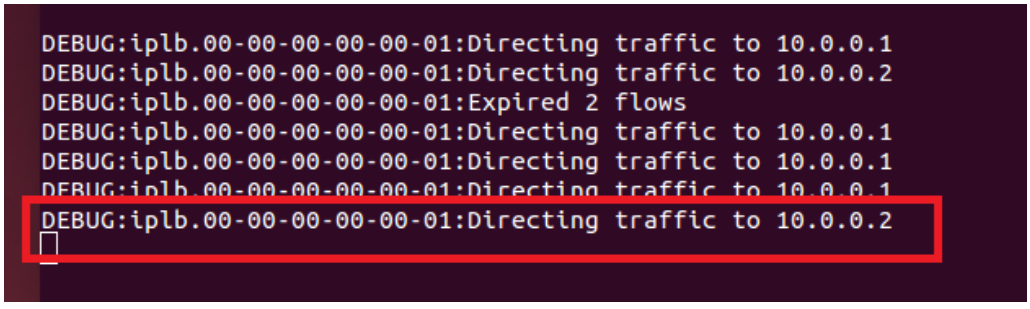

Fig. 31 **Redireccionamiento de tráfico al servidor http 1 [Fuente propia]**

Ahora, en la **Fig. 32** se refleja la solicitud enviada desde el host 3 al servidor *HTTP* 2:

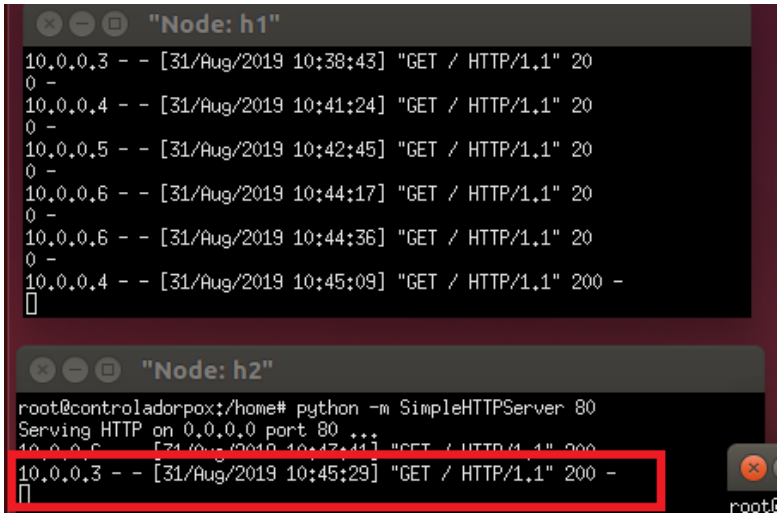

***Fig. 32* Solicitud desde host [Fuente propia]**

4.3 Análisis y resultados de la práctica #3

Para la práctica número tres (**Anexo C**) se utilizaron diferentes dispositivos como la tarjeta Zodiac FX, 2 tarjetas Raspberry Pi que actuaron como respectivos hosts para la ejecución de la práctica.

Se realizó la comunicación entre el controlador SDN de nombre POX el cual fue configurado para ser ejecutado en la tarjeta *Raspberry Pi 3 Model B* y tener una comunicación serial con la tarjeta *Zodiac FX* para acceder a la terminal de la tarjeta mencionada utilizando el programa Putty. En la **Fig. 33** se visualiza el acceso a la tarjet *Zodiac FX* desde PuTTY:

Fig. 33 **Terminal de Zodiac FX [Fuente propia]**

Con las configuraciones que se realizaron será posible que los dispositivos *Raspberry Pi* y la tarjeta *Zodiac FX* logren actuar en conjunto como un hub o concentrador que permita conectar varios dispositivos dentro de una misma conexión.

Es importante realizar la configuración de las interfaces de red de la *Raspberry Pi* en la que se ejecuta el controlador *SDN POX* donde eth0 se refiere a la red cableada y WLAN0 a la red inalámbrica de la *Raspberry Pi*. La **Fig. 34** es una guía de como configurar las interfaces de la *Raspberry Pi* para tener acceso tanto a la red WiFi como a la red Ethernet:

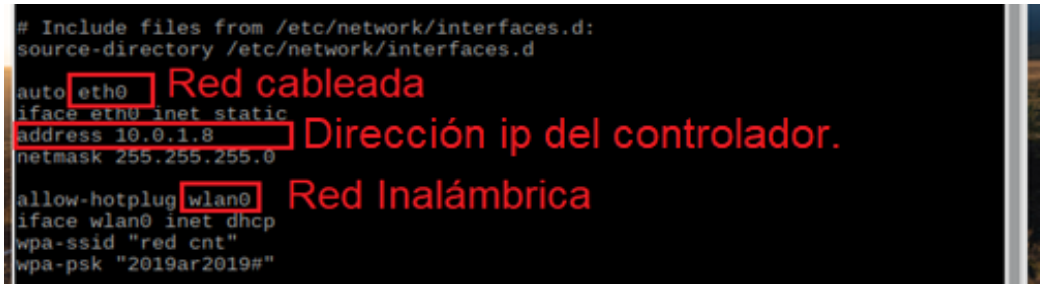

Fig. 34 **Configuración de las interfaces [Fuente propia]**

Luego de descargar y actualizar los archivos necesarios para el controlador *SDN* se inicia el controlador *POX* a continuación se utiliza el componente *OpenFlow*. of_01 este componente es parte del controlador y se comunica con la tarjeta *Zodiac FX* debido a que utiliza el protocolo *OpenFlow* versión 1.0. Todo esto se lo demuestra en la **Fig. 35**.

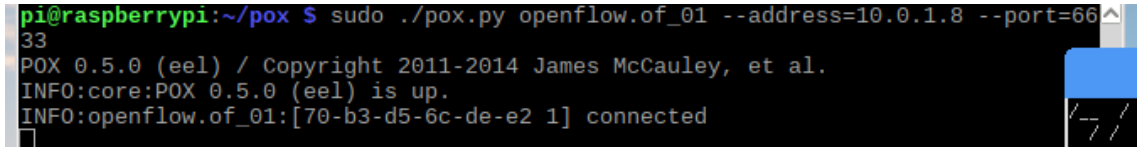

Fig. 35 **Inicio del controlador POX [Fuente propia]**

Se revisa la ejecución y conexión entre los dispositivos en la terminal de la tarjeta *Zodiac FX* para asegurarse que no haya errores previos a la ejecución del controlador *POX* con el complemento para que funcione como un hub o concentrador y permita la conexión.

Se realiza la conexión de la red de internet mediante un cable de red al puerto # 1 de la tarjeta *Zodiac FX* y se verifica que se hayan escrito flujos ya que esto sucede cuando

empieza el intercambio de paquetes por medio del controlador, así como se muestra en la **Fig. 36**.

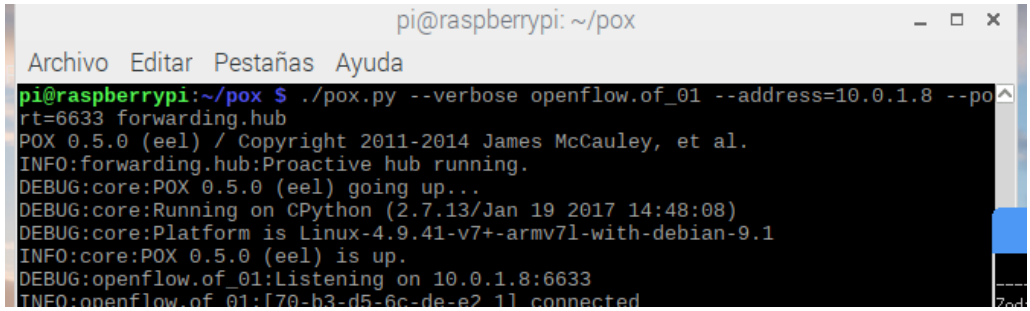

Fig. 36 **Inicio del controlador POX como un hub [Fuente propia]**

Para verificar si hay un flujo instalado en la tarjeta *Zodiac FX,* dentro del terminal que fue abierto por el comando PuTTY, se puede ingresar el comando show flows como se muestra en la **Fig. 37**.

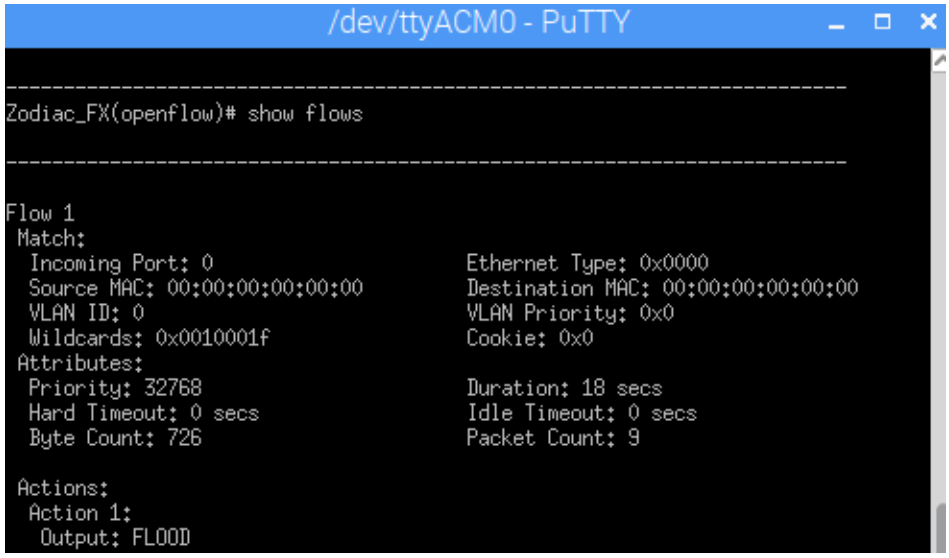

Fig. 37 **Flujo 1 [Fuente propia]**

Se establecieron conexiones en los puertos restantes como los puertos # 2 y 3 de dispositivos como las *Raspberry Pi* implementadas en el módulo obteniendo como resultado la asignación de direcciones *IP* a los dispositivos conectados dentro de la red y acceso a internet desde los hosts conectados a la tarjeta *Zodiac FX.*

4.4 Análisis y resultados de la práctica # 4

Para esta práctica se utilizó una máquina virtual ya configurada anteriormente, la cual estará disponible para descargar en el repositorio acordado. El propósito de la práctica es básicamente crear y eliminar reglas en el controlador instalado en dicha máquina virtual, el cual es *Floodlight*, también se observan los paquetes *OpenFlow* de Loopback mediante *Wireshark* (todo este proceso se lo puede revisar minuciosamente en el **Anexo D**), entre otras.

Con el fin de crear y eliminar las reglas establecidas, se usaron las tres topologías de *Mininet* (simple, linear o de árbol) para poder observar como reacciona en cada una de ellas. Después de creadas las tres topologías mencionadas, se puede examinar en el controlador *Floodlight* el diagrama de red de cada topología creada y mediante *Mininet* hacer ping en cada host de las topologías.

Las reglas creadas consisten en restringir el envío de datos de un host cualquiera a otro host (al momento de crear las reglas se pueden definir los hosts), en otras palabras, evita que por ejemplo el host 1 haga ping con el host 2 o host 3.

Hay dos formas de eliminar las reglas creadas, una opción es eliminar una a una en base a la ID de la regla o eliminar todas con un comando específico que se observa en el **Anexo D**.

4.5 Análisis y resultados de la práctica # 5

Se realizó la configuración de una red definida por software utilizando el controlador *SDN* de la marca HP llamado *HP VAN SDN CONTROLLER* siendo capaz de modificar la configuración de la red a través del protocolo *OpenFlow* (**Anexo E**).

El controlador utilizado para esta práctica maneja las versiones 1.0 y 1.3 del protocolo *OpenFlow*, en general un controlador puede determinar como los flujos se comportan dentro de la red mediante el reporte de los dispositivos que se encuentran en dicha red hacia el controlador en cada caso que se lo necesite: cuando hay nuevos ingresos de flujos en la red y cuando el dispositivo no tiene una definición de qué hacer con un paquete, dado el caso pregunta al controlador que acción debe tomar, el controlador indica a los dispositivos por medio de la actualización de las tablas de flujo de datos de los dispositivos administrados y realiza una acción que ha sido configurada.

Debido a los requerimientos del sistema que necesita este controlador para funcionar se realizó la instalación del sistema operativo Ubuntu versión 12.04.5 con el cual su arquitectura es compatible.

Instalando el repositorio de Ubuntu: **icehouse** se añade el archivo de la nube del cual se descargan y actualizan los archivos para trabajar con el controlador en la versión instalada de Ubuntu. Trabajando así con Keystone un servicio de OpenStack que proporciona autenticación de clientes, siendo Keystone un componente que ayuda en la autenticación del controlador. Al descomprimir el archivo de funcionamiento del controlador se encuentran errores debido a los requerimientos que el controlador necesita, como se puede visualizar en la **Fig. 38**.

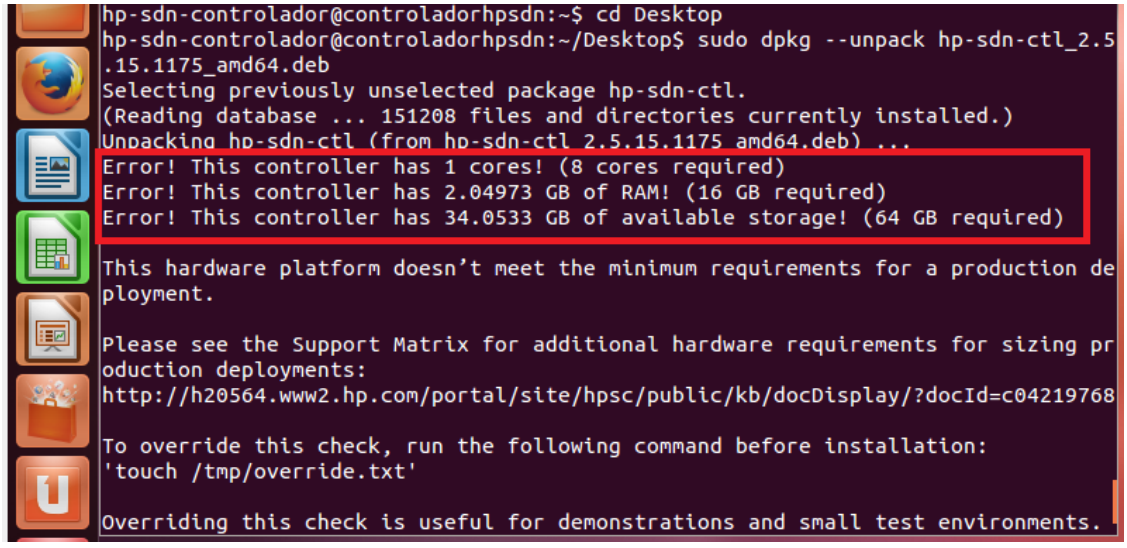

Fig. 38 **Error de descompresión del controlador [Fuente propia]**

El mensaje de error indica que la máquina virtual desde la cual se está trabajando no posee los requerimientos, pero estos sí se pueden modificar a través del comando "**touch**" en Ubuntu debido a que el sistema se encuentra instalado sobre una máquina virtual a diferencia de la *Raspberry Pi* donde no se pueden modificar aquellas opciones un motivo por el cual este controlador no puede ser ejecutado sobre una *Raspberry Pi*. Esto se observa en la **Fig. 39**.

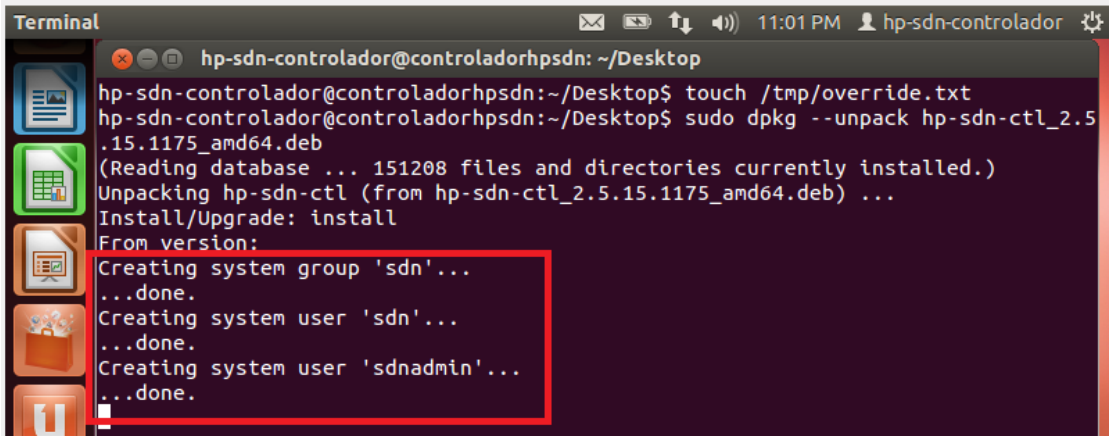

Fig. 39 **Descompresión del archivo [Fuente propia]**

Al descomprimirse el archivo se crean los usuarios de sistema y grupos con lo que trabaja el controlador, aquellos son utilizados por el servicio de autenticación Keystone, como se menciona anteriormente. En la **Fig. 40** se representa la ejecución de las credenciales.

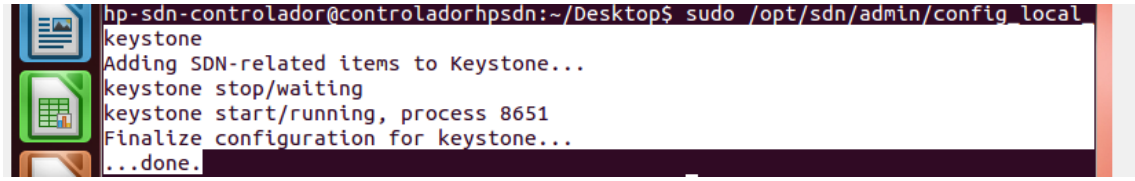

Fig. 40 **Ejecución de credenciales [Fuente propia]**

Al realizar la instalación del controlador y la respectiva actualización de paquetes para su funcionamiento, se establecen las credenciales de usuario con la configuración de Keystone. Luego de iniciar las credenciales por default (defecto) con Keystone, las credenciales asignadas serán:

- Usuario: sdn

- Contraseña: skyline

Se ingresa al controlador con la dirección *IP* de la máquina virtual donde se está ejecutando el controlador añadiendo el puerto de comunicación y añadiendo la excepción de seguridad en el navegador web, configurar el controlador y deshabilitar el modo híbrido que se encuentra establecido por default.

Este controlador posee robustez en su seguridad debido a eso todas las aplicaciones que se instalan en el controlador deben pasar una validación de firmas jar/zip por lo cual para la práctica se plantea un ambiente experimental/desarrollo donde las aplicaciones sin firmar no son admitidas, se procede a deshabilitar la validación.

Luego de deshabilitar la validación se realiza la instalación de la aplicación *Flow Maker* (Deluxe) la versión más completa a comparación de la versión gratuita de la aplicación de Northbound Networks encargada del desarrollo de la tarjeta *Zodiac FX*. En la **Fig. 41** se visualiza la pestaña de la aplicación Flow Maker en la interfaz gráfica web del controlador.

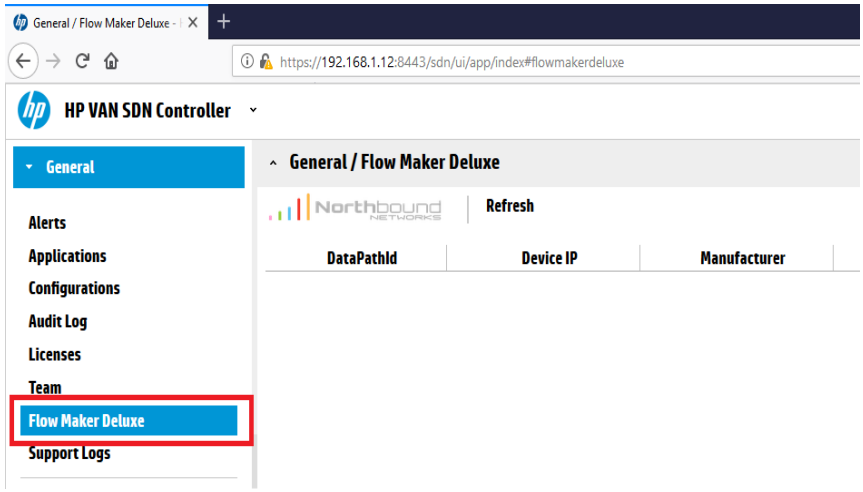

Fig. 41 **Flow Maker Deluxe [Fuente propia]**

Se ingresa mediante el programa Putty a la tarjeta *Zodiac FX* y se configuran las direcciones de la tarjeta y del controlador *OpenFlow*, como se señala en la **Fig. 42**.

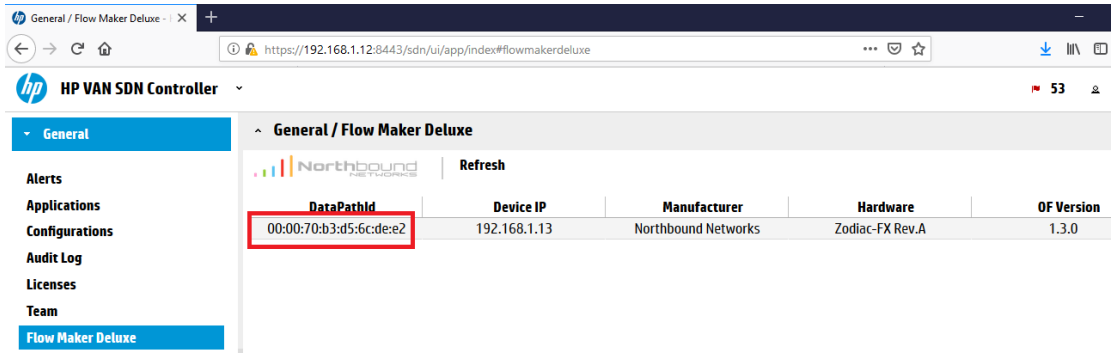

Fig. 42 **Tarjeta Zodiac FX detectada por el controlador [Fuente propia]**

Se instalan los sistemas operativos en las *Raspberry Pi* que se utilizan como host y se conecta una *Raspberry Pi al* puerto # 1 de la tarjeta *Zodiac FX*, se comprueba la dirección *IP* que fue asignada por la tarjeta *Zodiac FX*, esto se puede visualizar en la **Fig. 43**.

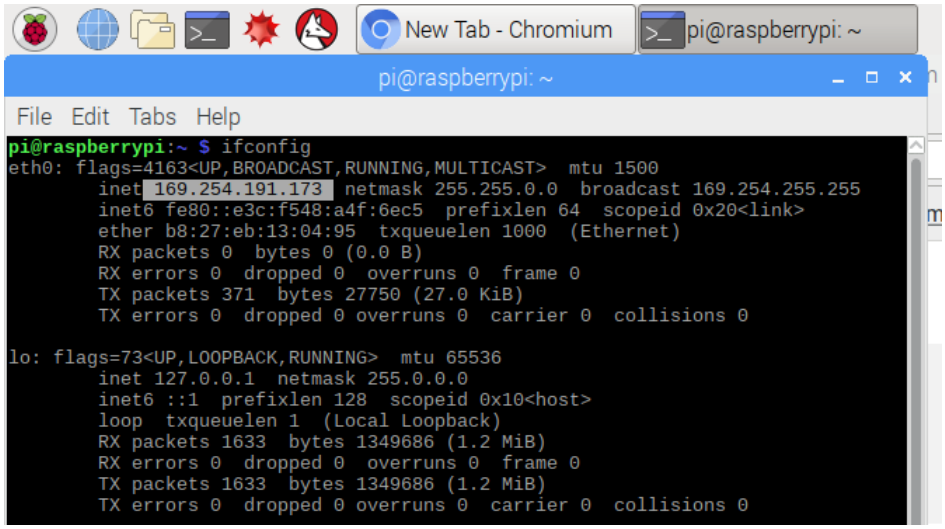

Fig. 43 **Raspberry Pi 3 Model B conectada [Fuente propia]**

Se realiza ping a la dirección *IP* de la *Raspberry Pi* conectada en el puerto # 1 de la tarjeta *Zodiac FX* lo cual genera tráfico y la escritura de tablas de flujo de datos al realizarse ping se indica **error en la transmisión. Error general**, como se observa en la **Fig. 44.** El error ocurre debido a que se realiza este paso en una red *SDN* que aún no ha sido programada.

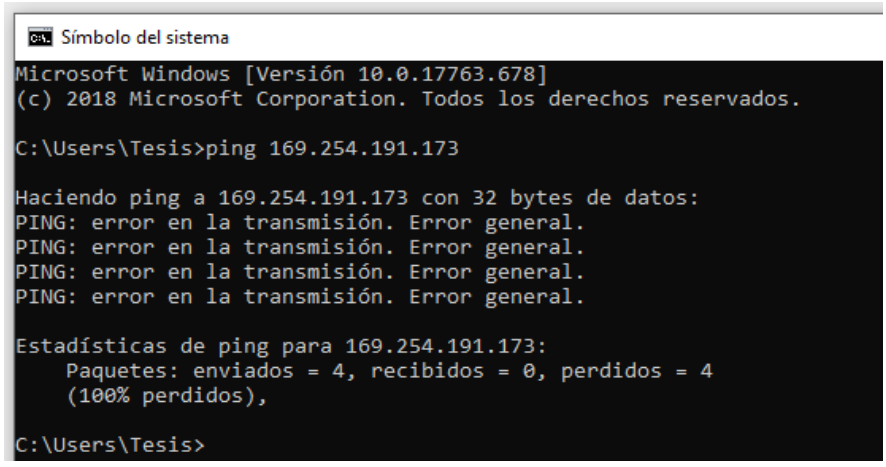

Fig. 44 **Error en la transmisión [Fuente propia]**

Al conectar el puerto # 3 a internet se visualizan los dispositivos dentro de la red y al conectar en el puerto # 2 de la tarjeta *Zodiac FX* la otra *Raspberry Pi* que actua como segundo host se obtiene la dirección *IP* asignada por la tarjeta *Zodiac FX.* En la **Fig. 45** se demuestra la topología de red formada.

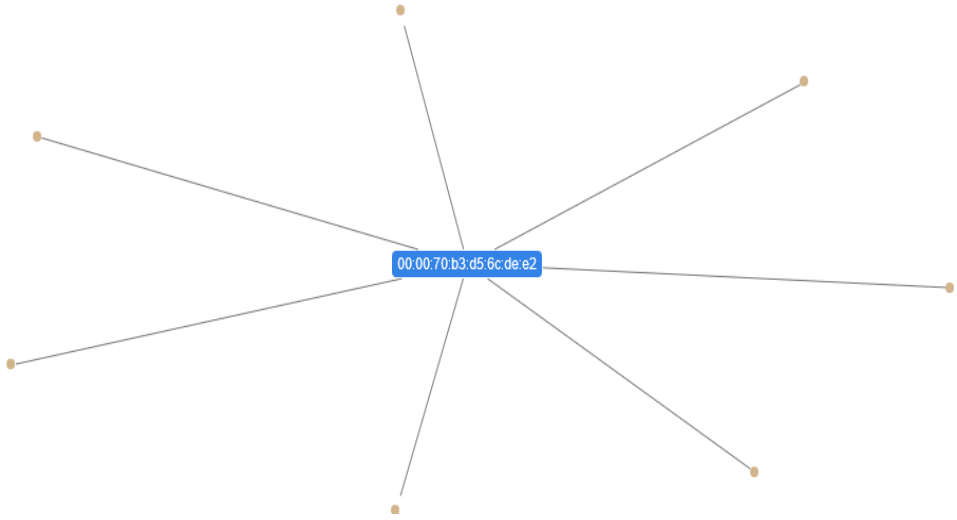

Fig. 45 **Topología de red [Fuente propia]**

Para obtener comunicación entre ambas Raspberry Pi que actúan como host al estar conectadas en los puertos # 1 y 2 se configuran los flujos con la aplicación Flow Maker, esto se lo observa en la **Fig. 46**.

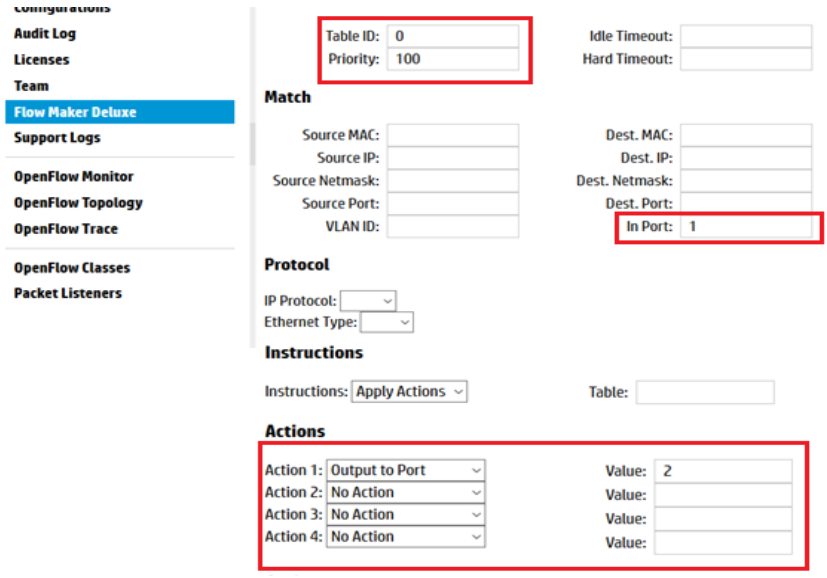

Fig. 46 **Configuración de tabla de flujos de datos [Fuente propia]**

Al crearse la configuración de comunicación desde el puerto 1 hacia el puerto 2 no se puede realizar el intercambio de mensajes y se muestra que el destino al que se hace ping es inalcanzable por lo cual se crea otra regla de configuración, pero en el sentido contrario ahora desde el puerto 2 al puerto 1.

Obteniendo como resultado la comunicación entre ambos dispositivos y enviando mensajes *ICMP.*

4.6 Análisis y resultado de la práctica # 6

Debido a que el controlador *OpenDaylight* no es compatible con la arquitectura *ARM*, arquitectura sobre la cual trabajan los procesadores de las tarjetas *Raspberries Pi*, el controlador se lo ejecuta en una máquina virtual con una imagen *ISO* de Ubuntu 16.04

o cualquier otra compatible, después, hay que configurar las interfaces de red para proceder con la instalación del controlador.

Para la debida y correcta configuración e instalación del controlador *OpenDaylight*, se deben descargar varios archivos de Python que son necesarios para el adecuado funcionamiento y no tener ningún inconveniente al momento de trabajar con este. Después se deberá cambiar el estado en línea de *SSH* e iniciar el servicio *SSH*. Una vez hecho todo lo anterior, se debe instalar Java ya que OpenDaylight funciona con Java, pero el repositorio fue actualizado y no se podrá agregar Java de la manera convencional, por eso se agregó Java Development Kit 8 (*JDK* 8). Ya instalado el controlador *OpenDaylight* se deben agregar ciertos complementos, como D-Lux que provee la interfaz web para el controlador, entre otros (esto lo podremos observar en el **Anexo F**).

En esta práctica se pueden observar las topologías creadas por la conexión física de las *Raspberries Pi* con la tarjeta *Zodiac FX* hacia el dispositivo donde se ejecuta el controlador como también la topología que se crea automáticamente con *Mininet*, esto se puede visualizar en la interfaz web del controlador *OpenDaylight* como también en la interfaz web de *OpenFlow Manager*.

Con la aplicación *OpenFlow Manager* se puede hacer una configuración minuciosa de las tablas de flujos de datos para el controlador, como por ejemplo la prioridad de las tablas de flujo, ID de la tabla de flujo, etc.

4.7 Análisis y resultado de la práctica # 7

Para esta práctica se utiliza *Mininet* para simular los hosts, pero también se puede hacer uso de las tarjetas *Raspberry Pi* como los hosts. Para el debido funcionamiento del controlador *SDN Ryu*, se deben descargar las librerías de *Python* (revisar el **Anexo G**) y también clonar la aplicación Firewall de *Ryu* para poder realizar dicha práctica.

Se pueden utilizar las versiones del sistema operativo raspbian strech 2017 y 2019, las cuales están disponibles para su descarga en el repositorio acordado, pero por cuestiones de compatibilidad y estabilidad se instaló la versión de 2017. En cada host se debe de realizar una configuración para su debida identificación al momento de crear las reglas, como también para la comunicación entre los hosts.

De igual forma que en la práctica # 4 (**Anexo D**), el propósito de ésta es crear y eliminar reglas que permiten la comunicación entre los hosts. Debido al uso de Mininet, hay posibilidad de crear reglas en los diferentes tipos de topologías que Mininet es capaz de simular, además de establecer la versión de *OpenFlow* a utilizarse.

4.8 Análisis y resultados de la práctica # 8

En esta práctica se configuraron 3 dispositivos *Raspberry Pi* y un mini *switch* conocido como *Zodiac FX*. La *Raspberry Pi* que funciona como servidor se configuró con el sistema operativo “raspbian strech”, específicamente la versión de 2017 ya que

es una de las versiones más estables que hay hasta ahora, después de ingresar los comandos para la instalación del controlador Ryu (**Anexo H**).

Los otros dispositivos *Raspberry Pi* pueden tener cualquier otra versión de raspbian, la tarjeta *Zodiac FX* debe de estar con su firmware más reciente y, por su puesto, tener la dirección *IP* de la tarjeta *Zodiac FX* que será el servidor.

El propósito de esa instalación es emplear la aplicación “Flow manager”, la cual nos permite ingresar flujos, métricas y tablas para que haya comunicación entre los dispositivos conectados en la tarjeta *Zodiac FX*. En la **Fig. 47** se muestra la interfaz web de la aplicación Flow Manager:

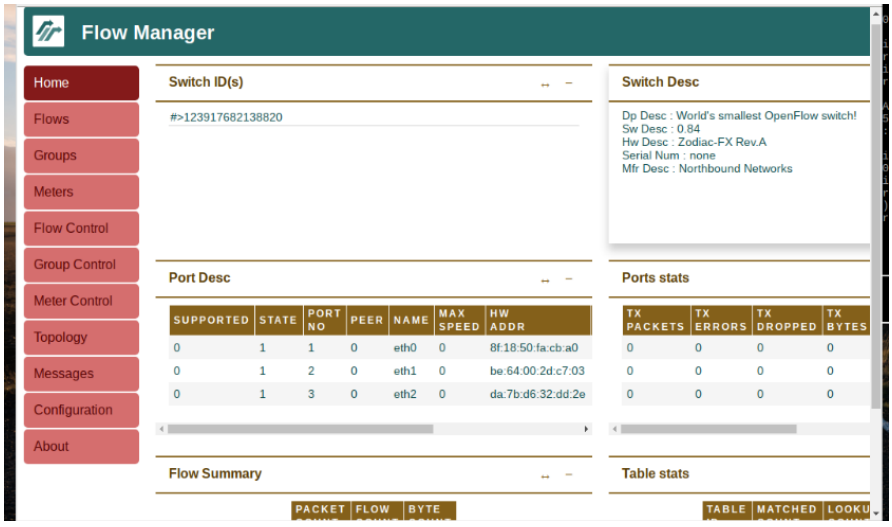

Fig. 47 **Flow Manager [Fuente propia]**

Para que la conexión se efectúe de manera idónea, se tiene que utilizar otra aplicación predeterminada del controlador *Ryu*, esta aplicación se llama “Simple Switch”

(podremos observar la forma de ejecución en el apartado de anexos), antes de ejecutar dicho comando, los otros dispositivos deben de estar conectados, ya que una vez que se ejecute el comando deben aparecer los datos de los hosts en la interfaz web de *Flow Manager*. La **Fig. 48** presenta la topología de red con dos dispositivos conectados a la tarjeta *Zodiac FX*.

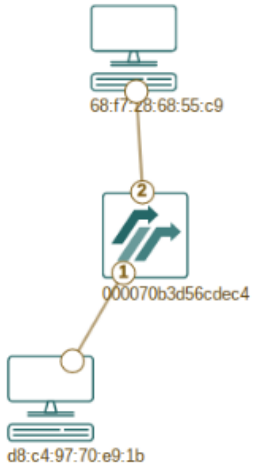

Fig. 48 **Topología de red en Flow manager [Fuente propia]**

Luego de realizar los pasos detallados en el **Anexo H** se puede comprobar que si hay comunicación entre los hosts haciendo ping entre ellos y ver los paquetes enviados y recibidos por los puertos de la tarjeta *Zodiac FX* en la interfaz gráfica web de *Zodiac FX* como se visualiza en la **Fig. 49**.

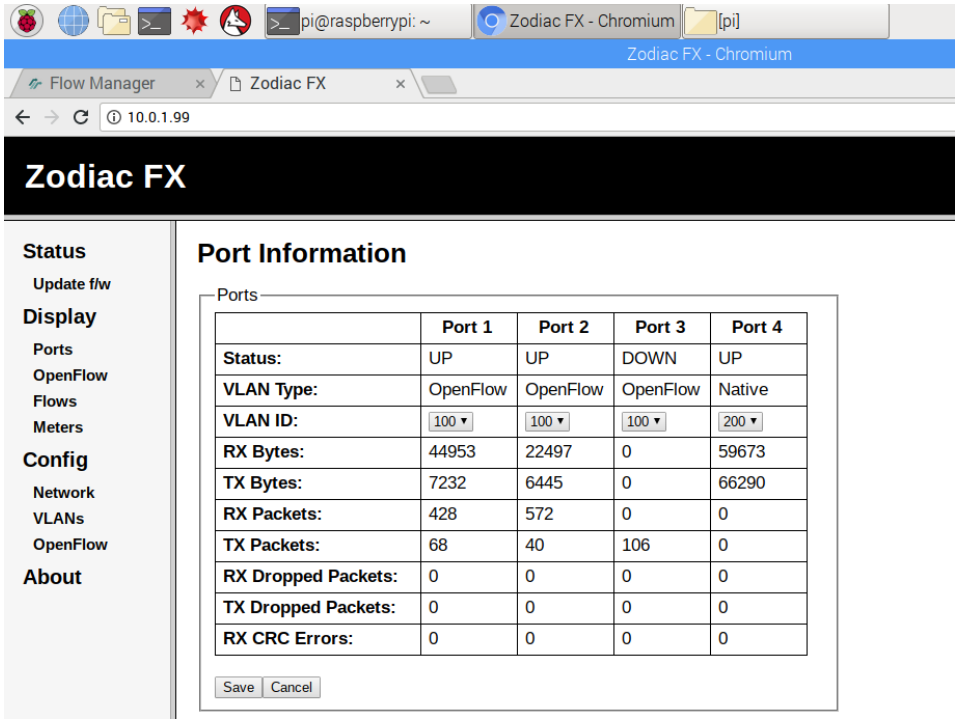

Fig. 49 **Información de puertos de la tarjeta Zodiac Fx [Fuente propia]**

4.9 Análisis y resultados de la práctica # 9

Para la práctica número nueve se realiza la implementación de un sistema de detección de intrusos utilizando el programa *Snort* junto con el controlador *Ryu*. Como primer paso luego de haber realizado la configuración del controlador en prácticas anteriores se debe configurar una nueva interfaz de red en la *Raspberry Pi* en la cual se ha ejecutado previamente el controlador *Ryu*.

Se crea esta nueva interfaz de red dentro del rango de direcciones *IP* en la cual se encuentra configurado el controlador, por lo tanto:

- eth0 es la interfaz de red en la cual se especifica la dirección *IP* del controlador.

- eth1 es la interfaz de red en la cual se especifica la dirección *IP* que se utiliza para establecer la comunicación entre el controlador *Ryu* y el *IDS Snort.*

Se deben establecer las interfaces de red para cada host. En la **Fig. 50** se detalla la topología a implementar:

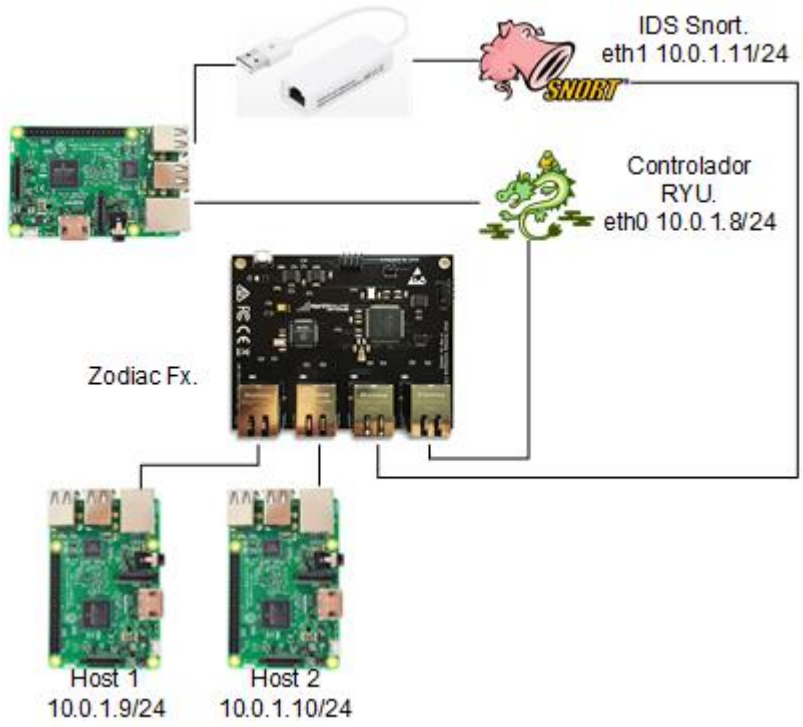

Fig. 50 **Topología a implementar [Fuente propia]**

Es importante recalcar que para el correcto funcionamiento de *Snort* se deben instalar ciertos programas, dependencias y librerías que son necesarias para obtener una integración eficiente entre el controlador *SDN Ryu* y el *IDS Snort*.

Snort funciona con un lenguaje de creación de reglas el cual permite monitorear en tiempo real el tráfico, para la interpretación de esas reglas se necesita instalar *Bison* y *Flex* estos programas funcionan en conjunto:

- *Bison*: Como un generador de analizadores sintácticos (compiladores).

- *Flex*: Como una herramienta que reconoce patrones léxicos en un fichero.

Instalar las dependencias de *Snort* como libpcap-dev que permite capturar paquetes en la capa de red, libdumbnet-dev una librería de red y libpcre3-dev para admitir expresiones regulares este paquete contiene los archivos de desarrollo, encabezados y documentación.

Luego de instalar Snort entre sus configuraciones se agrega la red que es monitoreada por el IDS. En la **Fig. 51** y **Fig. 52** se logra detallar las configuraciones a realizar:

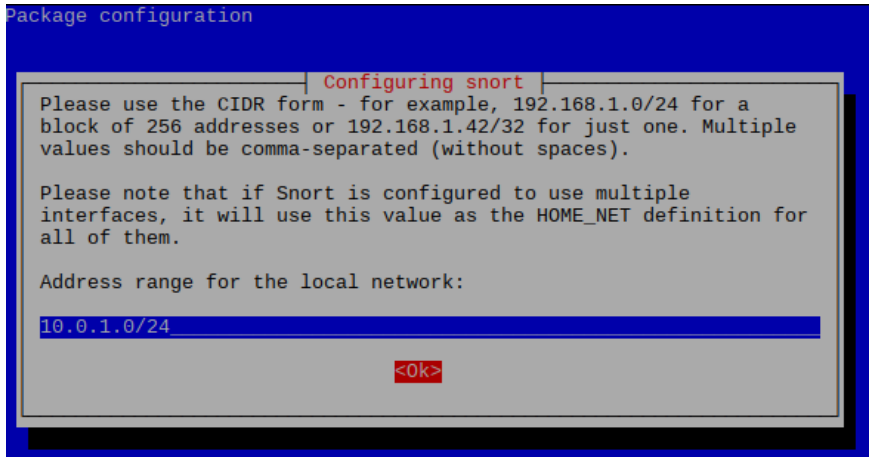

Fig. 51 **Configuración de la red monitoreada [Fuente propia]**

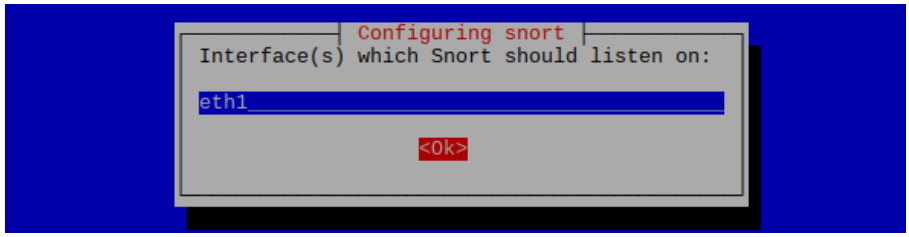

Fig. 52 **Configuración de la interfaz [Fuente propia]**

Se utiliza una aplicación intermediaria de nombre *Pigrelay*, su función es hacer de puente entre *Snort* y el controlador. Escuchando los mensajes de alerta y enviándolos al controlador *Ryu* usando el socket de red.

Se debe crear un archivo de reglas basados en la estructura de reglas para *Snort*, en el **Anexo I** se pueden visualizar todos los pasos a seguir. En la **Fig. 53** tenemos el archivo con las reglas creadas.

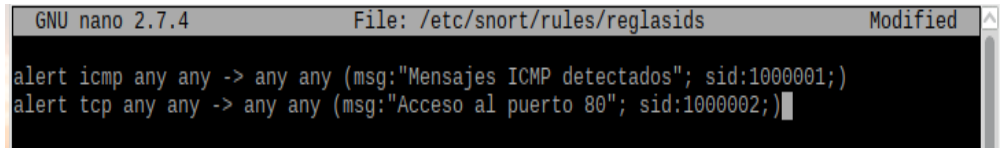

Fig. 53 **Archivo de reglas [Fuente propia]**

Las creaciones de ambas reglas indican cuando se esté intentando acceder al puerto 80 y también detectar cuando se está realizando la transmisión de mensajes *ICMP* o peticiones PING desde otro host.

Se configura el archivo del controlador *Ryu* para que ejecute:

1) Importación de librerías.
2) Inicio de la aplicación.
3) Definición del controlador.
4) Definición de variables.
5) Alerta de paquetes entrantes hacía el controlador.
6) Configuración del Switch.
7) Captura de la dirección MAC para evitar inundación.
8) Instalación de flujos para evitar un paquete entrante.

En el **Anexo I** encontramos el paso a paso para la integración del controlador *Ryu* y *Snort*, la importación de librerías, definición de variables, alerta de paquetes entrantes hacía el controlador, la configuración del *switch*, instalaciones de flujos, etc.

En la **Fig. 54** Se muestra el inicio del controlador *Ryu* junto con la aplicación Flow Manager, se visualizan los mensajes del inicio exitoso del controlador.

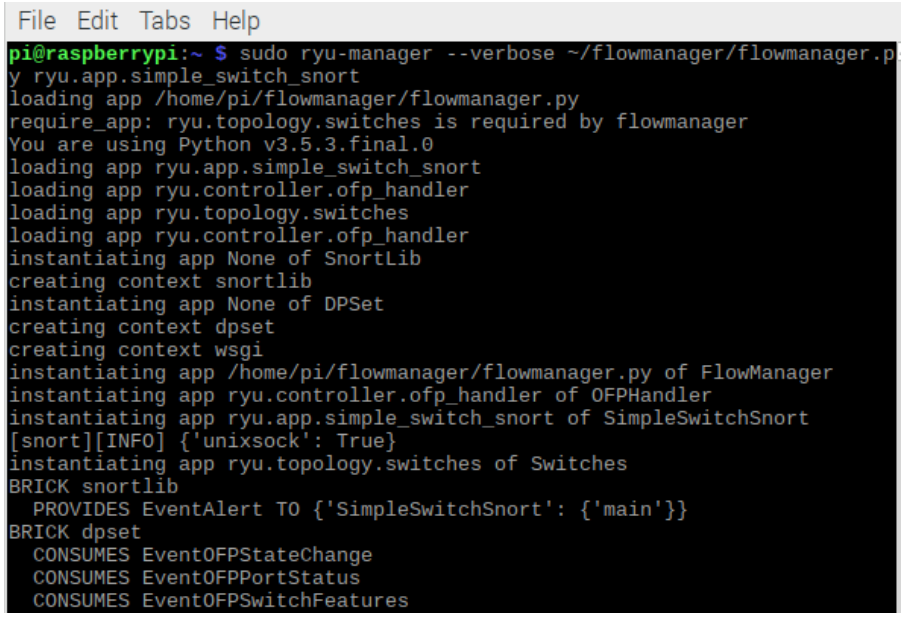

Fig. 54 **Inicio del controlador [Fuente propia]**

Seguir los pasos dados en el **Anexo I** para inicializar el controlador Snort, como se observa en **la Fig. 55**.

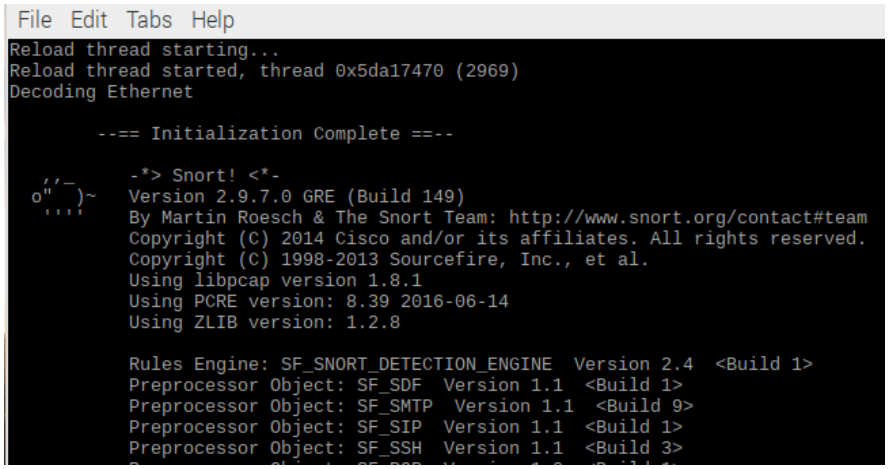

Fig. 55 **Inicio de Snort [Fuente propia]**

Teniendo en cuenta los hosts conectados a la tarjeta *Zodiac FX* que tienen interfaces de red configuradas, a estas Raspberry Pi se les asignaron direcciones *IP* dentro del rango de la red **10.0.1.0/24**:

- Raspberry Pi host #1= 10.0.1.9/24
- Raspberry Pi host #2= 10.0.1.10/24

Como resultado de la práctica se comprueba el correcto funcionamiento e integración del controlador *Ryu* con *Snort*, se realiza el envío de mensajes *ICMP* desde la *Raspberry Pi* que actúa como host # 2 hacia la *Raspberry Pi* que actúa como host # 1, como se

tiene en la **Fig. 56**; se recalca que este procedimiento se puede realizar desde cualquiera de los hosts conectados a la red a través de la tarjeta *Zodiac FX.*

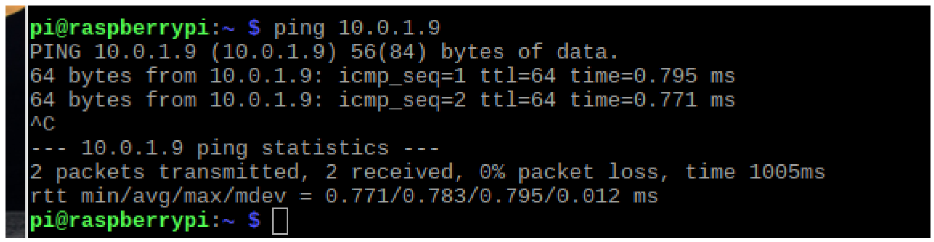

Fig. 56 **Envío de mensajes ICMP [Fuente propia]**

En la **Fig. 57** se observa un mensaje que denota que la conexión de pigrelay con el controlador *Ryu* ha sido un éxito.

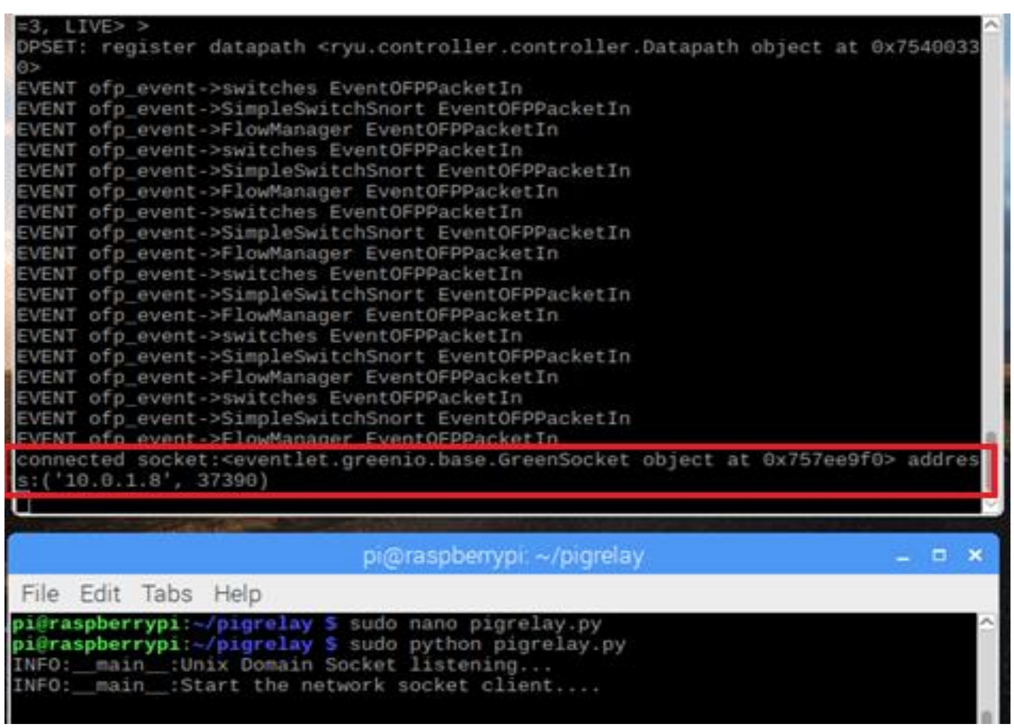

Fig. 57 **Conexión de pigrelay [Fuente propia]**

También se obtiene como resultado la visualización del mensaje configurado en la regla de detección de mensajes *ICMP* en el controlador *Ryu,* junto con una serie de alerta de eventos; todo esto se lo tiene plasmado en la **Fig. 58**. Los mensajes *ICMP* que se presentan son los siguientes:

- **src**: Dirección *IP* de la host fuente.

- **dst**: Dirección *IP* del host destino.

Fig. 58 **Detección de alerta y mensajes ICMP [Fuente propia]**

4.10 Análisis y resultados de la práctica # 10

Se puede realizar esta práctica ya sea con las versiones de raspbian stretch 2017 o 2019, no habría ningún problema al respecto con las versiones de éstas.

Los objetivos de esta práctica son dos en especial, la primera es crear la ruta de datos con *Open vSwitch* y la segunda es la creación de las tablas de flujo de datos mientras se hace la conexión con el controlador *Faucet*, pero para ello primero se realizan diversas configuraciones en Open vSwitch para la creación de su ruta de datos y esta ruta se la podrá observar creada en la interfaz web de *Grafana* (para información técnica y detallada revisar el **Anexo J**). Lo crucial en esta práctica es la debida configuración del archivo faucet.yaml que permite la creación de la ruta de datos tanto en *Open vSwitch* como con la tarjeta *Zodiac FX*, la ventaja es que con la tarjeta *Zodiac FX* la ruta de datos se crea automáticamente mientras que con la configuración de *Open vSwitch* hay que agregar diversos comando para la creación de la ruta de datos.

En la **Fig. 59** se tiene un paso sumamente importante al momento de configurar el archivo faucet.yaml, ya que esto permite verificar si la configuración se encuentra correctamente efectuada.

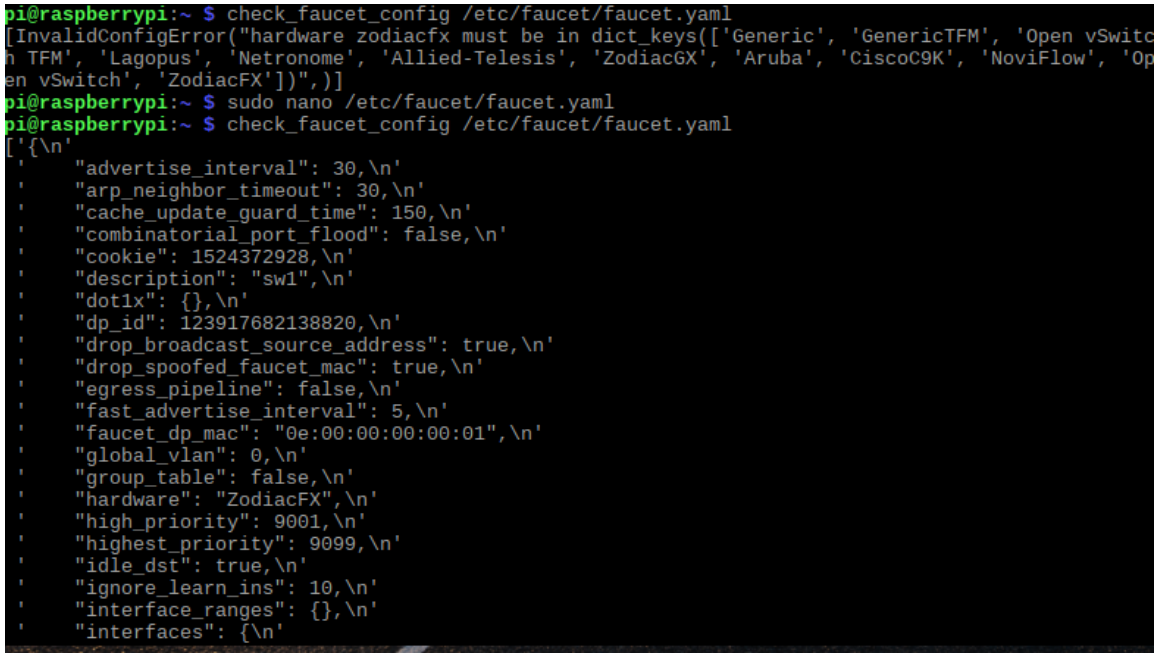

Fig. 59 **Verificación del archivo faucet.yaml [Fuente propia]**

Ya con toda la configuración establecida, también se pueden realizar diversas topologías para ir probando y experimentando, con *Open vSwitch*, pero en este caso lo importante es realizar la correcta conexión para que la ruta de datos se cree utilizando la tarjeta *Zodiac FX*.

Ya con la tarjeta conectada, para comprobar que haya comunicación se puede hacer ping de un host a otro y luego revisar el archivo de logs de *Faucet* para comprobarlo, siempre hay que revisar ese archivo para tener más información de lo que sucede con la conexión de los dispositivos.

5 Conclusiones y Recomendaciones

Conclusiones:

- Se cumplió con el objetivo principal del proyecto de titulación, pues se realizó el diseño de un módulo didáctico de red definida por software (SDN) para prácticas universitarias mediante el uso de hardware libre, el cual permite evaluar el comportamiento de los controladores *SDN* tales como *POX*, *Floodlight*, *HP VAN SDN CONTROLLER*, *OpenDaylight*, *Ryu* y *Faucet*.

- Pox es un controlador de lenguaje simple y con curva de aprendizaje fácil, pero de pocas funcionalidades debido a que solamente es compatible con la versión 1.0 de *OpenFlow*, esto se convierte en un factor limitante para su uso y calificación como controlador *SDN* ideal debido la falta de soporte de *OpenFlow* 1.3.

- Al utilizar Mininet se puede crear una red virtual a través de la interfaz de línea de comando en cuestión de segundos, esta red puede ser personalizada y desplegarse en un ambiente de producción sobre hardware real. Convirtiendo a Mininet en un emulador adecuado para ser utilizado en ámbitos de desarrollo, enseñanza e investigación.

- La arquitectura *ARM* que poseen los procesadores Broadcom modelo *ARMv8* implementados en las tarjetas *Raspberry Pi* 3 Modelo B no es compatible con el controlador *SDN OpenDaylight*, imposibilitando la instalación del controlador en la tarjeta mencionada.

- El controlador *Ryu* es compatible con las versiones 1.0, 1.2, 1.3, 1.4 y 1.5 además está completamente escrito sobre el lenguaje de programación *Python* lo cual hace que la curva de aprendizaje sea media, pero de manipulación sencilla.

- El controlador *Ryu* es opensource (de código abierto) por lo cual se encuentra de manera gratuita disponible bajo la licencia de apache 2.0 y existe una comunidad de usuarios que aportan cada vez más información sobre este framework expandiendo así la documentación sobre controladores *SDN*. Y también existe una comunidad de desarrolladores que crean aplicaciones y que facilitan el acceso a ellas como es el caso de Pigrelay que permite la integración entre el controlador *Ryu* y el detector de intrusos *Snort.*

- Durante el desarrollo de este proyecto se ha obtenido la exitosa implementación del módulo didáctico utilizando *OpenFlow* como principal protocolo, para establecer la comunicación entre dispositivos y controladores en cada práctica. Lo cual también ha permitido conocer el funcionamiento de este protocolo.

Recomendaciones:

- Concientizar en no retirar los dos jumpers que se encuentran colocados en los pines Master y Erase de la tarjeta *Zodiac FX*, ya que, si los jumpers se retiran de los pines mencionados, se eliminará el firmware con el que funciona la tarjeta y no se podrá programar de forma adecuada.

- Descargar e instalar los programas y aplicaciones subidos en el repositorio para su uso, cada uno de los programas fue revisado para verificar la compatibilidad con cada controlador *SDN* y correcto funcionamiento en el desarrollo de las prácticas.

- Tener en cuenta como requisito los conocimientos básicos sobre los sistemas operativos *GNU/Linux* en este caso *Ubuntu* y *Raspbian* para el desarrollo de las prácticas y control de la arquitectura de red.

- Instalar sobre una máquina virtual la imagen *ISO* de *Ubuntu* para poder realizar la ejecución del controlador *OpenDaylight*.

- Tomar en consideración que cuando se trabaje con el controlador Faucet se debe configurar de forma correcta el archivo faucet.yaml, ya sea que se desee trabajar con *Open vSwitch* o con la tarjeta *Zodiac FX*.

- Realizar un seguimiento de las alertas generadas por el controlador *Ryu* y el detector de intrusos *Snort* para que cuando sean almacenadas en una base de datos efectuar una revisión y comparación de las características antiguas y nuevas que poseen los ataques de red detectados y evitar que estos ataques resurgen.

- Desplegar una *SDN* con dispositivos de red programables a través de aplicaciones de software, en un entorno real y con auditorías de seguridad.

Referencias bibliográficas

[1]https://www.ibm.com/support/knowledgecenter/es/SSNE44_5.2.0/com.ibm.tpc_V52.doc/fqz0_c_service_class_config.html (Consultado el 8 de Septiembre del 2019)

[2] http://yuba.stanford.edu/cs244wiki/index.php/Overview (Consultado el 14 de Mayo de 2019)

[3] Search Data Center “Redes Definidas por Software (SDN).” https://searchdatacenter.techtarget.com/es/definicion/Redes-definidas-por-software-SDN (Consultado el 28 de Enero de 2019)

[4] https://www.aboutespanol.com/que-es-un-router-841387 (Consultado el 8 de Septiembre del 2019)

[5] https://www.aboutespanol.com/que-es-un-switch-841388 (Consultado el 8 de Septiembre del 2019)

[6] http://www.informeticplus.com/que-es-WiFi (Consultado el 8 de Septiembre del 2019)

[7] https://www.masadelante.com/faqs/tcp-ip (Consultado el 8 de Septiembre del 2019)

[8] https://www.speedcheck.org/es/wiki/ipv4/ (Consultado el 8 de Septiembre del 2019)

[9] http://www.maestrosdelweb.com/evolucionando-hacia-el-ipv6/ (Consultado el 8 de Septiembre del 2019)

[10] SDX Central, “Zodiac FX“ https://www.sdxcentral.com/products/zodiac-fx/ (Consultado el 28 de Enero de 2019)

[11] OSRG “Build SDN Agilely” https://osrg.github.io/Ryu/ (Consultado el 1 de Febrero de 2019)

[12] https://blog.desdelinux.net/opendaylight-el-futuro-de-las-redes-definidas-por-software-sdn/ (Consultado el 8 de Septiembre del 2019)

[13] https://faucet.nz/ (Consultado el 8 de Septiembre del 2019)

[14] https://prezi.com/enxvi_tvhxkf/sdn/ (Consultado el 8 de Septiembre del 2019)

[15] Project Floodlight, “Floodlight” http://www.projectFloodlight.org/Floodlight/ (Consultado el 28 de Enero de 2019)

[16] https://github.com/conapps/Devops-101/blob/master/Advanced%20Network%20Programmability/dia_1/3_netconf.md (Consultado el 9 de Septiembre del 2019)

[17] http://www.servervoip.com/blog/softswitch/ (Consultado el 10 de Septiembre del 2019)

[18] https://northboundnetworks.com/blogs/sdn/what-is-open-vswitch (Consultado el 10 de Septiembre del 2019)

[19] Virtualizadesdezero.com, “¿Qué es Openstack y por qué deberías saber de su existencia?” [Online]. Disponible: https://virtualizadesdezero.com/que-es-openstack/ (Consultado el 14 de Septiembre del 2019)

[20] Red hat, “¿Qué es el cloud computing?” [Online]. Disponible: https://www.redhat.com/es/topics/cloud (Consultado el 14 de Septiembre del 2019)

[21] https://prezi.com/dnhv55kp9tsb/que-es-virtual-box-y-para-que-sirve/ (Consultado el 20 de Septiembre del 2019)

[22] https://pandorafms.com/blog/es/que-es-grafana/ (Consultado el 24 de Enero del 2020)

[23] https://gauge.org/ (Consultado el 27 de Enero del 2020)

[24] https://aptira.com/monitoring-your-infrastructure-with-prometheus/ (Consultado el 27 de Enero del 2020)

[25] https://docs.openstack.org/keystone/latest/ (Consultado el 24 de Enero del 2020)

[26] https://github.com/CiscoDevNet/OpenDaylight-Openflow-App (Consultado el 22 de Enero del 2020)

[27] https://pandorafms.com/blog/es/que-es-snmp/ (Consultado el 9 de Septiembre del 2019)

[28] RUIZ, Andrés, “Redes Definidas por Software”, http://ribuc.ucp,edu,co:8080/jspui/bitstream/handle/10785/2788/CDMIST97.pdf?sequence=2 (Consultado el 25 de Enero de 2019)

[29] https://www.researchgate.net/figure/Traditional-Network-versus-SDN_fig1_319876305 (Consultado el 27 de Enero de 2019)

[30] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14-76, Jan 2015 (Consultado el 27 de Enero de 2019)

[31] FEAMSTER, REXFORD, ZEGURA, “The road to SDN: An Intellectual History of Programmable Networks”, https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf (Consultado el 28 de Enero de 2019)

[32] FOUKAS, MARINA, KONTOVASILIS, “Software Defined Networking Concepts” https://homepages.inf.ed.ac.uk/mmarina/papers/sdn-chapter.pdf (Consultado el 28 de Enero de 2019)

[33] FIGUEROLA, Norberto, “Redes Definidas por Software” https://articulosit.files.wordpress.com/2013/10/sdn.pdf (Consultado el 28 de Enero de 2019)

[34] http://www.networkcomputing.com/networking/7-essentials-of-software-defined-networking/d/d-id/898899 (Consultado el 28 de Enero de 2019)

[35] OPEN NETWORKING FOUNDATION, “SDN Architecture Overview” https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0pdf (Consultado el 28 de Enero de 2019)

[36] Internet Research Task Force IRTF, “Software-Defined Networking (SDN): Layers andArchitecture Terminology”, https://tools.ietf.org/html/rfc7426 (Consultado el 28 de Enero de 2019)

[37] MILLAN, Ramon, “SDN: El future de las redes inteligentes” http://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php (Consultado el 28 de Enero de 2019)

[38] OPEN NETWORKING FOUNDATION, “Software-Defined Networking (SDN) Definition”, https://www.opennetworking.org/sdn-resources/sdn-definition (Consultado el 28 de Enero de 2019)

[39] Mininet, “Mininet, an instant virtual network on your laptop (or other PC)” http://mininet.org/ (Consultado el 28 de Enero de 2019)

[40] Universidad Católica de Pereira “Mininet: una herramienta versátil para emulación y prototipado de Redes Definidas por Software.”

http://biblioteca.ucp.edu.co/OJS/index.php/entrecei/article/view/2492 (Consultado el 1 de Febrero de 2019)

[41] Open Networking “Mininet” https://www.opennetworking.org/mininet/ (Consultado el 1 de Febrero de 2019)

[42] Kick Starter “Zodiac FX: The world's smallest OpenFlow SDN switch.” https://www.kickstarter.com/projects/northboundnetworks/zodiac-fx-the-worlds-smallest-OpenFlow-sdn-switch?lang=es (Consultado el 1 de Febrero de 2019)

[43] SDX Central “What is Floodlight Controller?” https://www.sdxcentral.com/networking/sdn/definitions/what-is-Floodlight-controller/ (Consultado el 1 de Febrero de 2019)

[44] ABC Tecnología “¿Qué es Raspberry Pi 3 Model B y para qué sirve?” https://www.abc.es/tecnologia/informatica-hardware/20130716/abci-raspberry-como-201307151936.html (Consultado el 1 de Febrero de 2019)

[45] SDX Central, “What is Ryu Controller?” https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-Ryu-controller/ (Consultado el 28 de Enero de 2019)

[46] Tinajero, Edwin, “Implementación de un prototipo de switch OpenFlow de bajo costo utilizando una Raspberry Pi 3 Model B” http://bibdigital.epn.edu.ec/bitstream/15000/16737/1/CD-7333.pdf (Consultado el 17 de Mayo del 2019)

[47] http://sdnhub.org/tutorials/Ryu/ (Consultado el 18 de Mayo del 2019)

[48] Y .-D. Lin, “OpenFlow Versión Roadmap,” 2015. (Consultado el 18 de Mayo del 2019)

[49] OPEN NETWORKING FOUNDATION, “OpenFlow Switch Specification” https://www.opennetworking.org/images/stories/downloads/sdn-resorces/onf-specifications/OpenFlow/OpenFlow-switch-v1.3.4.pdf (Consultado el 14 de Mayo de 2019)

[50] N. McKeown et al., “OpenFlow,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, p. 69, Mar. 2008 (Consultado el 18 de Mayo del 2019)

[51] ARCHETE, Carlos “Despliegue de una maqueta de red basada en OpenFlow”, http://repositorio.unican.es/xmlui/bitstream/handle/10902/4484/Carlos%20Arteche%20Gonzalez.pdf?sequence=1 (Consultado el 14 de Mayo de 2019)

[52] MALDONADO. Diego “Diseño e implementación de una aplicación de red bajo la arquitectura SDN”, http://repository.javeriana.edu.co/bitstream/10554/12745/1/MaldonadoHidaldoDiegoArmando2014.pdf (Consultado el 16 de Mayo de 2019)

[53] OPEN NETWORKING FOUNDATION, “OpenFlow Switch Specification”, http://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf (Consultado el 16 de Mayo de 2019)

[54] https://github.com/CiscoDevNet/OpenDaylight-OpenFlow-App (Consultado el 14 de Septiembre del 2019)

[55] https://github.com/martimy/flowmanager (Consultado el 14 de Septiembre del 2019)

[56] https://www.androidpit.es/android-comprar-tarjeta-sd (Consultado el 14 de Septiembre del 2019)

[57] El androide libre, “Todo lo que necesitas saber sobre las tarjetas microSD” [Online]. Disponible: https://elandroidelibre.elespanol.com/2017/01/tarjetas-microsd-datos-problemas-informacion.html (Consultado el 14 de Septiembre del 2019)

[58] Tecnología fácil, “¿Qué es un cable Ethernet?” [Online]. Disponible: https://tecnologia-facil.com/que-es/que-es-un-cable-ethernet/ (Consultado el 14 de Septiembre del 2019)

ANEXOS

Anexo A

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 1

RECONOCIMIENTO DE HARDWARE Y SOFTWARE A UTILIZAR.

Objetivos:

- Identificar los dispositivos y programas que se utilizarán en la implementación del módulo.
- Aprender a establecer las configuraciones en la tarjeta *Zodiac FX*.

Instrucciones:

- Descargar los archivos y topologías correspondientes a cada práctica del repositorio de MEGA https://mega.nz/folder/39thDaqa#KitCcutD5cZR7WzLDlxiyw
- Configurar las opciones ideales para ejecutar un controlador sobre una máquina virtual.

- Realizar el flasheo (grabación) de un archivo de imagen en una tarjeta SD para trabajar con las *Raspberry Pi 3 Model B*.

Desarrollo:

Se utilizan varios controladores, la mayoría de estos funcionan sobre distribuciones Linux, se utilizará el programa **Virtual Box** para virtualizar versiones estables del sistema operativo Ubuntu.

Descargar la versión disponible del programa *Virtual Box* y su archivo de extensión del repositorio. Luego de haber descargado el programa *Virtual Box* se debe instalar con permisos de administrador al finalizar la instalación ejecutar el archivo de extensión. Este archivo de extensión permite que el programa *Virtual Box* ejecute archivos *OVA* exportados, en otras palabras, archivos de máquinas virtuales que han sido exportados con detalles y configuraciones específicas tal como el archivo *OVA* del controlador *Floodlight* que se utilizará en prácticas posteriores, se indica que aquella máquina virtual fue configurada con requerimientos especiales incluyendo la arquitectura Java sobre la cual trabaja el controlador y debe ser configurada de una manera específica para su correcto funcionamiento.

Para iniciar la configuración de una máquina virtual se indican los siguientes pasos a seguir, como se observa en la **Fig. 1**:

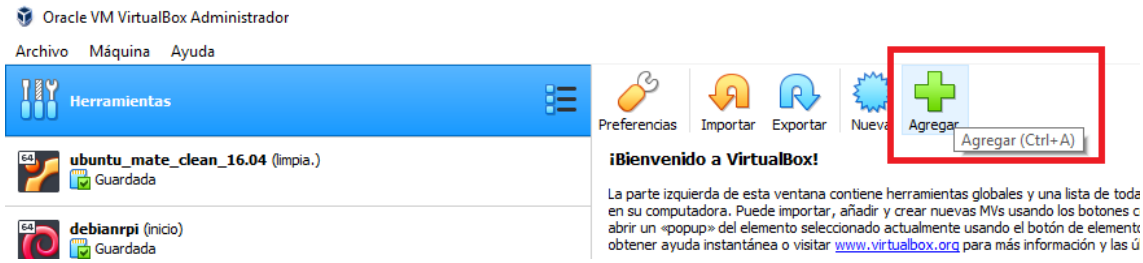

Fig. 1 **Creación de una máquina virtual [Fuente propia]**

En la **Fig. 2** se establece la arquitectura y versión sobre la cuales funciona la máquina virtual.

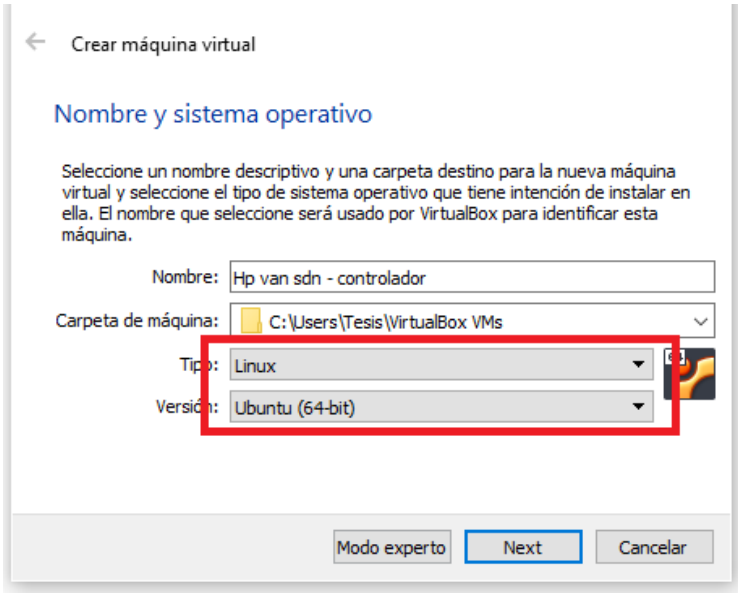

Fig. 2 **Tipo y versión de la máquina virtual [Fuente propia]**

En la **Fig. 3** se elige el tamaño de la memoria *RAM*, se sugiere utilizar de 2 Gigabytes *RAM* en adelante para obtener un correcto funcionamiento.

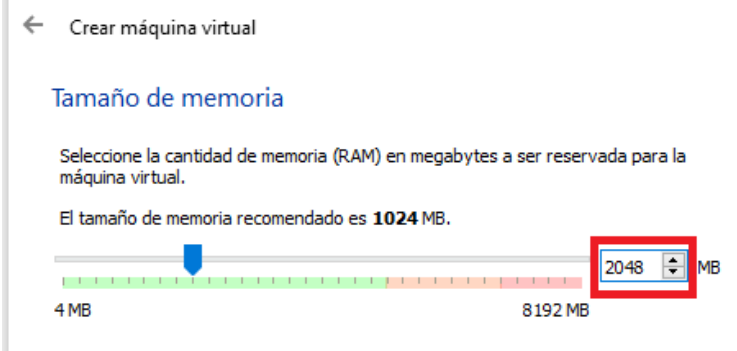

Fig. 3 **Tamaño de memoria de la máquina virtual [Fuente propia]**

La siguiente opción se refiere al disco duro, se debe elegir la opción "**crear un disco duro virtual ahora"**, esta opción permite instalar los sistemas operativos a virtualizar utilizando los recursos propios del equipo. Y elegir la opción VDI. La infraestructura de escritorio virtual (VDI) es una técnica de virtualización que permite el acceso a un escritorio virtualizado, que está alojado en un servicio remoto a través de Internet. Se refiere al software, hardware y otros recursos necesarios para la virtualización de un sistema de escritorio estándar. Todo esto lo tenemos en la **Fig. 4**.

VDI también se conoce como una interfaz de escritorio virtual.

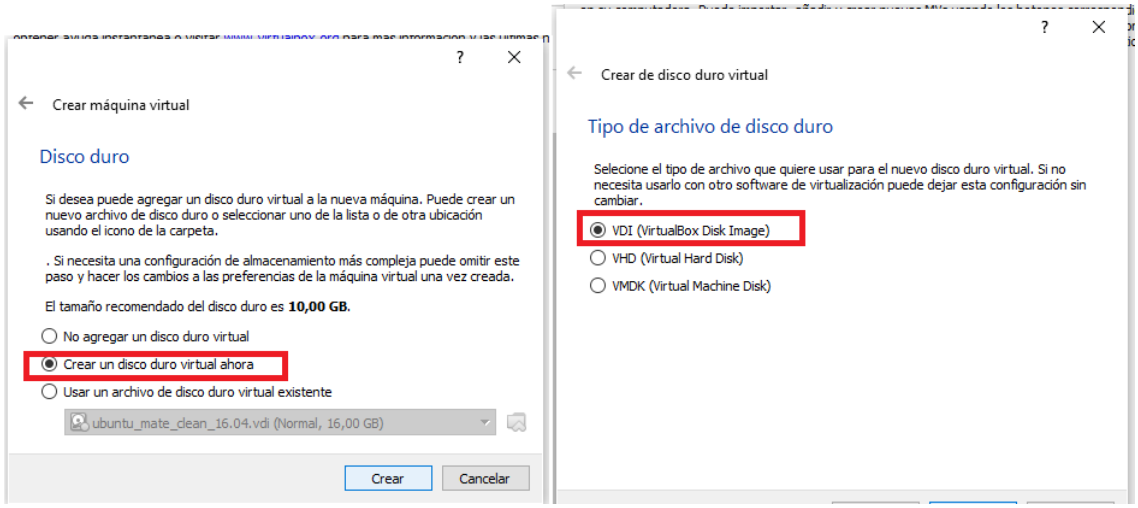

Fig. 4 **Disco de la máquina virtual [Fuente propia]**

Elegir la opción **reservado dinámicamente** se elige esta opción en caso de tener que redimensionar el disco en un futuro. Seleccionar la ubicación y tamaño del disco duro, se puede elegir lo que se considere adecuado, como se presenta en la **Fig. 5**.

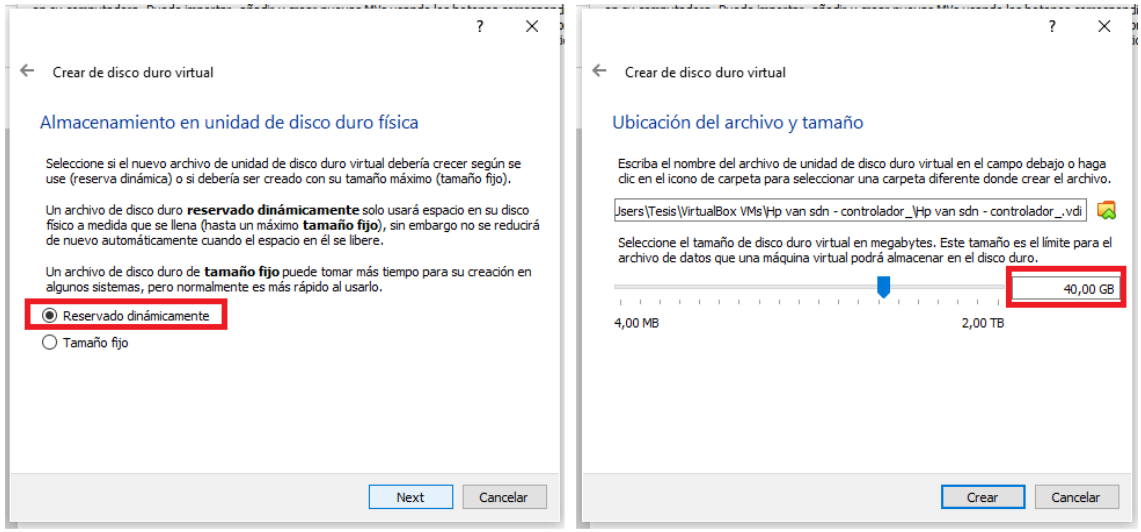

Fig. 5 **Almacenamiento y ubicación de la máquina virtual [Fuente propia]**

Posteriormente elegir el archivo a virtualizar siguiendo los pasos dados en la **Fig. 6**.

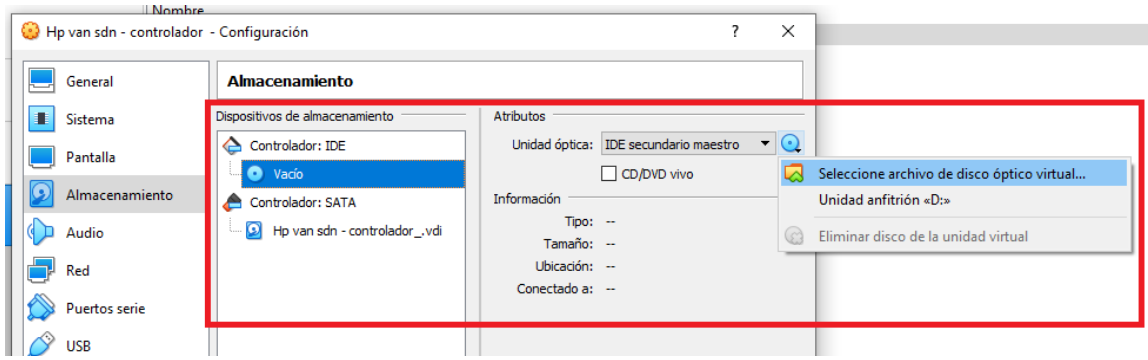

***Fig. 6* Selección de dispositivo de almacenamiento [Fuente propia]**

Se debe grabar el archivo de imagen .img que contiene el sistema operativo Raspbian recomendado para instalarse en la *Raspberry Pi 3 Model B* basado en la distribución de nombre Debian. Para cual se debe realizar el formateo de la tarjeta SD, hay dos modos de realizar este paso:

- El primer método es utilizar la herramienta incorporada en el sistema Windows para realizar el formateo, deseleccionar la opción de **Formato rápido** e iniciar el formateo de la tarjeta.

Esto se lo tiene en la **Fig. 7**.

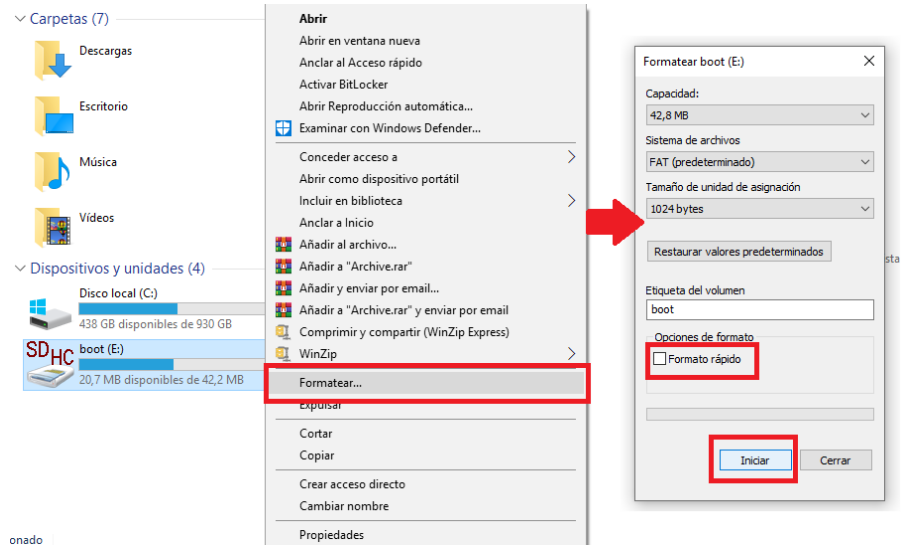

Fig. 7 **Formateo de tarjeta SD [Fuente propia]**

- El segundo método y el más recomendado debido que esta herramienta formatea de una manera específica la tarjeta SD para la grabación de archivos y el correcto formateo de la tarjeta SD, se debe descargar el archivo de instalación del programa SD Card Formatter y luego de que se haya instalado el programa proceder a formatear la tarjeta.

Es importante realizar la correcta selección del dispositivo a formatear en este caso la tarjeta SD, seleccionar la opción **Overwrite Format** e iniciar el formateo, tal cual se visualiza en la **Fig. 8**.

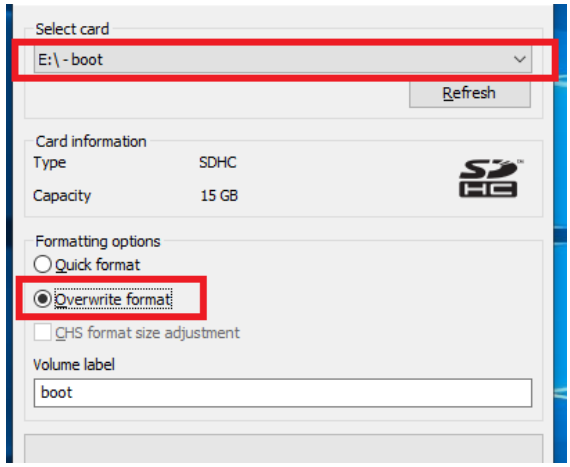

Fig. 8 **Formateo con aplicación SD Card Formatter [Fuente propia]**

Luego de realizar el formateo de la tarjeta SD, abrir el programa Win32 Disk Imager y seleccionar la unidad correspondiente a la tarjeta SD. En la **Fig. 9** se muestra la selección de la imagen Raspbian.

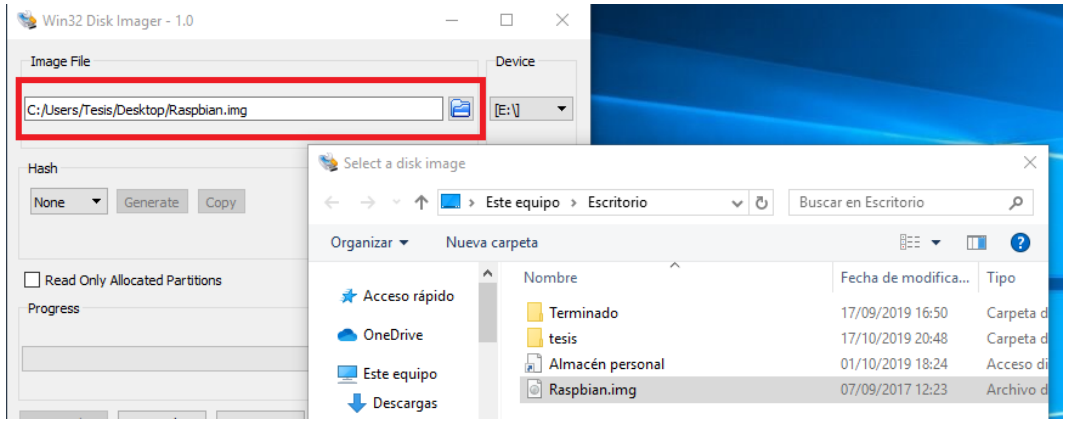

Fig. 9 **Sistema operativo Raspbian [Fuente propia]**

Seleccionar **Write** para grabar la imagen del sistema operativo y luego elegir la opción **Yes** para confirmar la escritura de la tarjeta SD. Estos pasos son mostrados en la **Fig. 10** y en la **Fig. 11** se muestra la finalización del proceso de escritura:

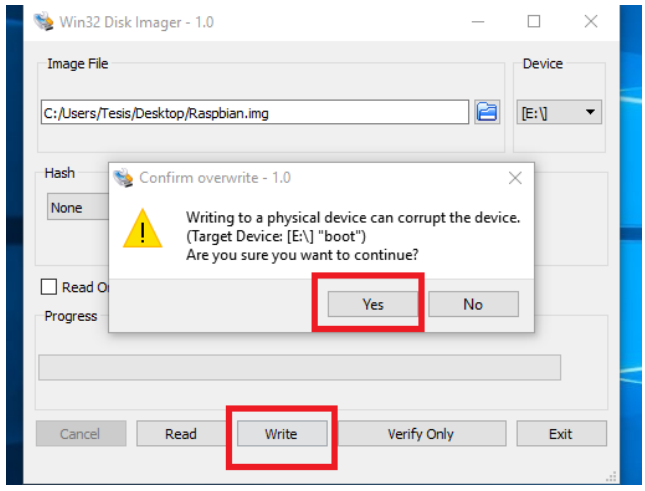

Fig. 10 **Escritura del archivo .img [Fuente propia]**

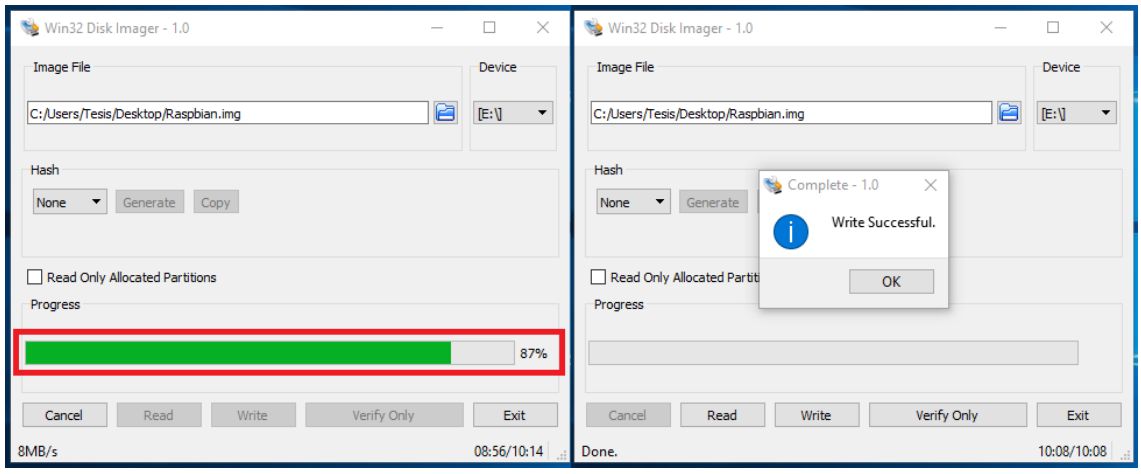

Fig. 11 **Finalización del proceso [Fuente propia].**

Se deben realizar ciertas configuraciones en el sistema Debian instalado en la *Raspberry Pi* como, por ejemplo: la zona horaria, las interfaces necesarias para la práctica y el idioma como se puede observar en las **Fig. 12**, **Fig. 13, Fig. 14**, **Fig. 15** y **Fig. 16.**

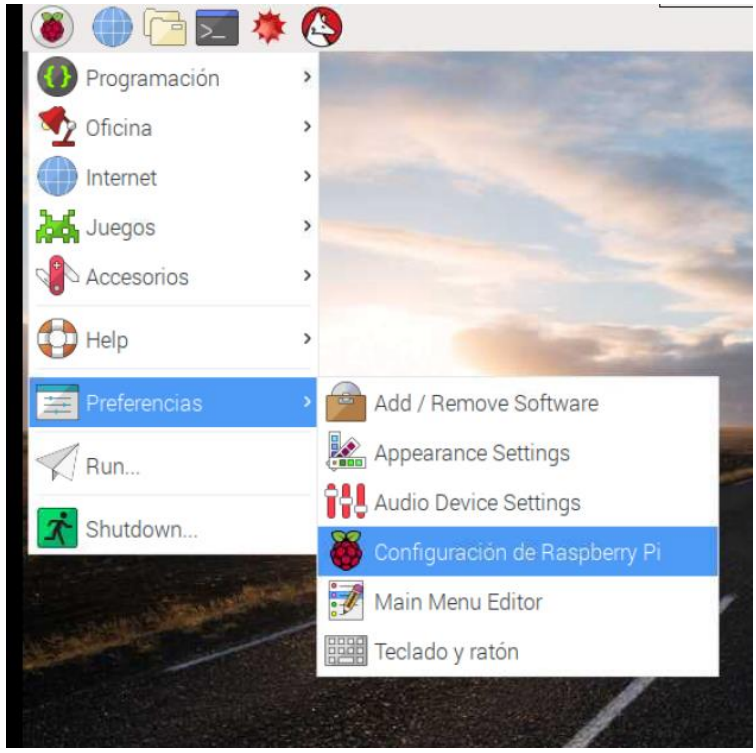

Fig. 12 **Configuración de raspbian [Fuente propia]**

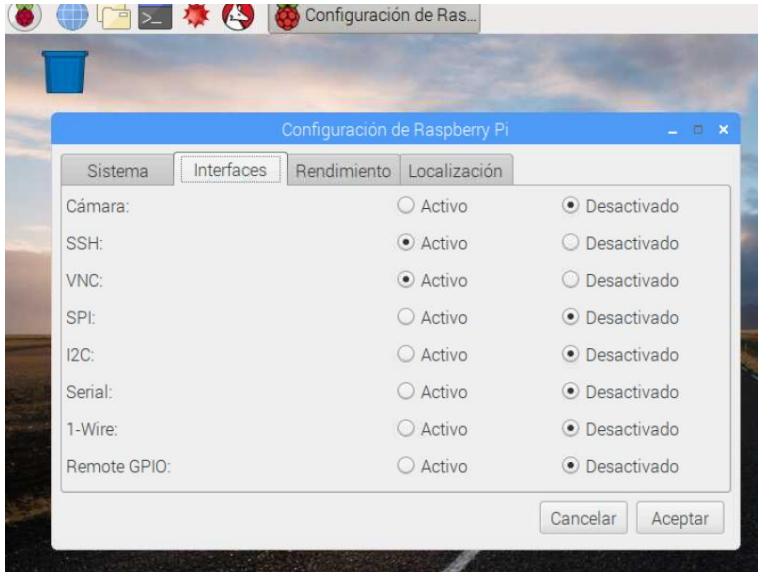

Fig. 13 **Interfaces de raspbian [Fuente propia]**

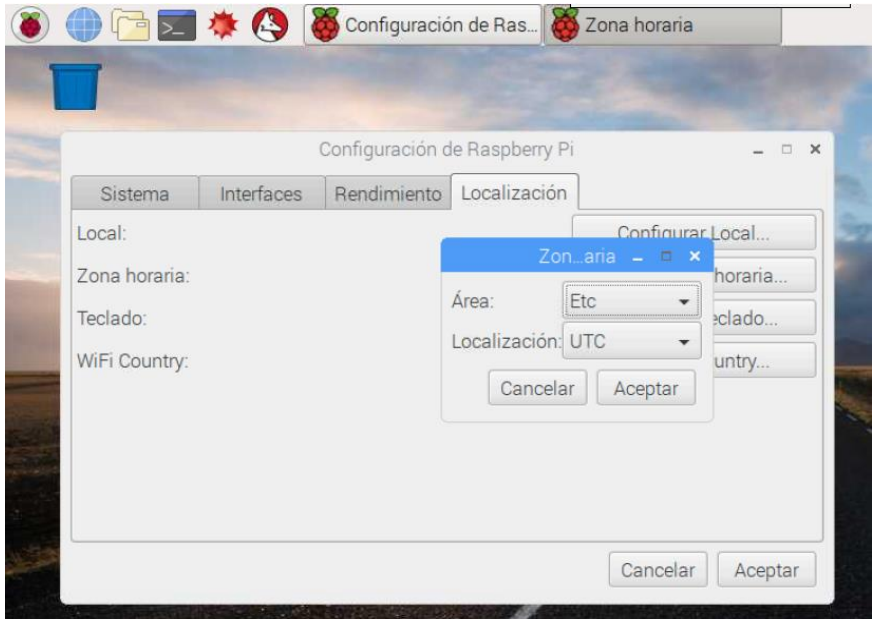

Fig. 14 **Zona horaria [Fuente propia]**

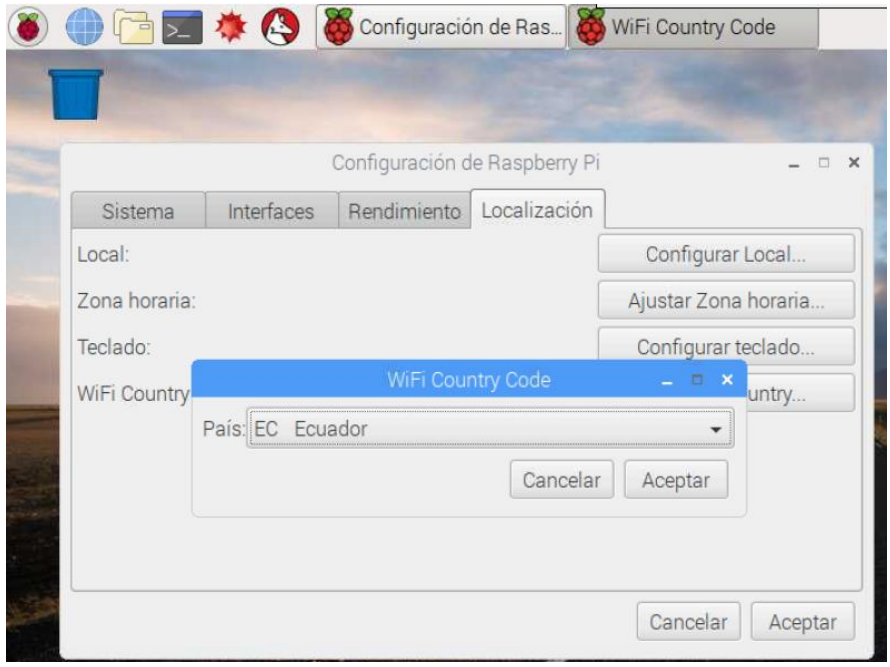

Fig. 15 Código WiFi del país [Fuente Propia]

.

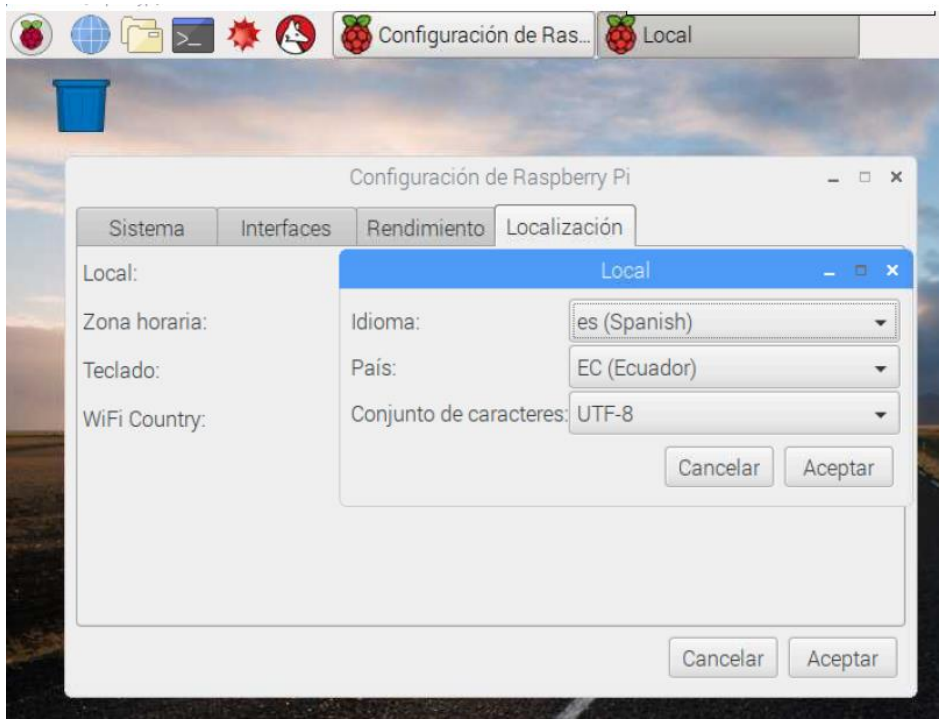

***Fig. 16* Idioma del teclado [Fuente propia]**

Se debe tener presente que en la tarjeta *Zodiac FX* no se deben retirar los jumpers señalados en la **Fig. 17** y **Fig. 18**.

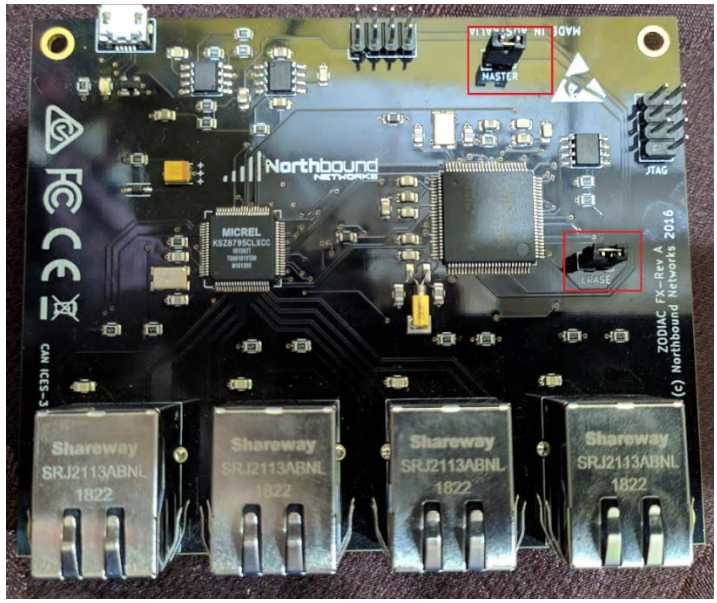

Fig. 17 **Tarjeta Zodiac FX [Fuente propia]**

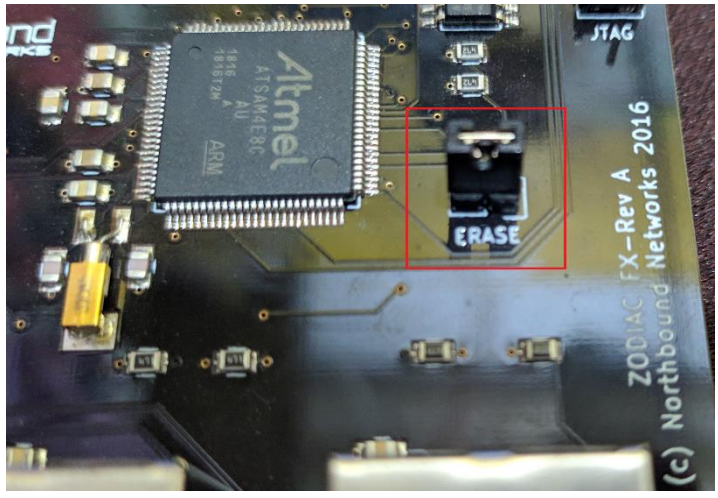

Fig. 18 **Erase de la tarjeta Zodiac FX [Fuente propia]**

Ya que se puede incorporar al módulo una pantalla LCD de 5 pulgadas, para que ésta funcione debemos de ingresar una serie de comandos, los cuales son los siguientes:

- **git clone https://github.com/Elecrow-keen/Elecrow-LCD5.git**
- **cd Elecrow-LCD5/**
- **cd Elecrow-LCD5**
- **chmod +x Elecrow-LCD5**
- **sudo ./Elecrow-LCD5**

En la **Fig. 19** se muestra la funcionalidad del digitalizador de la pantalla LCD de 5 pulgadas.

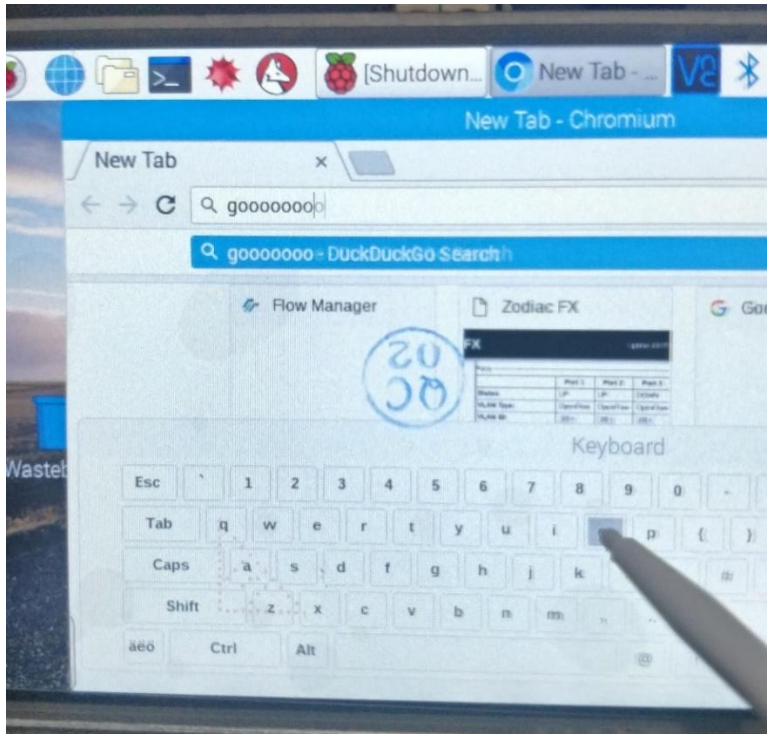

Fig. 19 **Funcionamiento de la pantalla táctil LCD [Fuente propia]**

Para que aparezca el teclado virtual para usar el digitalizador o pantalla táctil, se ingresa la siguiente serie de comandos:

- **sudo apt-get install matchbox-keyboard**
- **cd Desktop**

Se debe de crear un documento en el escritorio llamado teclado.sh y dentro de ese documento ingresar lo siguiente:

- **sudo nano teclado.sh**

- **#!/bin/bash**
- **matchbox-keyboard**
- **cd Desktop**
- **sudo chmod +x teclado.sh**
- **reboot**

Una vez se haya reiniciado la *Raspberry Pi*, saldrá un mensaje de aplastar “Enter” para continuar, damos “Enter”, a continuación, se debe de iniciar la *Raspberry Pi*. Después de aquello debe aparecer un ícono en el escritorio con el nombre “teclado.sh”, dar doble click y ejecutar como se representa en la **Fig. 20**.

Fig. 20 **Teclado virtual de la Raspberry Pi [Fuente propia]**

Anexo B

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 2

BALANCEO DE CARGAS UTILIZANDO UN CONTROLADOR SDN.

Objetivos:

- Aprender a utilizar un controlador SDN como un balanceador de carga.
- Dirigir las solicitudes http de clientes a servidores http predefinidas.

Instrucciones:

- Descargar el archivo de imagen ISO llamado ubuntu-14.04.6-desktop-amd64.iso del repositorio de archivos.
- Realizar la virtualización del sistema operativo Ubuntu versión 14.04 mediante el programa Virtual Box.
- Instalar y configurar Mininet para realizar la conexión entre el controlador POX; los clientes y servidores http.

Desarrollo:

Iniciar la virtualización de la imagen ISO de Ubuntu 14.04 descargada del repositorio, la cual es una versión estable de Ubuntu que permitirá trabajar con el controlador POX y Mininet. Luego iniciar la máquina virtual e iniciar la instalación del sistema operativo Ubuntu virtualizado. En la **Fig. 1** se observa la selección de la máquina virtual, en este caso es Ubuntu.

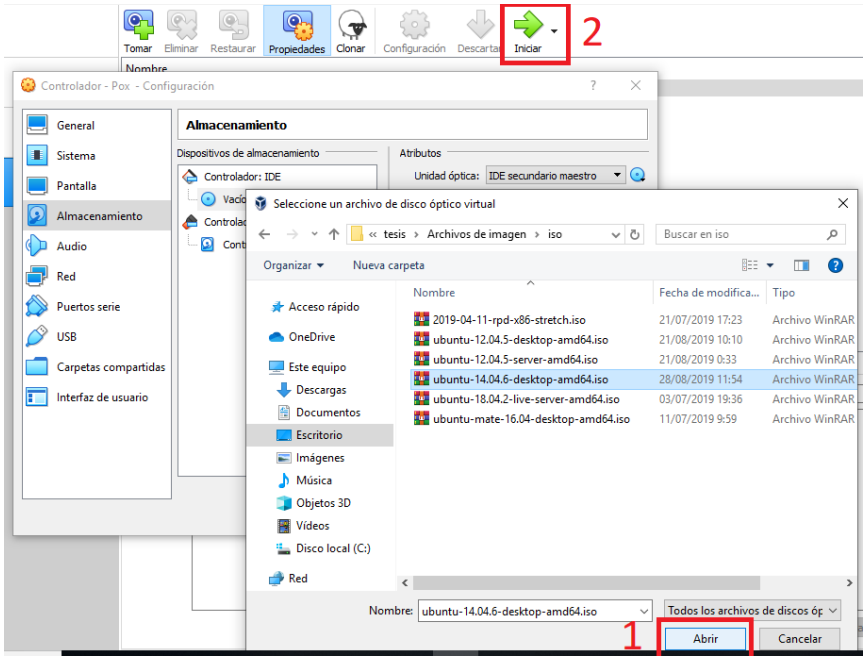

***Fig. 1* Carga de Ubuntu [Fuente propia]**

Crear la máquina virtual y cargar la *ISO* de Ubuntu, una vez hecho esto, se presenta la siguiente pantalla de instalación de Ubuntu, tal cual se presenta en la **Fig. 2**, seleccionar el idioma español y la opción de instalar Ubuntu.

Fig. 2 **Bienvenida de Ubuntu [Fuente propia]**

Deseleccionar la opción "**Descargar actualizaciones mientras se instala"**, para que el proceso de instalación sea más rápido y seleccionar la opción "**Continuar"**. Seguido de esto, seleccionar la opción de "**Borrar disco e instalar Ubuntu"**, dar clic en "**Instalar ahora"** y seleccionar "**Continuar"**. En la **Fig. 3** se tienen las alertas de las modificaciones y las particiones que se procederán a formatear para la instalación de la máquina virtual.

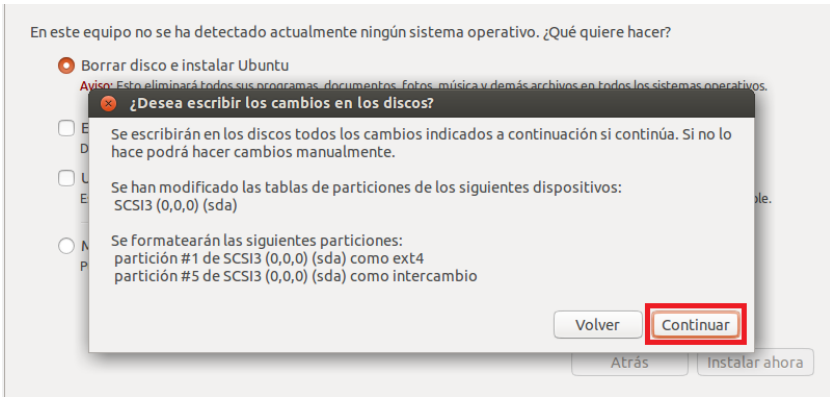

Fig. 3 **Instalación de Ubuntu [Fuente propia]**

Configurar la zona horaria: América – Guayaquil. Luego configurar el teclado a español (latinoamericano). Escribir un nombre para el equipo, nombre de usuario, contraseña. Y elegir la opción “**Iniciar sesión automáticamente”**. Por ejemplo, como se considera en la **Fig. 4**.

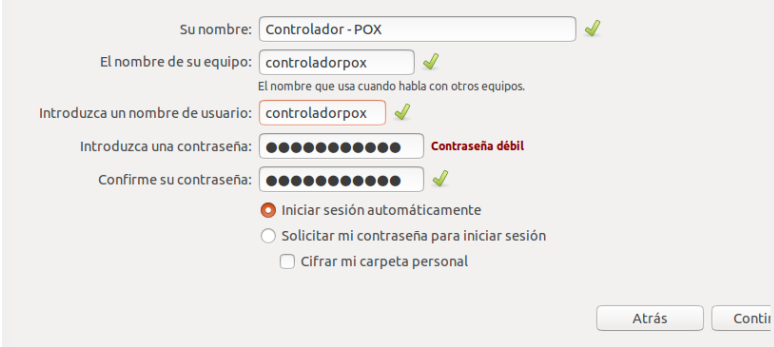

Fig. 4 **Credenciales de la máquina virtual [Fuente propia]**

Una vez realizados estos pasos se debe de iniciar la maquina virtual mostrando el escritorio de ésta, como se visualiza en la **Fig. 5**.

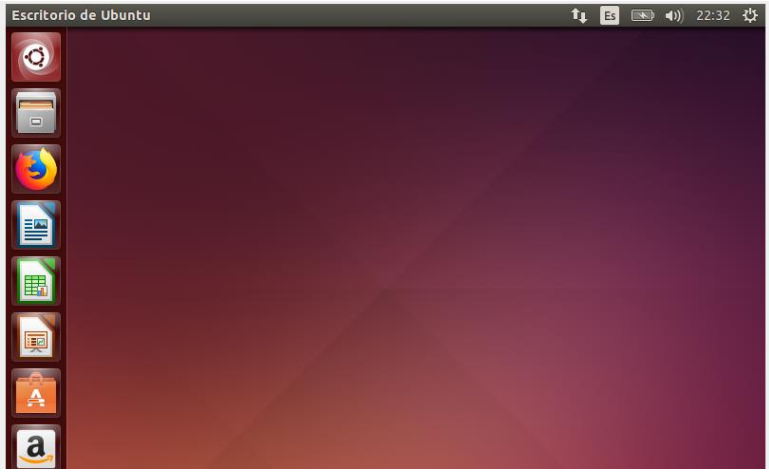

Fig. 5 **Escritorio de Ubuntu [Fuente propia]**

Reiniciar la máquina virtual y abrir la **terminal** de Ubuntu. Por defecto, la cuenta root no está activada en Ubuntu, sin embargo, la cuenta de superusuario o root es la que dispone de permisos más elevados en el sistema lo que en Windows es equivalente al administrador.

Razón por la cual Linux divide la cuenta de superusuario o root de una cuenta normal de usuario. Para ejecutar los permisos de superusuario siempre se debe utilizar el comando **su,** que significa “substitute user” para cambiar un usuario desde la terminal. Como, por ejemplo: **su** {nombre del usuario}.

También se obtienen privilegios de root utilizando el comando **sudo** este comando permite la ejecución de comandos como un usuario común el cual pide la contraseña del usuario en uso, este comando simula ser otro usuario sin cambiar el usuario actual. Ejemplo: **sudo** apt-get install {nombre-del-paquete-a-instalar}.

Estos comandos funcionan durante 15 minutos con permisos de superusuario. Para tener de manera permanente los permisos de superusuario se debe ingresar la combinación de ambos comandos dando como resultado el comando **sudo su**.

En la **Fig. 6** se tiene el ingresa el comando **sudo su** en la terminal para tener los permisos.

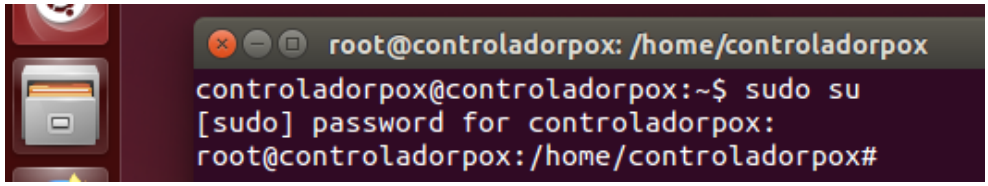

Fig. 6 **Superusuario [Fuente propia]**

Se debe instalar *Mininet* programa que permite la emulación de redes definidas por software permitiendo la creación de topologías de red y conexión a controladores remotos como se realizará en esta práctica (# de practica) con el controlador POX.

Ingresar en la terminal el comando: **apt-get install Mininet**.

La instalación se la tiene en la **Fig. 7**.

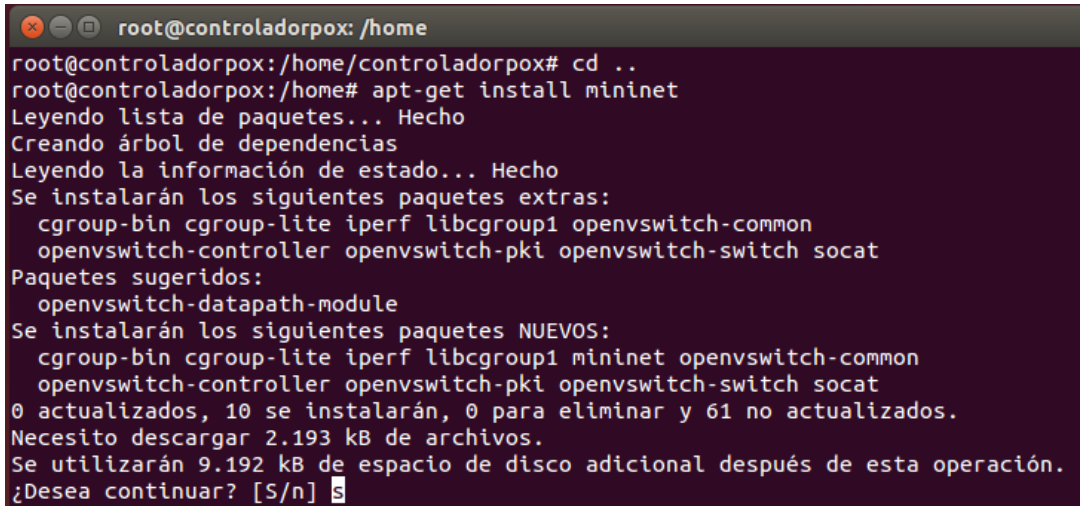

Fig. 7 **Instalación de Mininet [Fuente propia]**

Ingresar el comando **mn –c** para limpiar alguna topología creada, como se muestra en la **Fig. 8**.

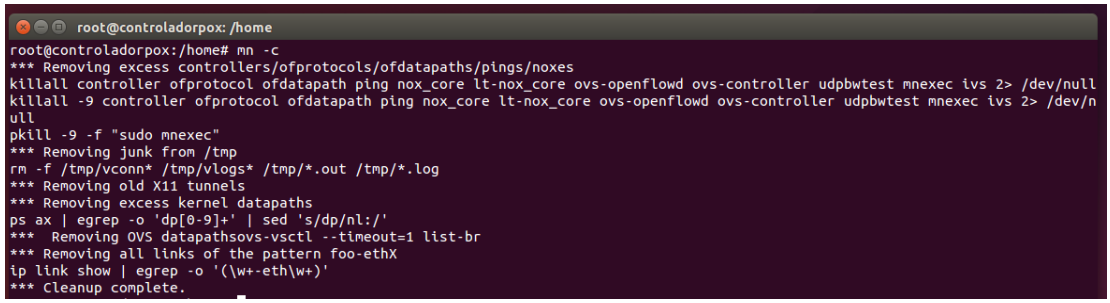

Fig. 8 **Limpieza de topologías [Fuente propia]**

Se debe instalar GitHub para clonar el repositorio de Mininet: **apt-get install git.**

Luego se deben clonar los archivos de Mininet con el comando "**git clone**" con las últimas actualizaciones desde su repositorio, el cual es "git://github.com/Mininet/Mininet". En la **Fig. 9** se demuestra como se procede con la clonación.

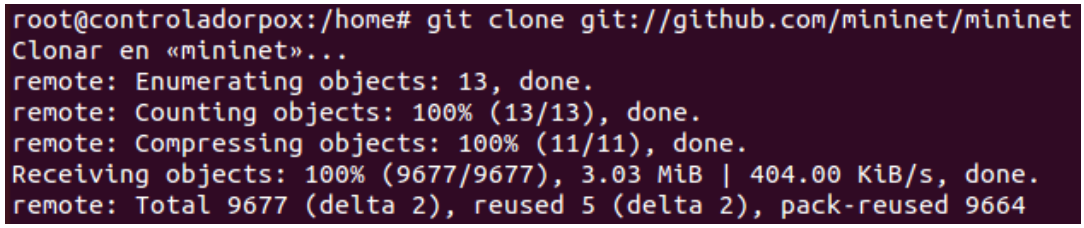

Fig. 9 **Clonación del repositorio de Mininet [Fuente propia].**

Situarse en el directorio de Mininet con el comando: **cd mininet**.

Utilizar el comando **git tag** para visualizar la lista de versiones disponibles de Mininet, como se tiene en la **Fig. 10**.

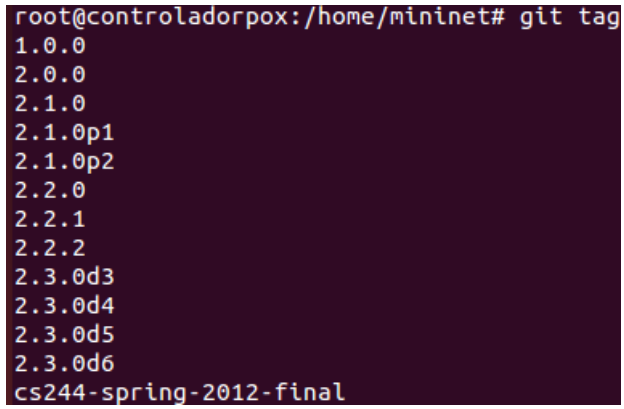

Fig. 10 **Versiones de Mininet [Fuente propia].**

Elegir y cambiar a la última rama o versión de Mininet con el comando **git checkout**. Esto se lo puede apreciar en la **Fig. 11**.

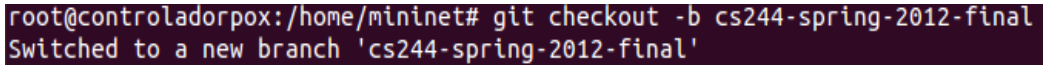

Fig. 11 **Cambio a última versión [Fuente propia].**

Regresar al directorio de raíz con el comando **cd.** Instalar las últimas actualizaciones de Mininet descargadas usando el siguiente comando, como se lo tiene en la **Fig. 12**: **mininet/util/install.sh -a**.

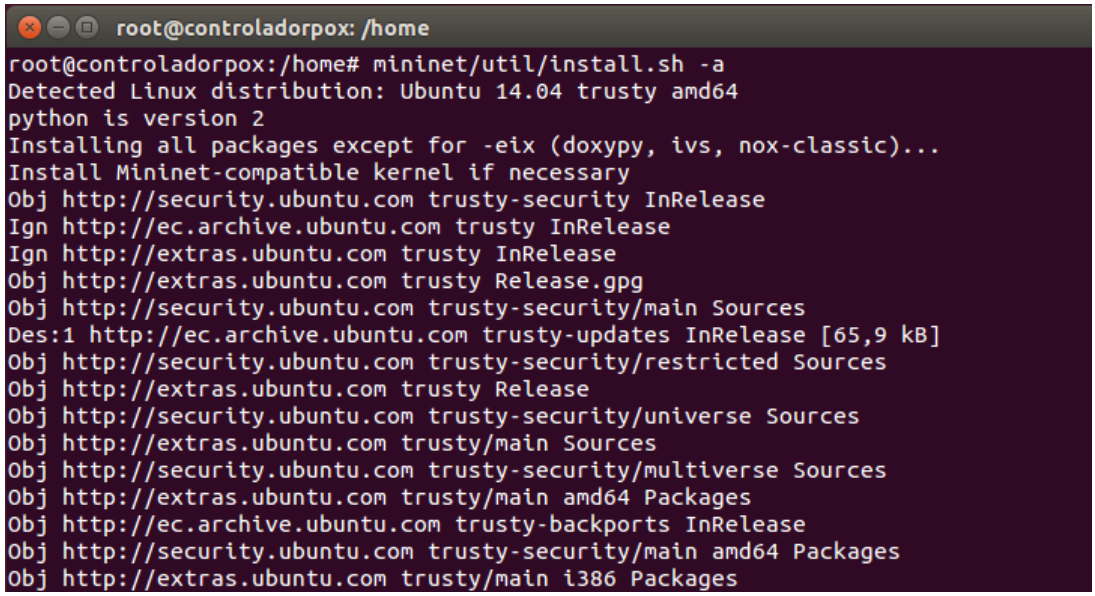

Fig. 12 **Instalación de actualizaciones [Fuente propia]**

En la **Fig. 13** tenemos los directorios en los cuales se encuentra la instalación de *Mininet* y POX

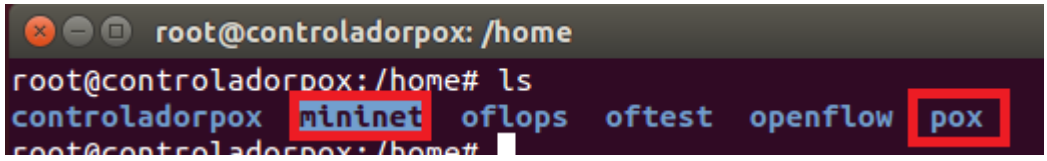

Fig. 13 ***Directorios [Fuente propia]***

Para crear una topología por default o simple en el simulador se utiliza el comando **mn** acompañado del comando **sudo** ya que este comando permite el inicio de Mininet. En la **Fig. 14** se crea una topología simple, la cual está integrada por: 1 *switch*, 1 controlador y 2 hosts.

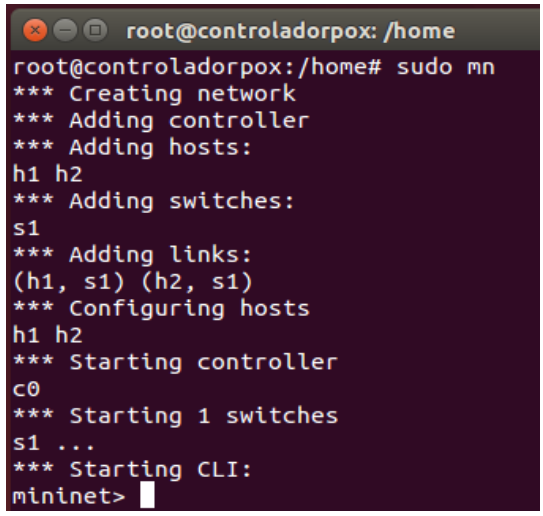

Fig. 14 **Topología simple de Mininet [Fuente propia]**

El comando **dump** sirve para visualizar la información de los dispositivos simulados. En la **Fig. 15** se tienen los datos de dichos dispositivos.

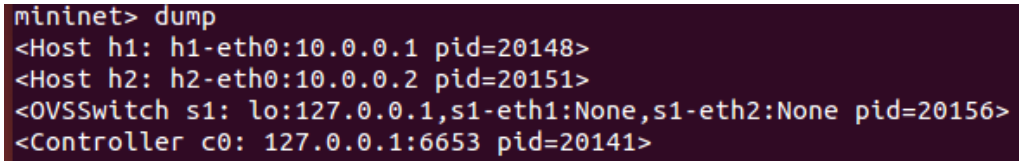

Fig. 15 Información de dispositivos [Fuente propia]

Enviar un paquete *ICMP* o ping entre hosts para comprobar la comunicación, como se tiene en la **Fig. 16**.

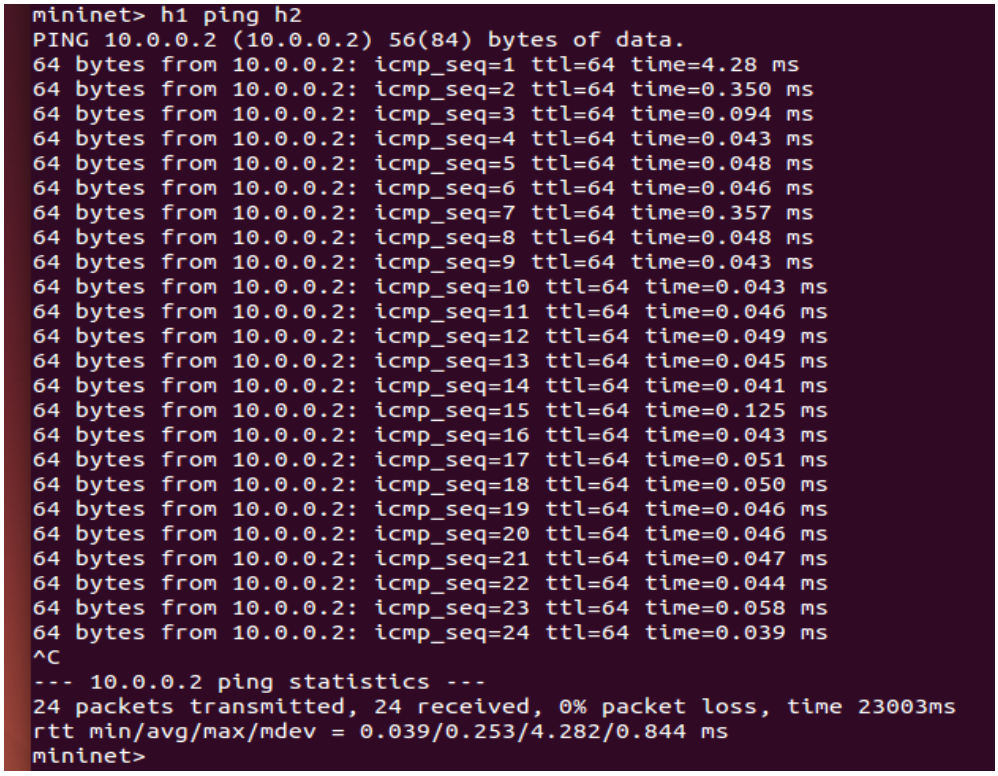

Fig. 16 Comunicación comprobada [Fuente propia]

Salir de Mininet: **exit**.

Ahora se procede a establecer al controlador POX como un balanceador de cargas. En la **Fig. 17** se tiene la topología que se debe seguir cuando el controlador POX está establecido como un balanceador de cargas.

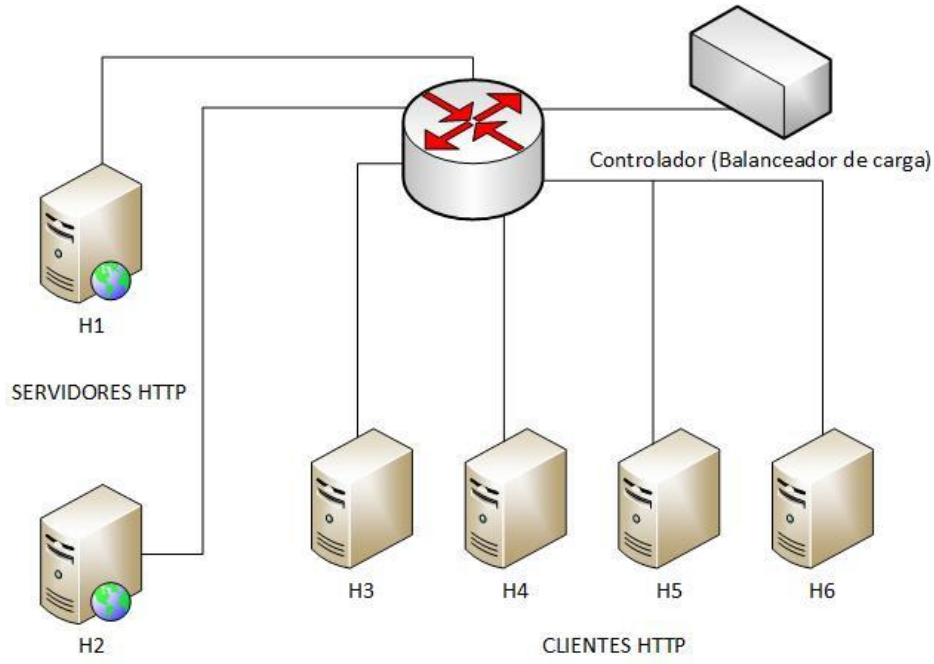

Fig. 17 **Topología de POX como balanceador de carga [Fuente propia]**

Para el balanceador de cargas se debe crear una topología simple que está conformada por 6 hosts, 1 *switch* y un controlador remoto. Estructura del comando a utilizarse:

- **sudo mn**: inicia *Mininet*.

- **--topo single, N**: Se crea un *switch* conectado a N hosts, se indica que N es el número de hosts deseados.

- **--controller=remote, port=xxx**: Inicia el controlador remoto, para esta práctica el controlador elegido es *POX* el cual será conectado en los siguientes pasos y se comunicará a través del puerto especificado.

Se debe utilizar el siguiente comando: **sudo mn --topo single, 6 --controller=remote, port=6633**

Abrir una terminal para el host H1 y H2 que actuaran como servidores *HTTP* con el comando **xterm**.

Iniciar el servidor *HTTP* en los hosts 1 y 2: **python –m SimpleHTTPServer 80**.

Abrir otra terminal de Ubuntu e ingresar como superusuario con el comando **sudo su** y dirigirse al directorio de raíz con **cd**. Luego se debe iniciar el controlador *POX* este controlador ofrece una plataforma para crear aplicaciones de tipo *OpenFlow*. *POX* permite el desarrollo y creación rápida de prototipos de software de control de red. *POX* trabaja con *Python* por lo cual cuenta con un intérprete para su ajuste y depuración llamado py. Estructura del comando a utilizarse:

- **pox/pox.py**: Inicia *POX* mediante su interprete py.

- **log.level**: *POX* utiliza la infraestructura de registro de *Python*. Cada componente tiene sus propios logs (registradores), cuyo nombre se muestra como parte del mensaje de registro. Los registradores o logs en realidad forman una jerarquía: es posible que tenga un log "foo" con un sublogger "bar", que en conjunto se conocería como "foo.bar". Además, cada mensaje de registro tiene un "nivel" asociado, que corresponde a la importancia (o gravedad) del mensaje. El componente log.level le permite configurar que logs muestran el nivel de detalle. Los niveles de registro de más a menos severos son:

- ✓ CRITICAL
- ✓ ERROR
- ✓ WARNING
- ✓ INFO
- ✓ DEBUG

Para este caso se utilizará **--DEBUG** que permite visualizar los problemas o información cuando se inicia el controlador.

misc.ip_loadbalancer: Componente para iniciar el balanceador de cargas. Se debe asignar una dirección *IP* a una página web, en este caso será la dirección *IP* 10.0.1.1 en la cual estará el controlador *POX* y una lista de direcciones *IP* al servidor. Los nuevos flujos *TCP* que son dirigidos a la página web serán redirigidos aleatoriamente a cualquiera de los servidores *HTTP* existentes. Los servidores *HTTP* son revisados periódicamente a través del envío de mensajes ARP para confirmar que se encuentren activos.

- **--ip**: Dirección IP del controlador donde también se aloja la página web a la que se dirigirá tráfico.

- **--servers**: Direcciones IP de los servidores *HTTP*.

Así es como queda el siguiente comando que se debe utilizar para iniciar el controlador POX como un balanceador de cargas, el comando se lo visualiza en la **Fig.**

18: **pox/pox.py log.level --DEBUG misc.ip_loadbalancer --ip=10.0.1.1 --servers=10.0.0.1,10.0.0.2**

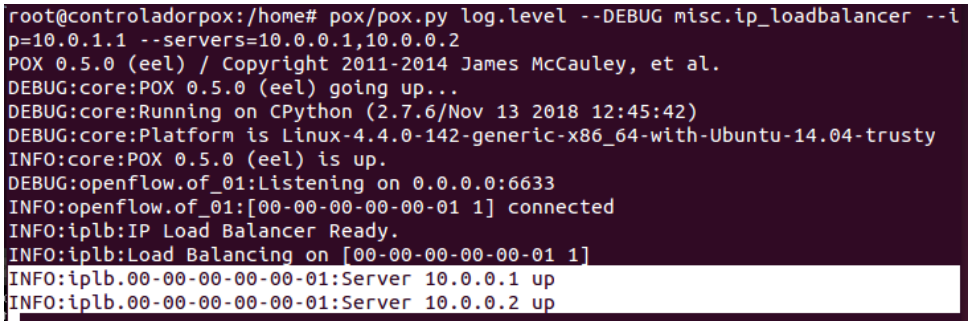

Fig. 18 **Inicio del controlador POX como un balanceador de cargas [Fuente propia]**

Abrir terminales para los hosts restantes con el comando: **xterm h3 h4 h5 h6**.

Con el comando **curl** se envía tráfico al controlador desde host # 3, tal cual como se tiene en la **Fig. 19**.

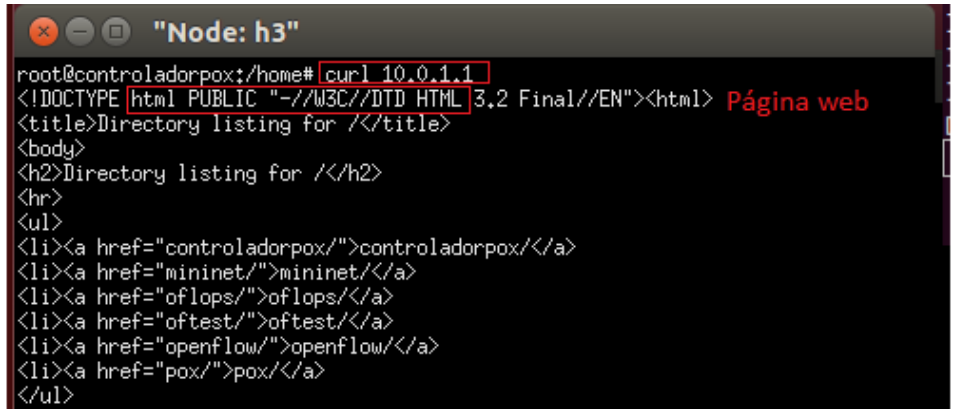

Fig. 19 **Envío de tráfico [Fuente propia]**

En la terminal donde se tiene iniciado el controlador se puede observar un mensaje que demuestra el redireccionamiento del tráfico al servidor *HTTP* 1 con dirección 10.0.0.1, como se tiene en la **Fig. 20**.

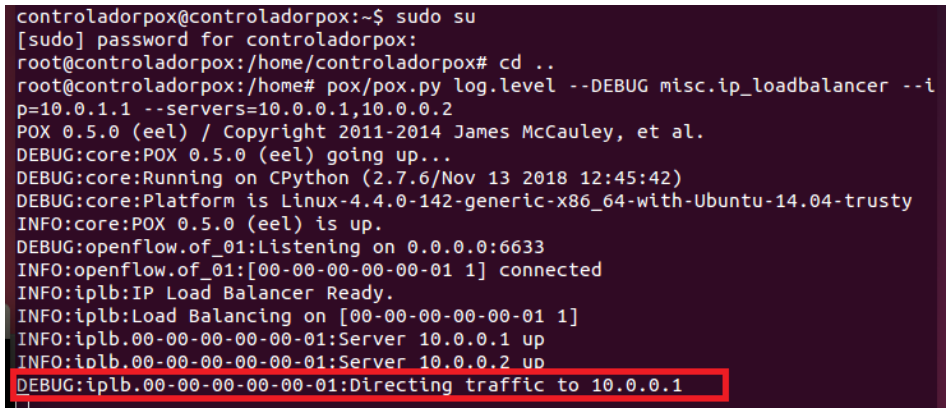

Fig. 20 **Redireccionamiento de tráfico [Fuente propia].**

En el terminal del host 1 se puede observar un mensaje de solicitud desde el host 3 con dirección 10.0.0.3 hacia el servidor *HTTP* 1, el cual está situado en el host 1. Esto lo podemos visualizar en la **Fig. 21**.

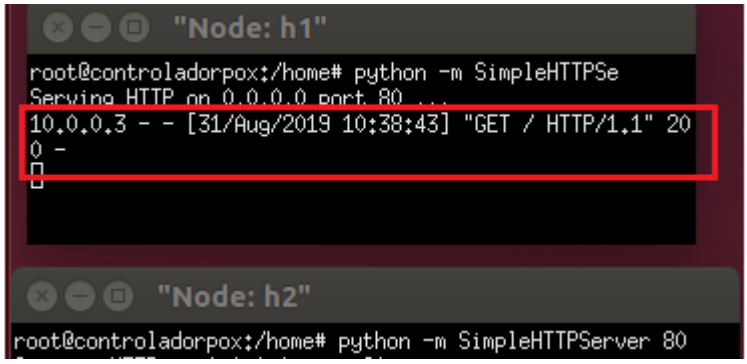

Fig. 21 **Mensaje de solicitud [Fuente propia]**

Todo este proceso se lo repite en los diferentes hosts.

Y, por último, salir de Mininet con el comando “**exit**”, como se muestra en la **Fig. 22**.

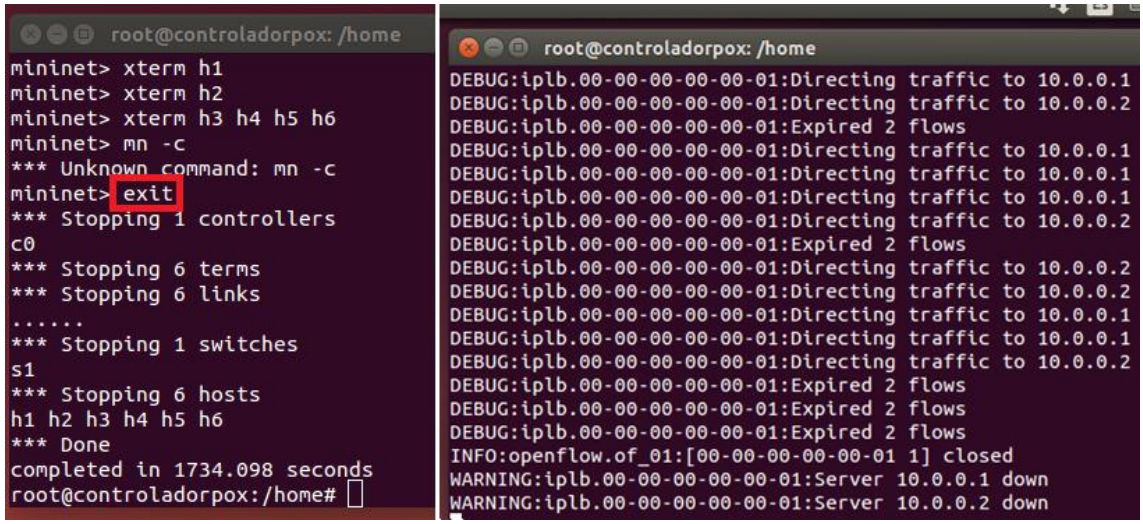

Fig. 22 **Salida de Mininet [Fuente propia]**

Anexo C

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 3

USO DEL CONTROLADOR SDN POX COMO UN HUB.

Objetivos:

- Utilizar el controlador *SDN POX* como un hub.
- Acceder a Internet utilizando el *switch SDN*.

Instrucciones:

- Instalar el sistema operativo raspbian en las tarjetas *SD*.
- Realizar las conexiones de los hosts a utilizarse.
- Configurar las interfaces de red que van a ser utilizadas.

Desarrollo:

Conectar la tarjeta *Zodiac FX* vía *USB* a la *Raspberry Pi* y actualizar los paquetes a su última versión, obtener el puerto en el cual ha sido conectada la tarjeta *Zodiac FX* e instalar el programa putty accediendo como superusuario: **sudo apt-get update**.

Una vez conectada la tarjeta *Zodiac FX* a la *Raspberry Pi* vía *USB* se debe ingresar en la terminal el comando **dmesg** este comando obtiene información de arranque del sistema, mensajes generados durante la depuración de aplicaciones, mensajes del núcleo (kernel: parte central de un sistema operativo), entre otros datos y los dispositivos conectados a la *Raspberry Pi.*

Al utilizar este comando se muestra la información del puerto en el cual ha sido conectada la tarjeta *Zodiac FX*, el nombre del dispositivo y su productor: “cdc_acm 1-1.3:1.0: **ttyACM0**: *USB* ACM device.”

Descripción del puerto en el cual se ha conectado la tarjeta *Zodiac FX*. En la **Fig. 1** se tiene la información del sistema:

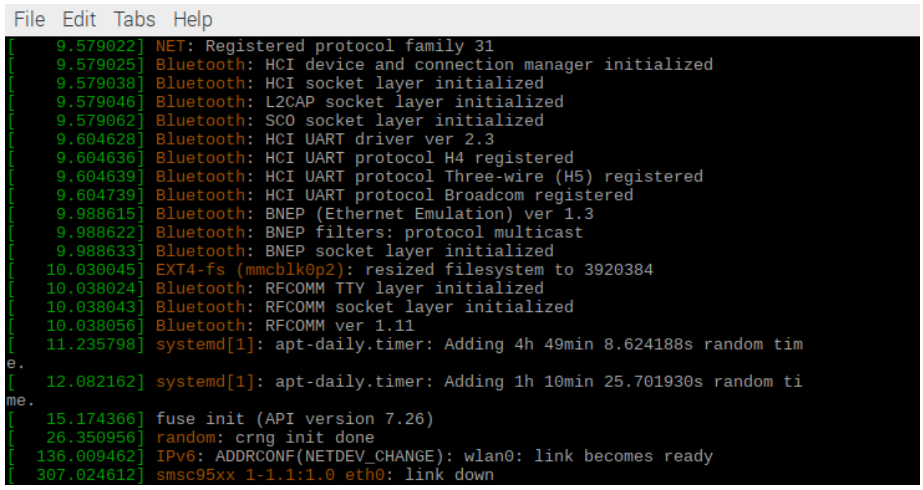

Fig. 1 **Información del sistema [Fuente propia]**

En la **Fig. 2** se observan los datos y la información de la tarjeta Zodiac FX ya conectada:

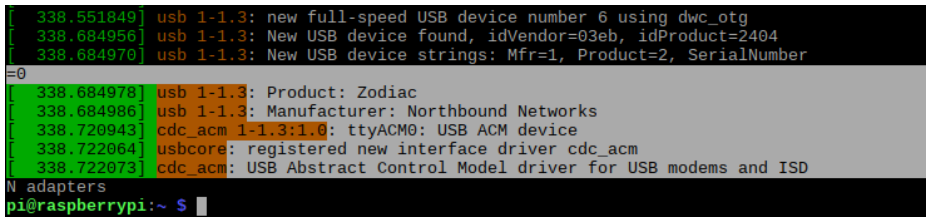

Fig. 2 **Datos de la tarjeta Zodiac FX conectada [Fuente propia]**

Putty es el programa que nos permite conectarnos a la *Zodiac FX* por medio de una terminal de manera serial, soporta *SSH* y muchos protocolos más. Cuando termine su instalación iniciaremos putty ya sea desde la terminal con el comando **putty** o buscándolo en las aplicaciones instaladas en la *Raspberry Pi*: **sudo apt-get install putty**.

En el programa putty se elige la opción **serial** y se debe especificar el puerto donde se encuentra conectada la tarjeta *Zodiac FX*. La informacion se obtuvo con el comando **dmesg** en pasos previos, pero, cabe recalcar que el puerto serial de la *Zodiac FX* siempre es **ttyACM0**. Esto se lo puede visualizar de una forma adecuada en la **Fig. 3**.

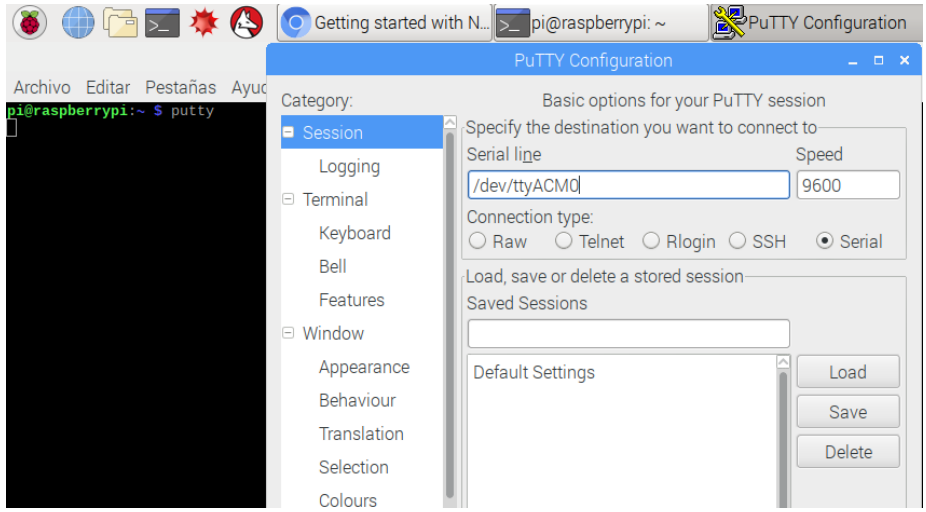

***Fig. 3* Acceso a la Zodiac FX [Fuente propia]**

Una vez se haya accedido a la terminal de la *Zodiac FX*, se debe de dar “Enter” para poder visualizar el logo del mini *switch*, como se demuestra en la **Fig. 4**.

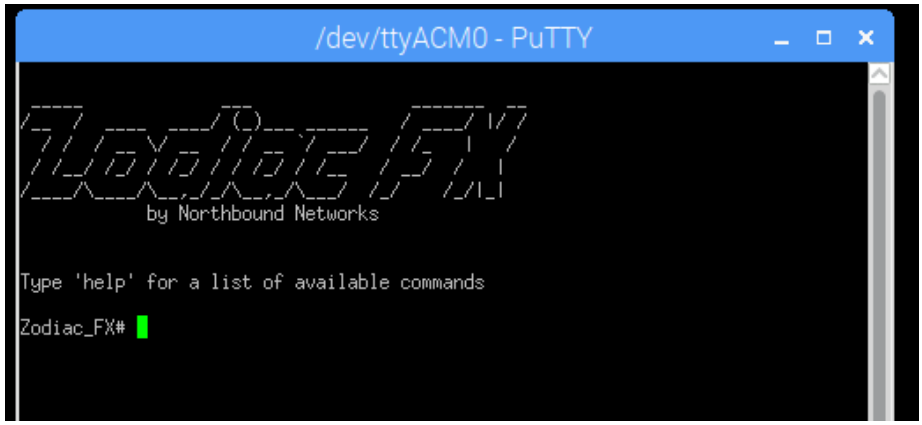

Fig. 4 **Terminal de Zodiac FX [Fuente propia]**

Se digita "**config**" para acceder al apartado de configuración. Se aconseja siempre, como primer paso, hacer un reseteo de fábrica a la configuración de la *Zodiac FX*, esto se lo realiza con el comando "factory reset" ara tener mas información sobre la configuración de la tarjeta *Zodiac FX*, se puede visualizar con el comando "**show config**", como se señala en la **Fig. 5**.

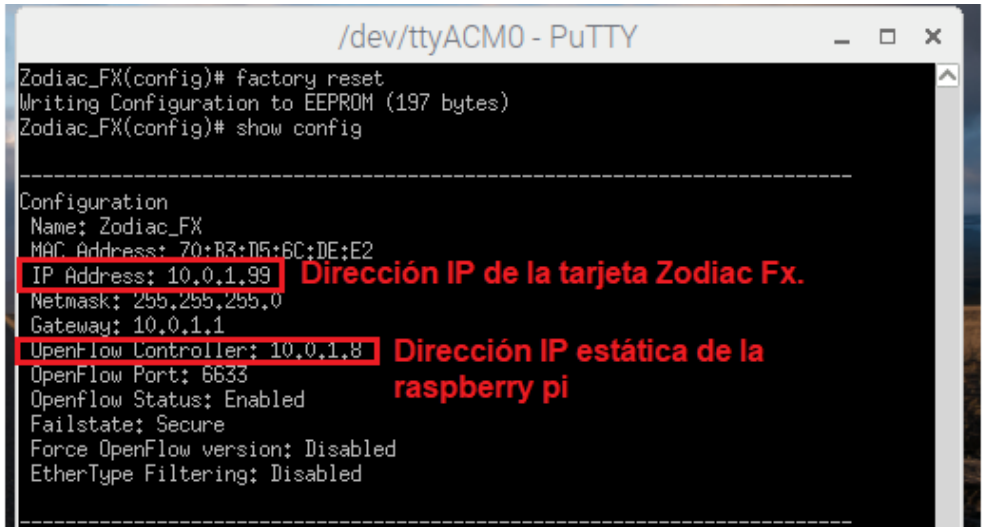

Fig. 5 **Restauración de la configuración [Fuente propia]**

Ingresar al modo de configuración de la tarjeta *Zodiac FX* utilizando el comando **config** y restaurar la configuración de fábrica con el comando **factory reset**.

Salir del modo de configuración con el comando **exit** y utilizar el comando **show ports** para observar la información de los puertos, se debe visualizar que el puerto 4 tipo *OpenFlow* de la tarjeta *Zodiac FX* se encuentra activo (UP) ya que está conectado a la *Raspberry Pi* por medio del cable de red y el resto de puertos se muestran en estado inactivo (DOWN) debido a que no se encuentran conectados dispositivos en esos puertos ethernet por el momento.

Se deben configurar las interfaces de red de la *Raspberry Pi* en la terminal accediendo como superusuario con el editor nano se accede al archivo /etc/network/interfaces: **sudo nano /etc/network/interfaces**.

Donde “eth0” se refiere a la red cableada y “wlan0” a la red inalámbrica de la *Raspberry Pi*. Se especifica que la red cableada debe ser configurada de manera estática con una dirección *IP* en el mismo rango que la dirección *IP* de la tarjeta *Zodiac FX* y la red inalámbrica configurada por medio del protocolo *DHCP* el cual asignará una dirección *IP* dinámica a la *Raspberry Pi*. En el campo wpa-ssid se debe especificar el nombre de la red *WiFi* a la que nos conectaremos y la contraseña en el campo y wpa-psk como se muestra en la **Fig. 6**.

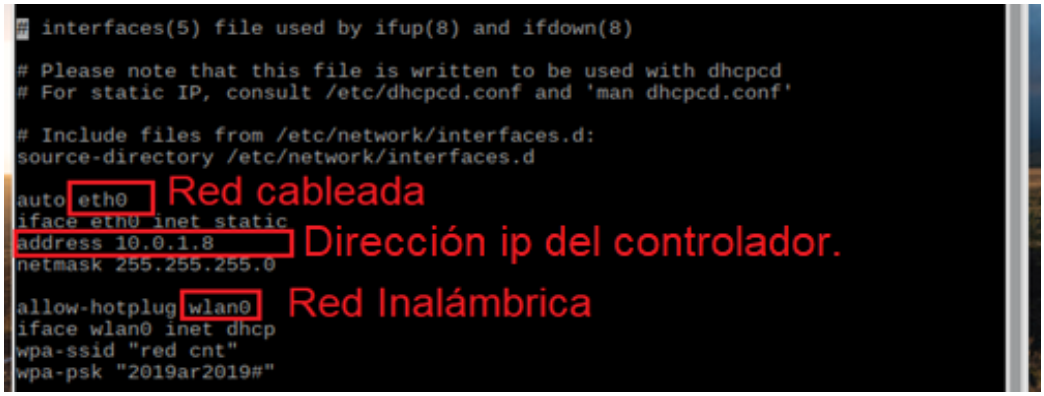

Fig. 6 **Configuración de las interfaces de red [Fuente propia]**

Guardar la configuración utilizando la combinación de teclas **Ctrl** + **o** y **Ctrl** + **x** para salir del editor nano. Reiniciar la *Raspberry Pi* y conectar el cable de red al puerto ethernet de la *Raspberry Pi* hacia el puerto # 4 de la tarjeta *Zodiac FX* y probar la conectividad haciendo ping a la *Zodiac FX.* Recordando que la dirección *IP* de fábrica de la *Zodiac FX* es 10.0.1.99, como se puede observar en la **Fig. 7**.

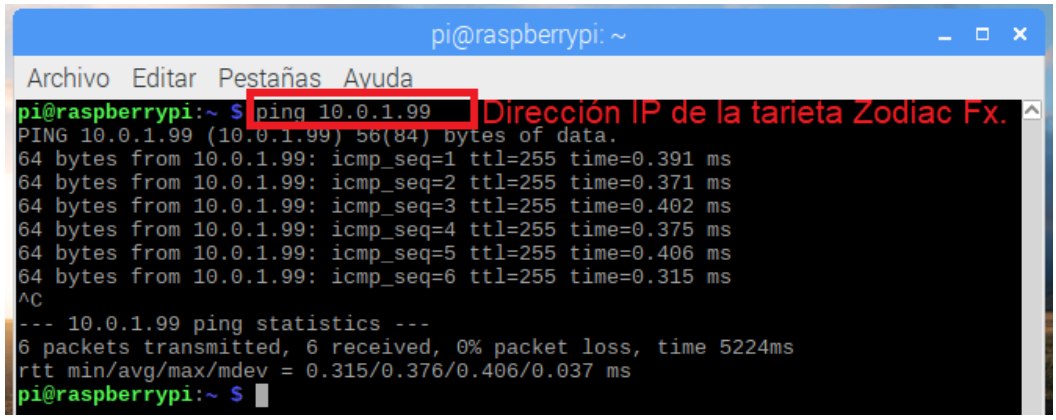

Fig. 7 **Prueba de conectividad [Fuente propia]**

Instalar GitHub con el comando: **sudo apt-get install git**.

Se deben clonar los archivos de *POX* con las últimas actualizaciones desde su repositorio. Clonar con el comando **git clone https://github.com/noxrepo/pox.git** y **cd pox** para ubicarse en la carpeta. Para iniciar el controlador *POX* se utiliza la estructura de *Python* y el componente **openflow.of_01** comunicándose por el puerto **6633**.

El componente **openflow.of_01** se comunica con los *switches OpenFlow* con versión 1.0 del protocolo. Cuando se cargan otros componentes que usan *OpenFlow*, este componente generalmente se inicia con valores predeterminados automáticamente. Sin embargo, es posible que se desee iniciarlo manualmente para cambiar sus opciones. También es posible iniciarlo manualmente para ejecutarlo varias veces (por ejemplo, para escuchar conexiones *OpenFlow* en múltiples puertos).

En la **Fig. 8** se tiene el inicio del controlador usando el comando: **sudo ./pox.py openflow.of_01 --address=10.0.1.8 –port=6633**.

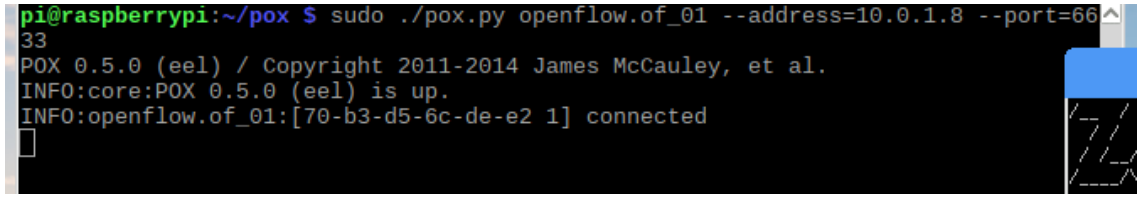

Fig. 8 **Inicio del controlador [Fuente propia]**

A continuación, se presenta una descripción de los comandos a utilizar como se muestra en la **TABLA I**.

TABLA I

Descripción de los comandos [Fuente propia].

Comando.	Opción por defecto.	Descripción.
--certificate=<X>	NA	Habilita el modo *SSL* y especifica un archivo certificado.
--ca-cert=<X>	NA	Habilita el modo *SSL* y especifica un certificado para validar los *switches*.
--private-key=<X>	NA	Habilita el modo *SSL* y especifica un archivo clave.
address=<X>	all addresses (Todas las direcciones)	Especifica las direcciones *IP* de las interfaces a trabajar.
--port=<X>	6633	Especifica el puerto *TCP* para escuchar conexiones.

Ingresar al apartado ***OpenFlow*** de la tarjeta *Zodiac FX* y mostrar el estado con **show status** siguiendo los pasos de la **Fig. 9**.

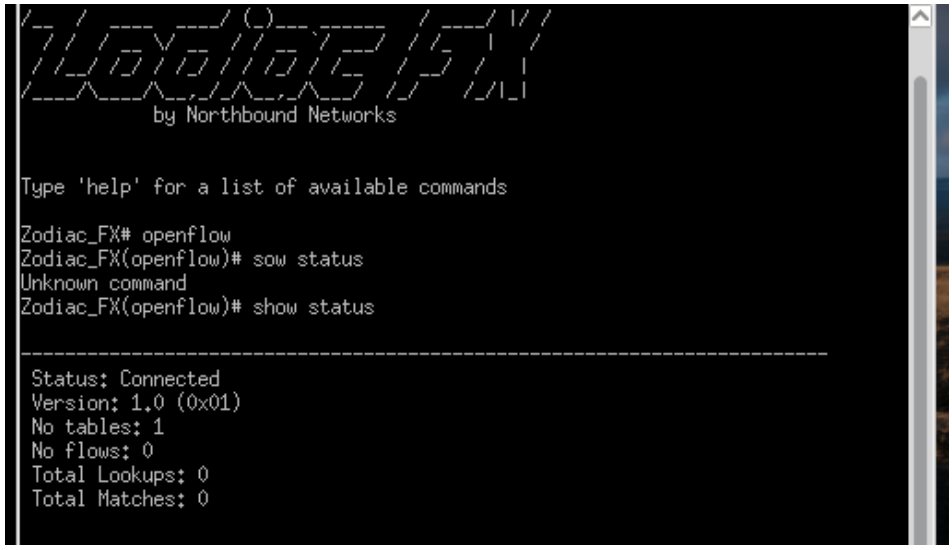

Fig. 9 **Apartado OpenFlow de Zodiac FX [Fuente propia]**

Utilizar **ctrl** + **c** para finalizar una ejecución.

Al finalizar la ejecución, se revisa nuevamente el estado *OpenFlow* de la *Zodiac FX* para poder asegurar que el estado de ésta se encuentre en "**Disconnected**", como se puede visualizar en la **Fig. 10**.

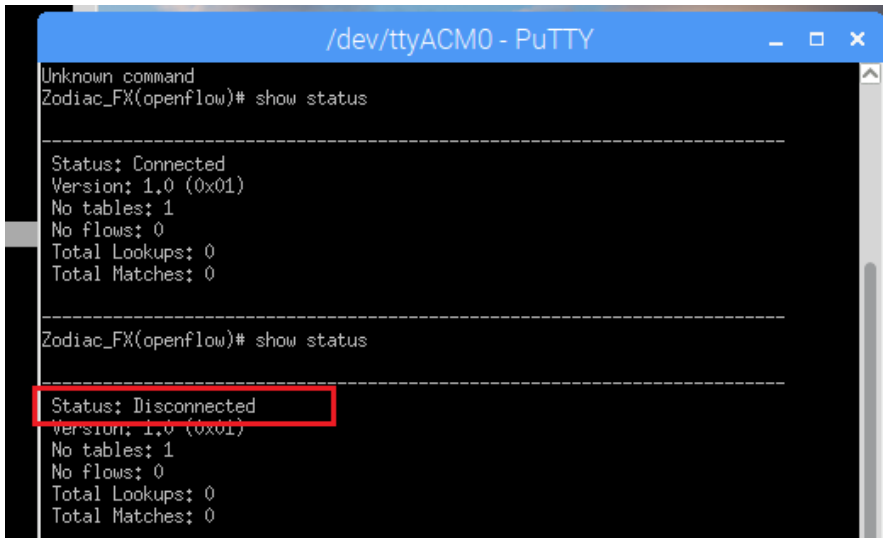

Fig. 10 **Estado desconectado [Fuente propia]**

En pasos anteriores se probó la conexión del controlador *POX* con la tarjeta *Zodiac FX*. A continuación, se iniciará el controlador como un hub (concentrador) que proporcionará conexión a internet. Usando el comando "**sudo ./pox.py --verbose openflow.of_01 --address=10.0.1.8 –port=6633 forwarding.hub**" se tiene el inicio del controlador *POX* como un hub, tal cual como se puede observar en la **Fig. 11**.

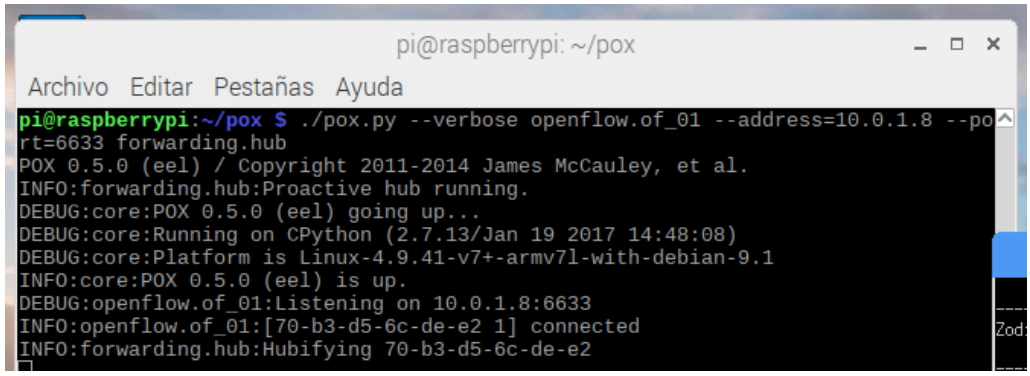

Fig. 11 **Inicio de POX como un hub [Fuente propia]**

Conectar el puerto 1 de la tarjeta *Zodiac FX* a la red de la universidad por medio de cable de red. Y en la terminal de la tarjeta *Zodiac FX* ingresar al modo "**OpenFlow**" y mostrar los flujos con "**show flows**". Se debe visualizar el flujo 1, el cual es el intercambio de paquetes entre la red de la Universidad y la tarjeta *Zodiac FX* por medio del controlador. Estos datos los podemos examinar en la **Fig. 12**.

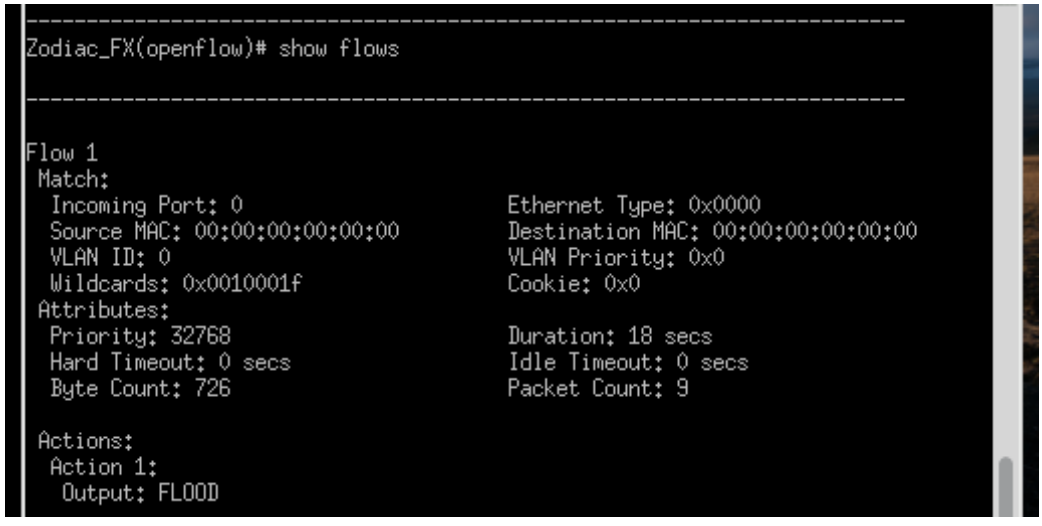

Fig. 12 **Flujo 1 [Fuente propia]**

Para ingresar a la interfaz gráfica de la tarjeta *Zodiac FX* accediendo a través de un navegador web ingresando su dirección *IP*, como se tiene en la **Fig. 13**.

Fig. 13 **Interfaz gráfica de la Zodiac FX [Fuente propia]**

Dentro de la interfaz gráfica, se pueden estudiar diferentes apartados, como, por ejemplo: el estado de los puertos, los flujos instalados en la tarjeta, la configuración, etc.

Para el armado de esta red se utilizarán dos *Raspberry Pi* que actuarán como unos hosts que accederán a internet por medio del controlador *POX* funcionando como un hub de red.

Previamente se debe grabar el sistema operativo en la tarjeta *SD* para ser instalado en las dos *Raspberry Pi* que actuaran como hosts.

Se conecta una *Raspberry Pi* en el puerto 3 y se puede visualizar en la **Fig. 14** el estado activo del puerto y el intercambio de paquetes desde el host hacia el controlador POX.

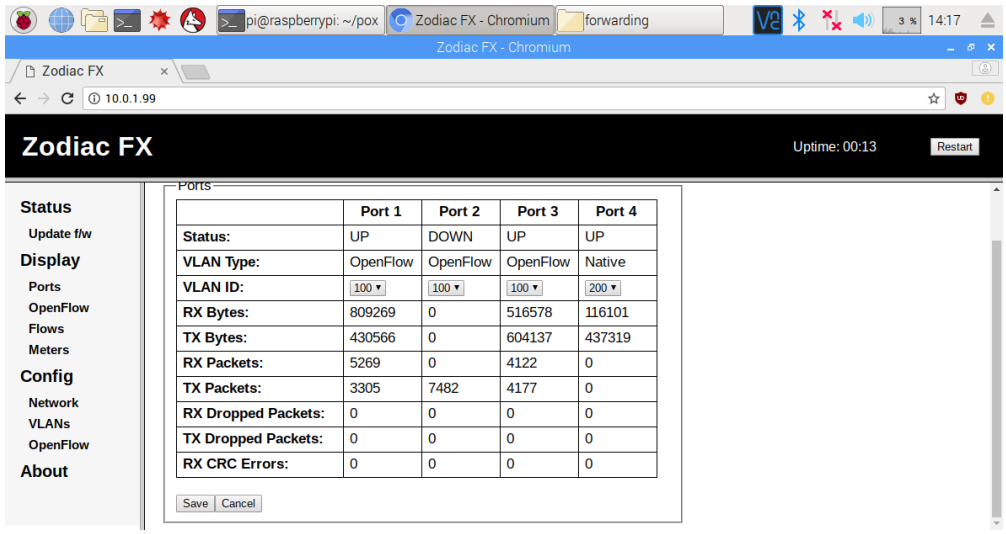

***Fig. 14* Estado de puertos de la tarjeta Zodiac FX. [Fuente propia]**

Luego se conecta una laptop en el puerto # 2 y se puede visualizar en la **Fig. 15** que le fue asignada una dirección *IP* dentro de la red. Para la práctica se debe utilizar una de las *Raspberry Pi* implementadas en el módulo.

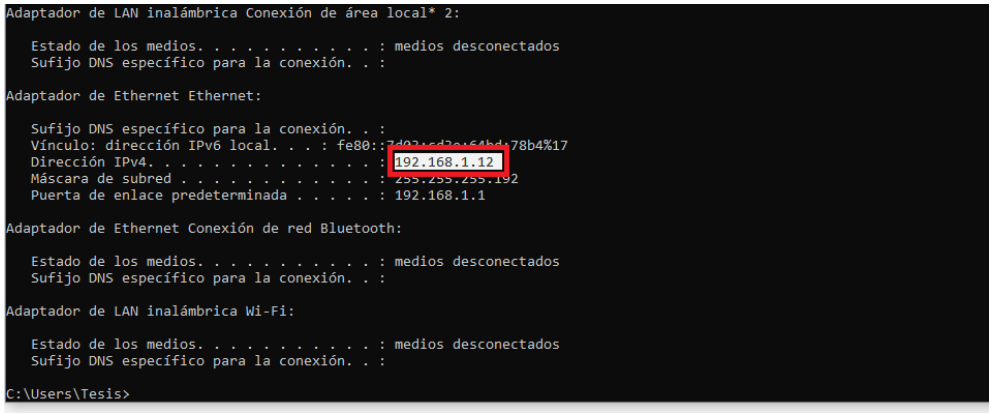

Fig. 15 **Asignación de dirección IP [Fuente propia]**

Una vez hecho esto, en la laptop, abrir un explorador y verificar si hay acceso a internet. En la **Fig. 16** se muestra que está conectada a la tarjeta *Zodiac FX* y en la **Fig. 17** se demuestra el acceso a internet mediante la conexión a la tarjeta *Zodiac FX*.

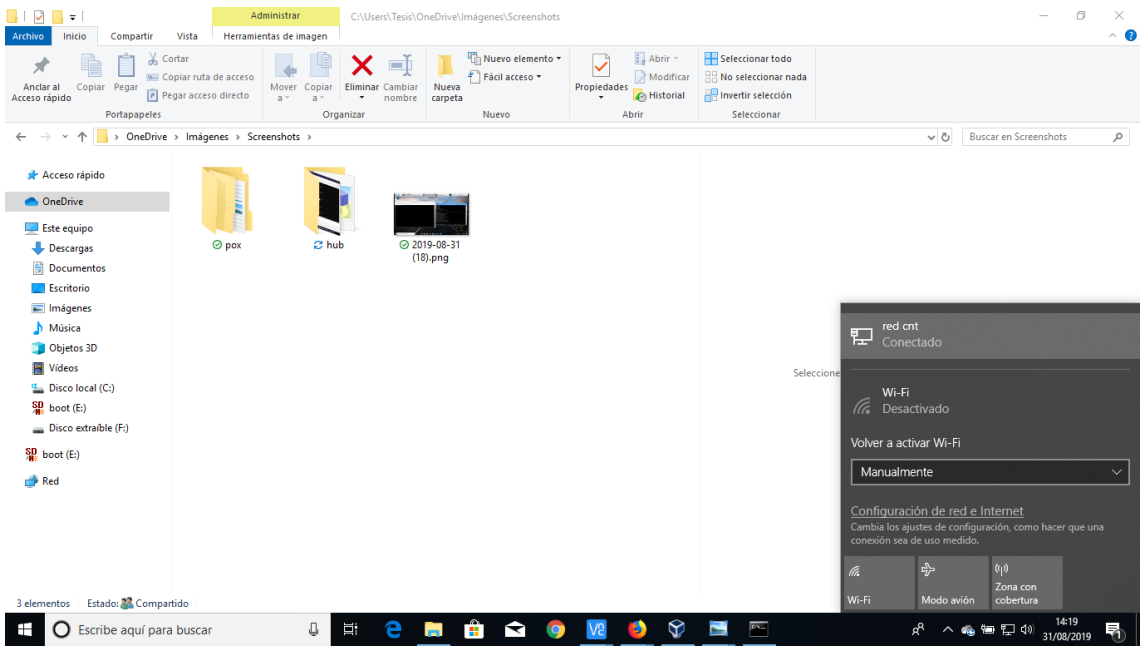

Fig. 16 **Laptop conectada a la Zodiac FX [Fuente propia]**

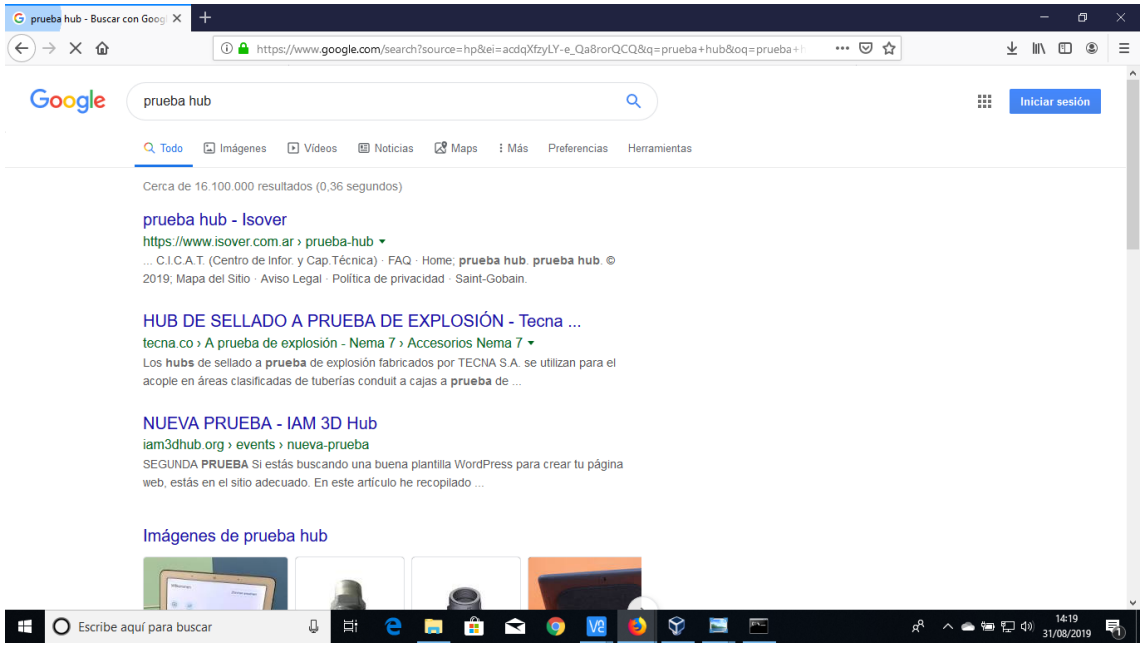

Fig. 17 **Acceso a internet [Fuente propia]**

Anexo D

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 4

RED DEFINIDA POR SOFTWARE CON EL CONTROLADOR FLOODLIGHT.

Objetivos:

- Analizar las topologías creadas en la interfaz web de *Floodlight*
- Crear y eliminar reglas en el controlador Floodlight.
- Observar los paquetes *OpenFlow* de la interfaz Loopback en el software Wireshark.

Instrucciones:

- Descargar el archivo *OVA* para exportarlo a la máquina virtual (en este caso Virtual Box), se la podrá descargar por medio del repositorio acordado.
- Instalar el software Wireshark.
- Descargar Python con todas sus librerías para su correcto funcionamiento.

Desarrollo:

El OVA que se encuentra en el repositorio ya tiene instalada por defecto el controlador *Floodlight*, también todos los programas necesarios para su debido funcionamiento, cuyas credenciales son como **usuario floodlight** y como **contraseña floodlight**. Ésta es la forma más idónea para proceder a realizar las pruebas venideras. Una vez descargado el archivo, habrá que impórtalo a la máquina virtual de esta forma:

Abrir la máquina virtual, dar clic en "**archivo**", luego seleccionar "**importar servicio virtualizado**", como se demuestra en la **Fig. 1**:

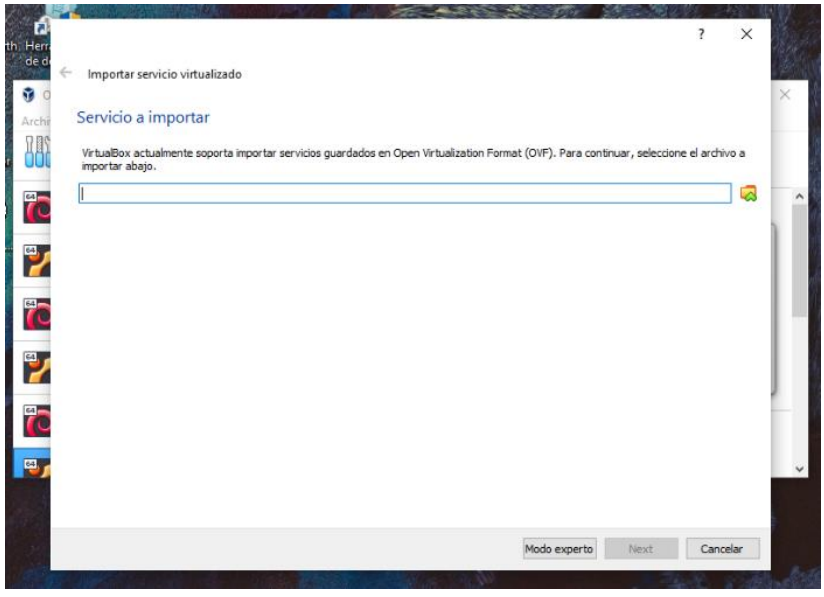

Fig. 1 **Importación de una máquina virtual [Fuente propia]**

Debe de aparecer una ventana para ingresar la dirección en donde se encuentra el archivo, se procede a dar clic en el ícono para buscar. Una vez hallado el archivo

seleccionar siguiente e importar. En la **Fig. 2** se muestra la ventana de importación de un servicio virtualizado:

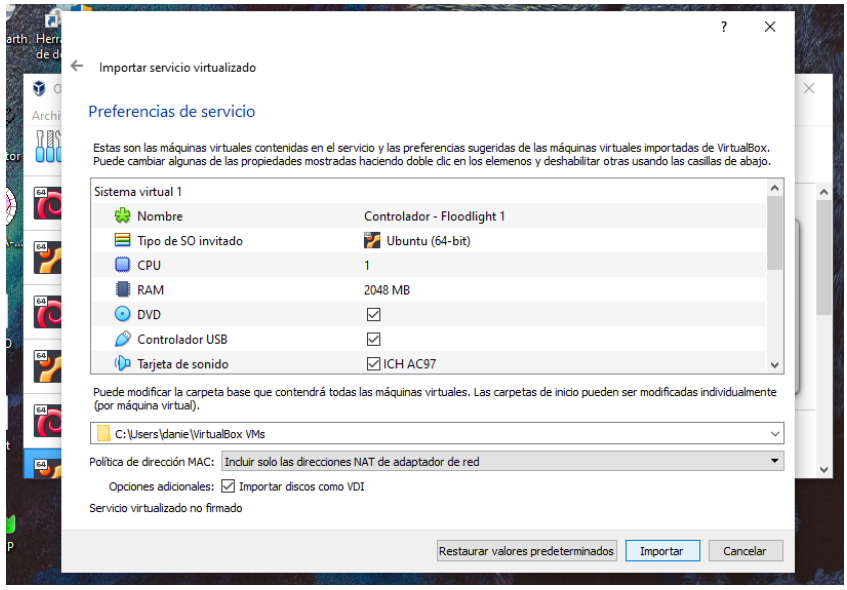

***Fig. 2* Ventana de preferencias de servicio [Fuente propia]**

Cuando finalice la importación del *OVA*, abrir la máquina virtual para iniciar la configuración de todo lo necesario. Cuando se encuentre corriendo la máquina, ejecutamos la terminal y nos situaremos en la ruta del controlador, como en la **Fig. 3**, con los siguientes códigos: **cd floodlight** y después **ant**.

```
floodlight@floodlight:~$ cd floodlight
floodlight@floodlight:~/floodlight$ ant
Buildfile: /home/floodlight/floodlight/build.xml

init:
    [mkdir] Created dir: /home/floodlight/floodlight/target/bin
    [mkdir] Created dir: /home/floodlight/floodlight/target/bin-test
    [mkdir] Created dir: /home/floodlight/floodlight/target/lib
    [mkdir] Created dir: /home/floodlight/floodlight/target/test

compile:
    [javac] Compiling 512 source files to /home/floodlight/floodlight/target/bin
    [javac] Note: Some input files use or override a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
     [copy] Copying 54 files to /home/floodlight/floodlight/target/bin

compile-test:
    [javac] Compiling 88 source files to /home/floodlight/floodlight/target/bin-
test
    [javac] Note: /home/floodlight/floodlight/src/test/java/net/floodlightcontro
ller/core/internal/OFSwitchHandshakeHandlerVer13Test.java uses unchecked or unsa
fe operations.
```

Fig. 3 **Ruta del controlador [Fuente propia]**

Ejecutar el controlador *Floodlight* por medio de Java con el siguiente comando: **java -jar target/floodlight.jar**. Esto se muestra en la **Fig. 4**.

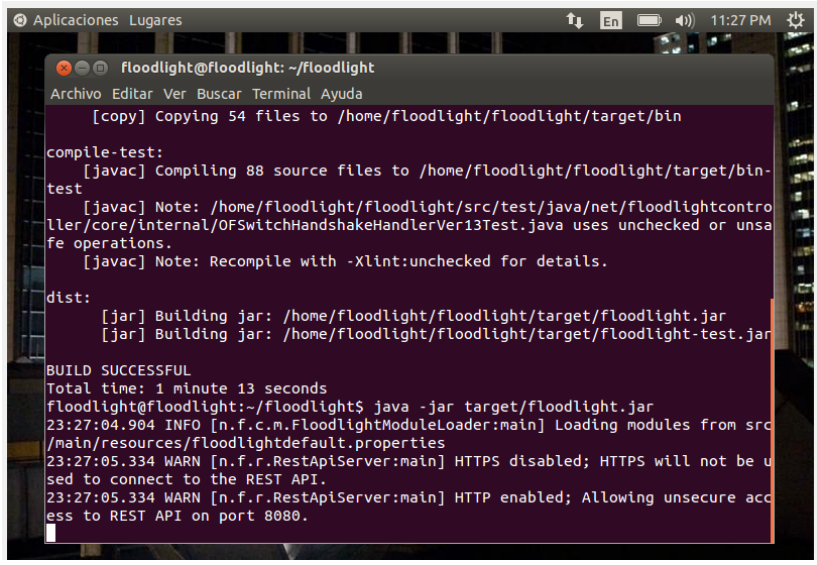

Fig. 4 **Ejecución del controlador [Fuente propia]**

Cuando el controlador ya se encuentre activo, abrir otra terminal de Ubuntu para consultar la dirección *IP* mediante el siguiente comando: **ifconfig**.

Una vez que se ha obtenido la dirección *IP* (de la eth0), se procede a realizar una conexión *SSH* a esa dirección *IP* mediante la siguiente línea de comando: **ssh -Y floodlight@"ip eth0"**. En la **Fig. 5** se observa la ejecución del comando dado anteriormente.

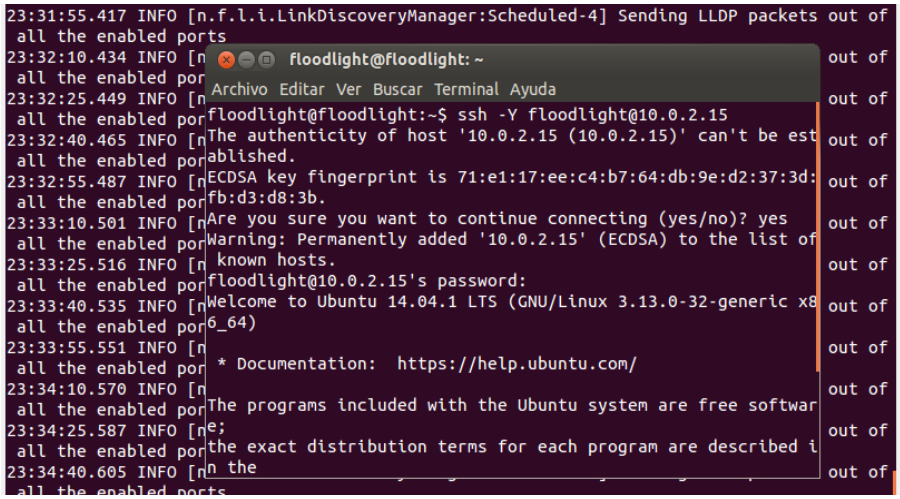

***Fig.* 5 Conexión SSH [Fuente propia]**

Como en esta *OVA* ya se preconfiguraron todos los programas que se pueden utilizar para la práctica, por ende, también viene instalado el software *Wireshark*, el cual es un analizador de protocolos utilizado para ver el tráfico que pasa a través de una red. En la **Fig. 6** se examina la ejecución del programa *Wireshark*.

Se debe iniciar con el siguiente comando: **sudo wireshark**.

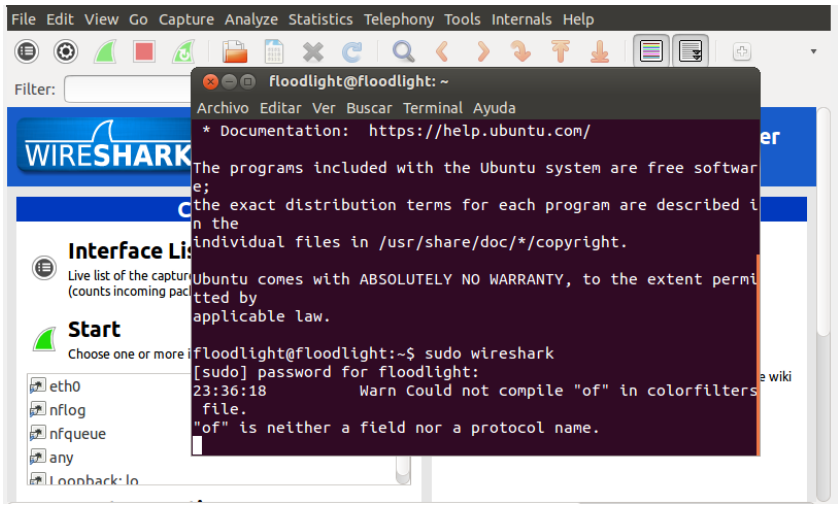

Fig. 6 **Inicialización de Wireshark [Fuente propia]**

En la venta de *Wireshark* se debe iniciar el escaneo en la interfaz loopback, para ello seleccionar loopback y dar clic en iniciar, como se muestra en la **Fig. 7**:

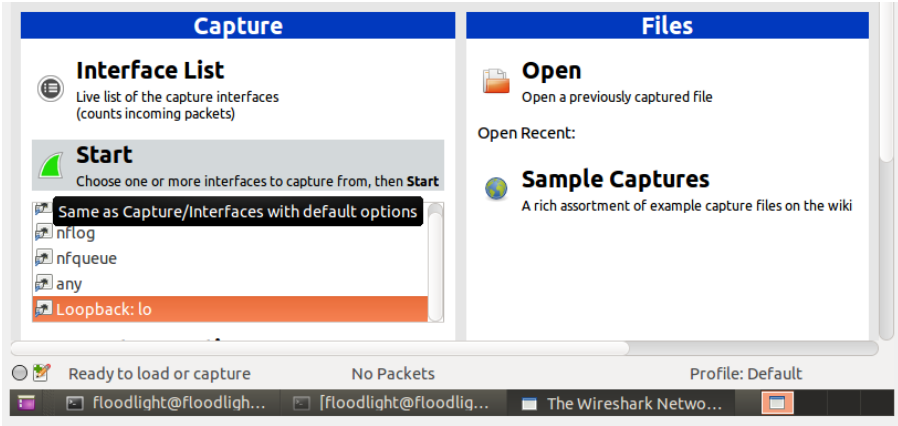

Fig. 7 **Escaneo de loopback [Fuente propia]**

En la **Fig. 8** se muestran todos los paquetes de tráfico, pero, como se muestran algunos protocolos que no son relevantes para esta práctica, se puede seleccionar un filtro para observar solamente los paquetes del protocolo *OpenFlow*.

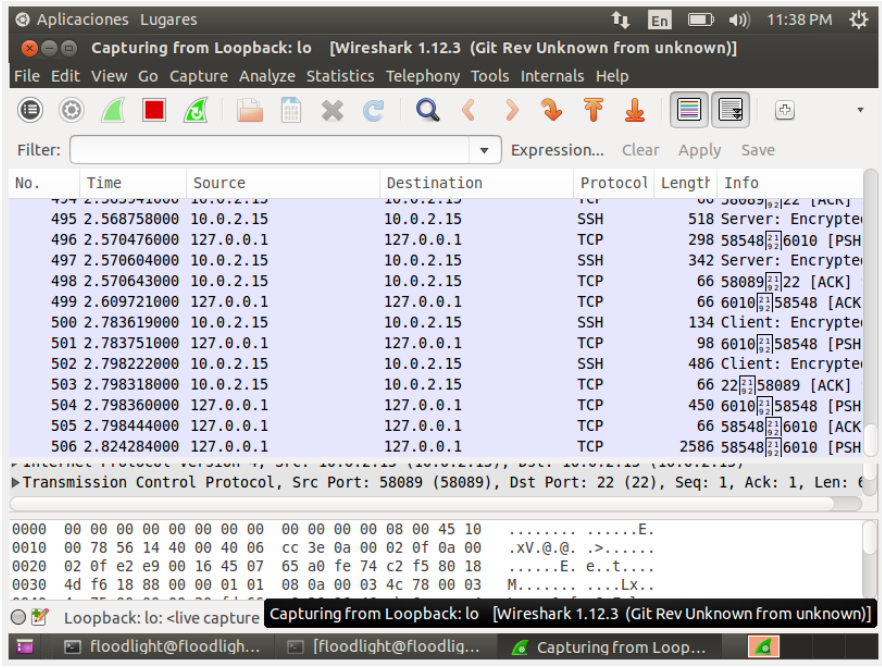

Fig. 8 **Paquetes de tráfico [Fuente propia]**

A continuación, iniciar *Mininet* con una topología por defecto, en este caso se utilizan las tres topologías que están por defecto en *Mininet*, las cuales con: single, lineal y de árbol. Primero realizar una prueba con la topología single con el siguiente comando: **sudo mn --topo single,3 --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13**.

Donde la *IP* 127.0.0.1 es la dirección *IP* de la loopback y OpenFlow13 es la versión de OpenFlow que se ejecutará.

Para realizar el envío de paquetes de un host a otro, la forma más fácil es hacer ping en cada uno de ellos, en la **Fig. 9** se muestra el ping entre h1 y h2, es decir, entre el host 1 y el host 2. Para ello utilizar el siguiente comando: **h1 ping h2**.

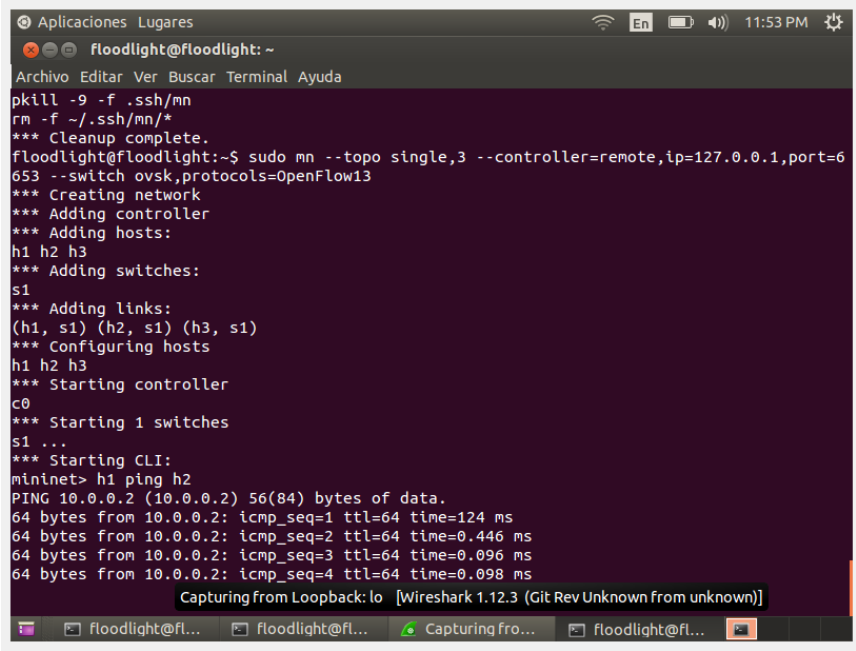

Fig. 9 **Ping entre h1 y h2 [Fuente propia].**

Observar los paquetes *OpenFlow* en el programa *Wireshark* que se ejecutó anteriormente. En la **Fig. 10** se obswervan todos los paquetes *OpenFlow* enviados debido al ping realizado anteriormente:

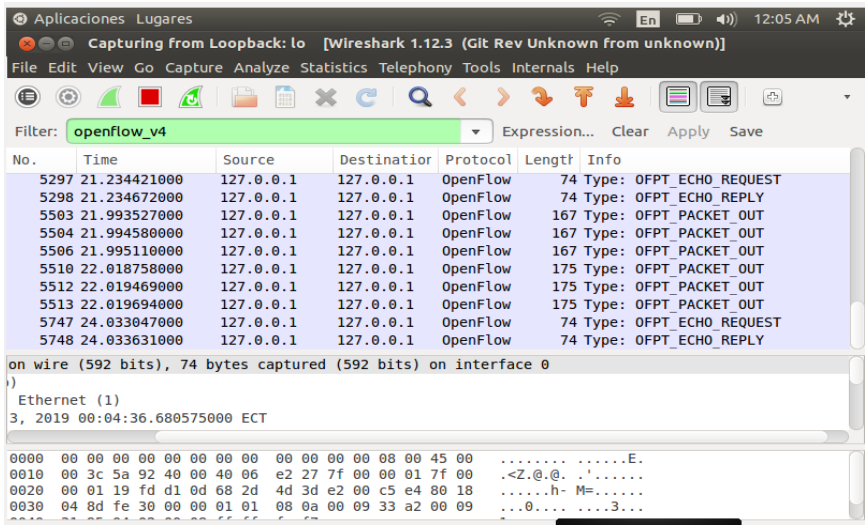

Fig. 10 **Paquetes OpenFlow [Fuente propia].**

Si se desea tener más información sobre algún paquete *OpenFlow*, dar doble clic para abrir cualquier paquete y se abrirá otra venta con dicha información.

Ahora lo siguiente es ingresar a la interfaz web del controlador, pero antes se debe instalar el paquete curl, el cual se usará más adelante. Para instalar la herramienta curl utilizar el siguiente comando: **sudo apt-get install curl**.

Una vez finalizada la instalación de curl, abrir el navegador para poder ingresar a la interfaz del controlador, se ingresa escribiendo la siguiente dirección: **http://localhost:8080/ui/index.html**. En la **Fig. 11** se evidencia la interfaz web del controlador *Floodlight*.

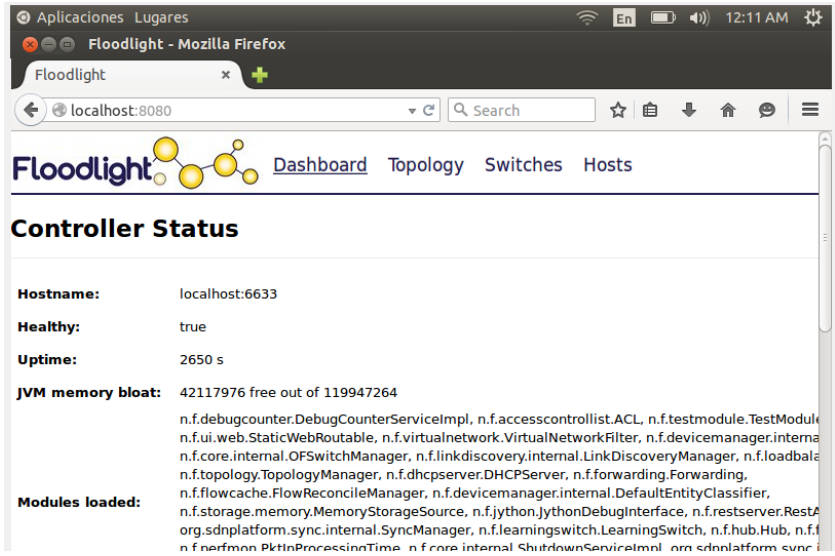

Fig. 11 **Interfaz web de Floodlight [Fuente propia]**

En este apartado se observan diferentes puntos, por ejemplo la topología formada anteriomente con *Mininet*, se puede ver la direccion *IP* y *MAC* de cada host creado con el comando ejecutado. Para ello dar clic en Topology como se muestra en la **Fig. 12**:

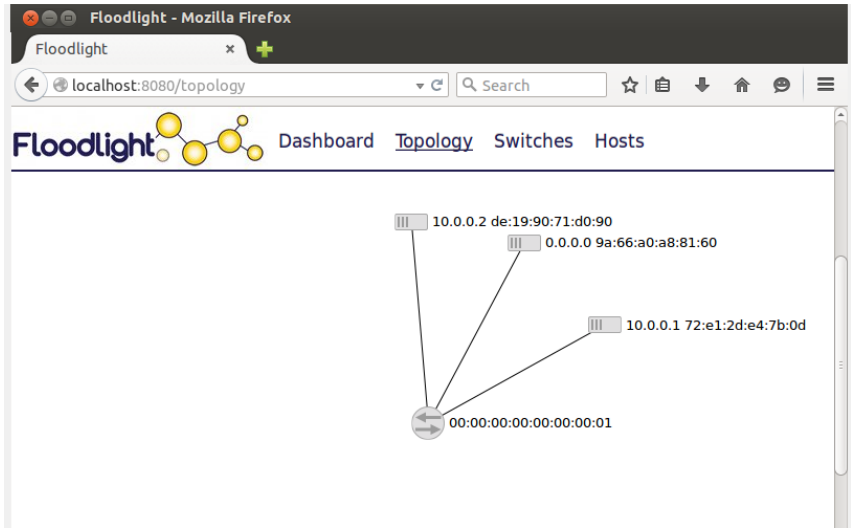

Fig. 12 Topología formada con Mininet [Fuente propia]

Ahora realizar una prueba con la topología linear de Mininet, la cual se crea ingresando el siguiente comando en una terminal tal cual se encuentra en la **Fig. 13**: **sudo mn --topo linear,4 --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13**.

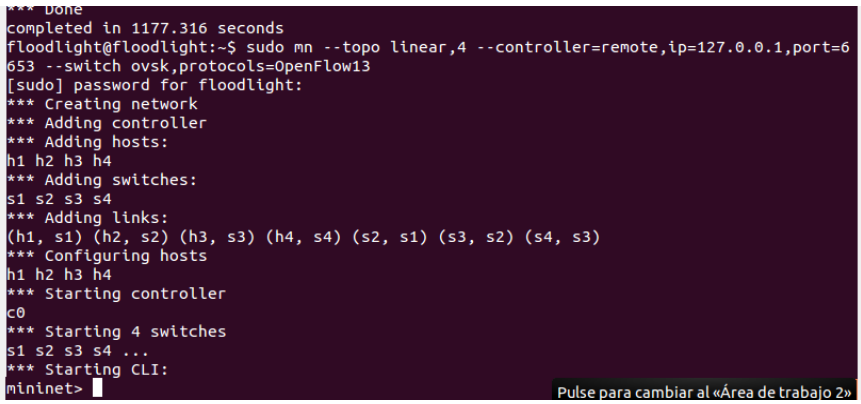

Fig. 13 **Topología Linear [Fuente propia]**

Si se desea crear una topología linear de 3, solamente se debe cambiar **"topo linear,4"** por **"topo linear,3"**. En este caso se creó una topología linear con 4 hosts y 4 *switches*. Recargar la interfaz web de Mininet y observar la información de cada host y *switch* en la interfaz gráfica como se visualiza en la **Fig. 14** y **Fig.15**:

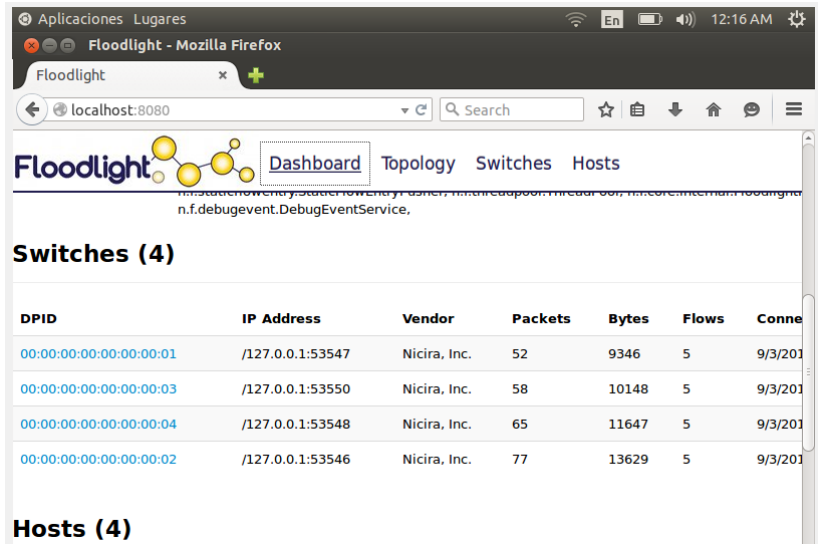

Fig. 14 **Switches virtualizados [Fuente propia]**

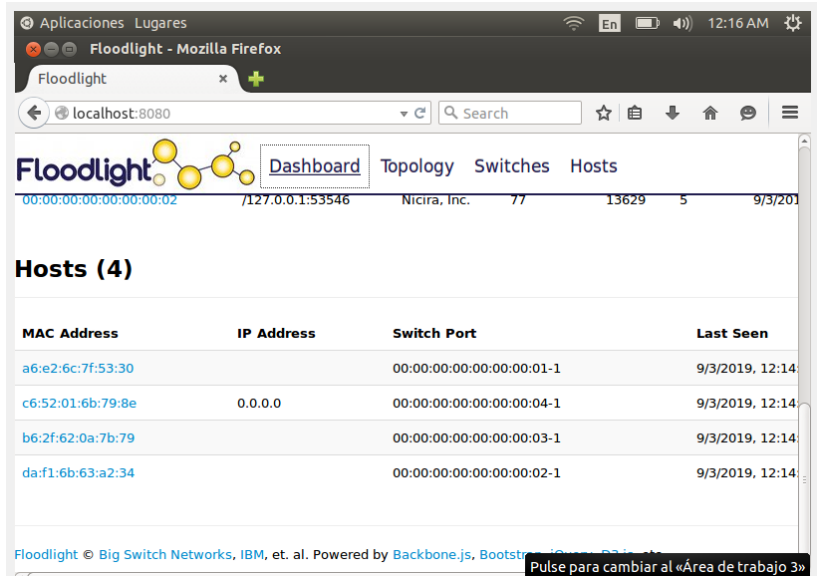

Fig. 15 **Hosts virtualizados [Fuente propia]**

Si se desea verificar la topología creada, ingresar nuevamente al apartado de topología de la interfaz web. La topología se la visualiza en la **Fig. 16**:

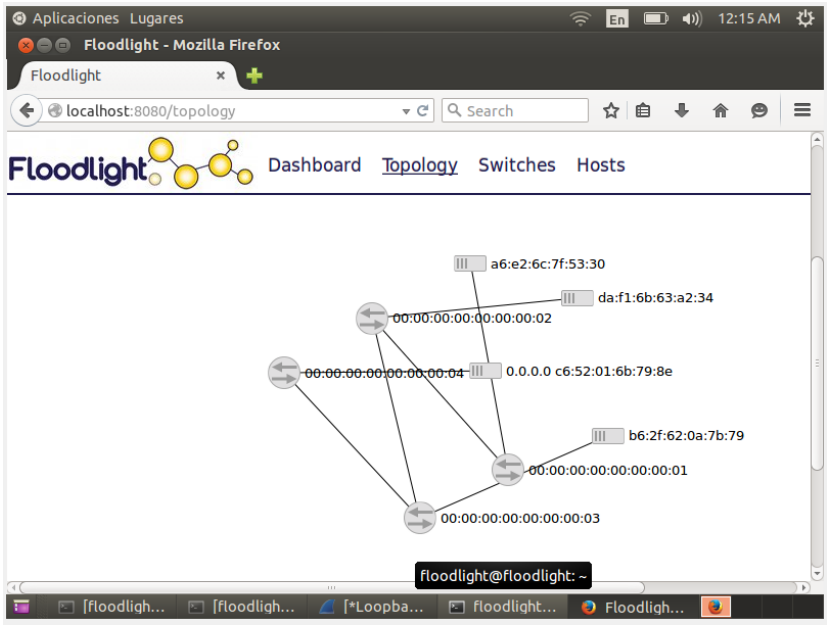

Fig. 16 **Topología linear [Fuente propia]**

Para capturar los paquetes de la topología linear, hacer un ping total para agilizar el proceso de hacer ping a cada host, como se observa en la **Fig. 17**. Para realizar el ping a todos escribir en la terminal, donde se corrió la topología linear de Mininet, lo siguiente: **pingall**.

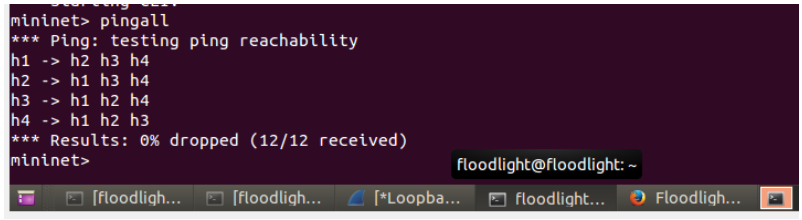

Fig. 17 **Ping a todos los hosts [Fuente propia]**

Ingresar de nuevo a Wireshark para observar los paquetes *OpenFlow* enviados y recibidos, si es necesario ejecutar de nuevo desde una terminal diferente. Una vez que

se haya hecho ping, actualizar la interfaz web para comprobar que los paquetes hayan cambiado.

Ya comprobado todo esto, ingresar a la terminal donde se ejecutó la topología linear de mininet para detener la actuación con **ctrl** + **z** y digitamos **exit**. Ahora realizar una prueba con la última topología por defecto de mininet, la cual es la topología árbol, en la **Fig. 18** se muestra la creación de dicha topología, escribir el siguiente comando en una terminal:

sudo mn --topo tree --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13.

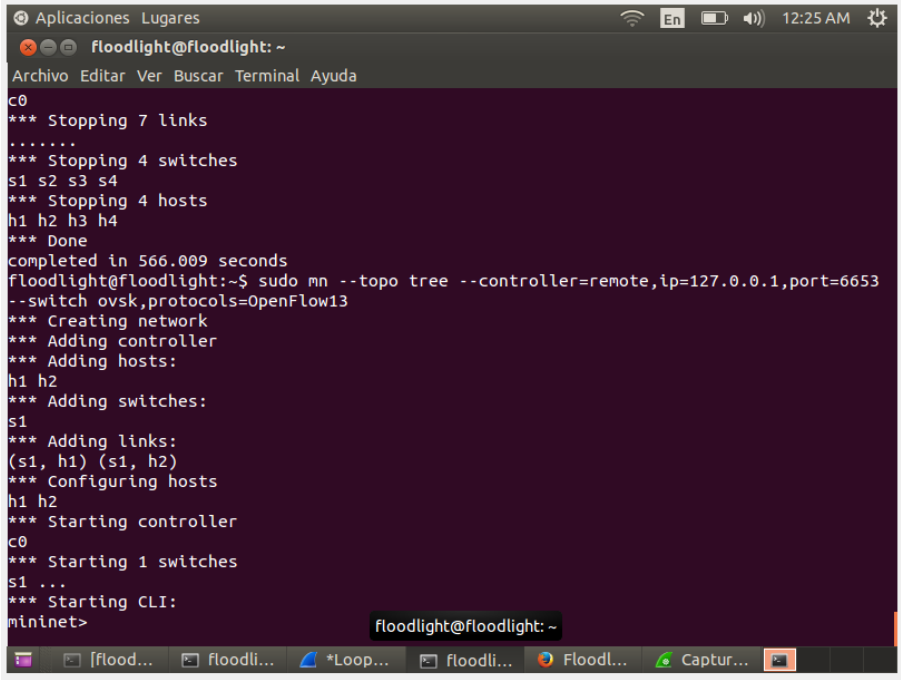

Fig. 18 **Ejecución de topología árbol [Fuente propia]**

Si se actualiza la interfaz web y se ingresa a la pestaña de topología, se observa la topología creada mediante el comando anterior. De igual forma que las anteriores, hacer ping entre los hosts para que se envíen paquetes. Utilizar lo siguiente: **h1 ping h2**.

Abrir Wireshark para observar los paquetes *OpenFlow*, como se muestra en la **Fig.19**.

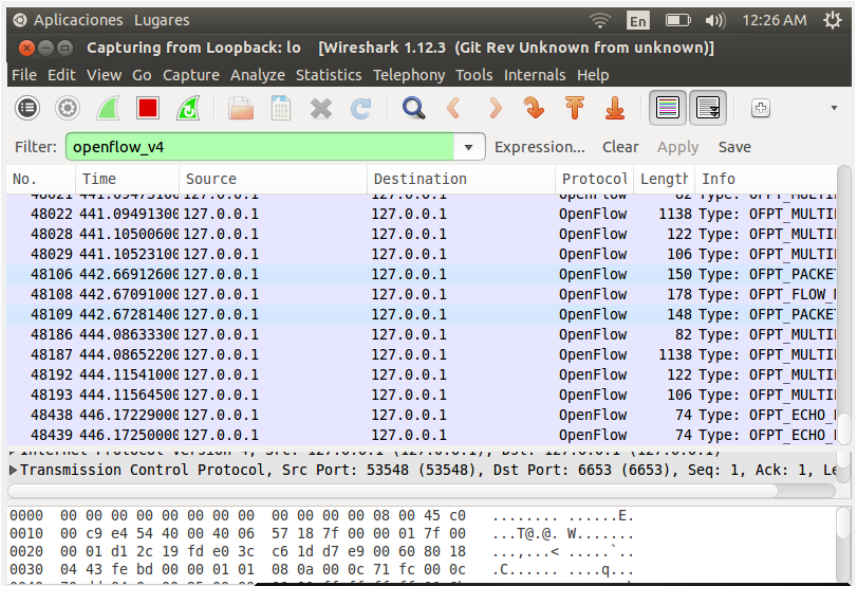

Fig. 19 **Paquetes OpenFlow de la topología árbol [Fuente propia]**

Actualizar la interfaz web y observar que los paquetes en el apartado del *switch* han cambiado, eso quiere decir que los paquetes pasaron por ese *switch*.

Detener la ejecución de Mininet para probar la topología single pero ahora con más hosts. Recordar que para la creación de la topología single el comando era: **sudo mn -**

-topo single,3 --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13.

Para hacerlo con 6 hosts en lugar de "**topo single,3**" escribir "**topo single,6**". El comando a ingresar para la topología single con 6 hosts sería: **sudo mn --topo single,6 --controller=remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13**.

Abrir y actualizar la interfaz gráfica para poder observar la topología elaborada mediante el comando ingresado anteriormente. En la **Fig. 20** se muestra la topología elaborada con el comando dado anteriormente.

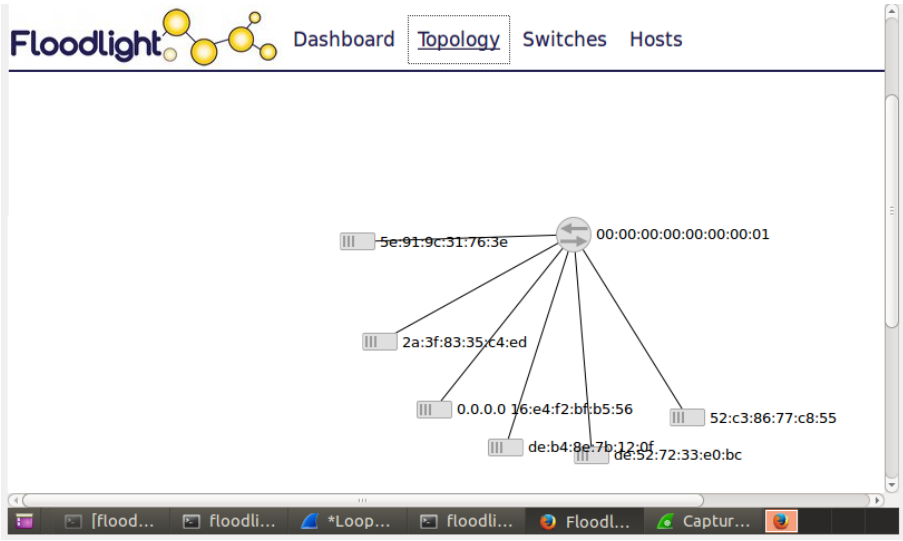

Fig. 20 Topología single con 6 hosts [Fuente propia].

Donde la dirección 127.0.0.1 es la dirección *IP* de la loopback y "OpenFlow13" es la versión de *OpenFlow* que se ejecutará. Lo siguiente es hacer ping a cada uno de los hosts, en este caso se utiliza **pingall** como se muestra en la **Fig. 21**:

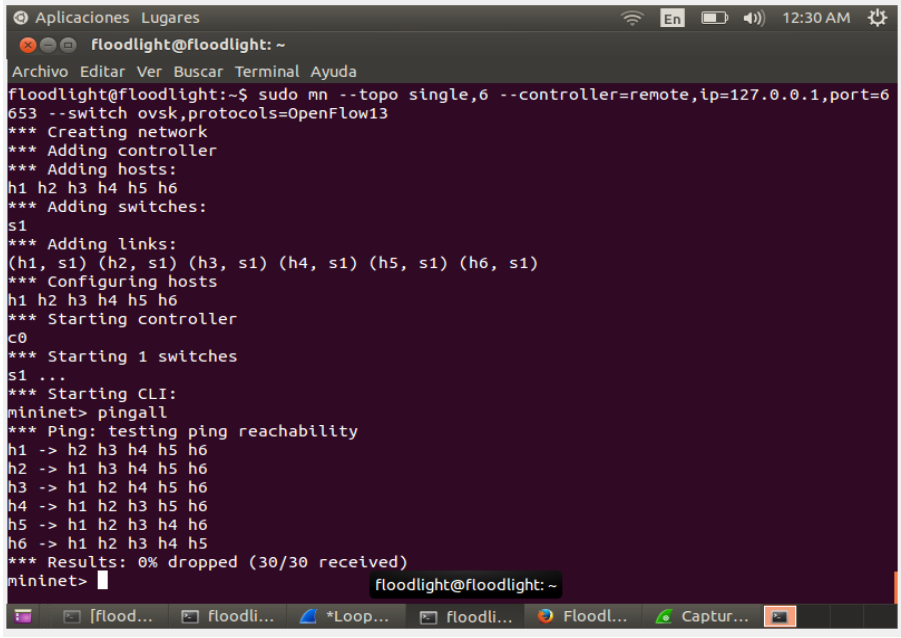

Fig. 21 **Ping entre los 6 hosts [Fuente propia]**

En Wireshark observar que ya hay paquetes *OpenFlow* enviados y recibidos.

Abrir nuevamente la interfaz web para asegurar que los paquetes hayan pasado por el *switch*. También se pueden observar los hosts creado junto con toda la información referente a ellos.

Ahora mediante curl, pasar comandos por web, es decir, añadir ciertas reglas para que, por ejemplo, se impida hacer ping de un host a otro, como se muestra en la **Fig. 22**. Para añadir la regla de negación de comunicación entre el host 1 y el host 2: **curl -X POST -d '{"src-ip":"10.0.0.1/32","dst-ip":"10.0.0.2/32","action":"deny"}' http://localhost:8080/wm/acl/rules/json**.

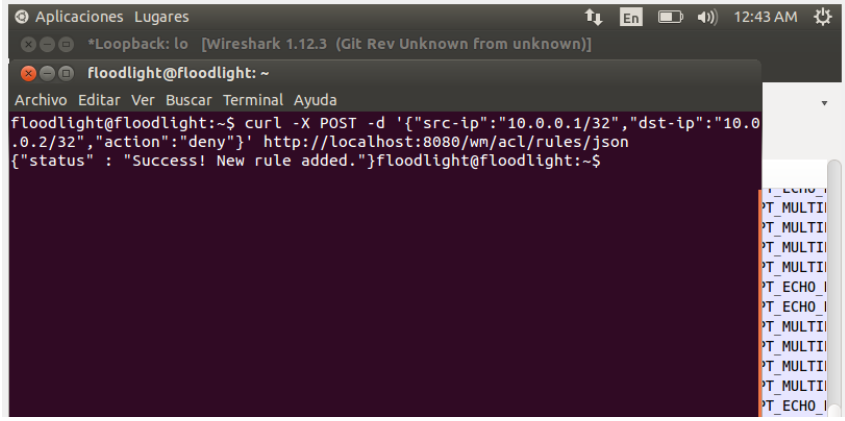

Fig. 22 **Denegación de ping [Fuente propia]**

Donde **10.0.0.1/32** es la dirección *IP* del host1 y **10.0.0.2/32** es la dirección *IP* del host 2. Para ver las reglas existentes utilizar el siguiente comando: **curl http://localhost:8080/wm/acl/rules/json | python -mjson.tool**. En la **Fig. 23** se muestra más detalladamente las reglas existentes.

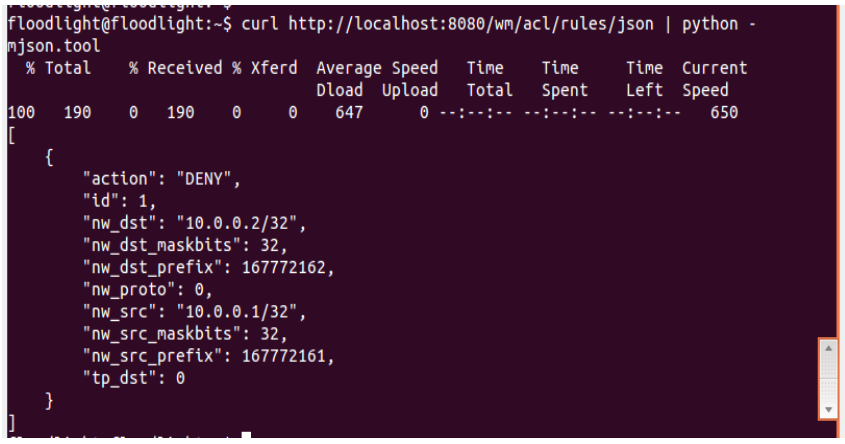

Fig. 23 **Reglas existentes del controlador [Fuente propia].**

Ahora comprobar que la regla funcione, en una terminal haremos ping entre h1 y h2, como se muestra en la **Fig. 24**:

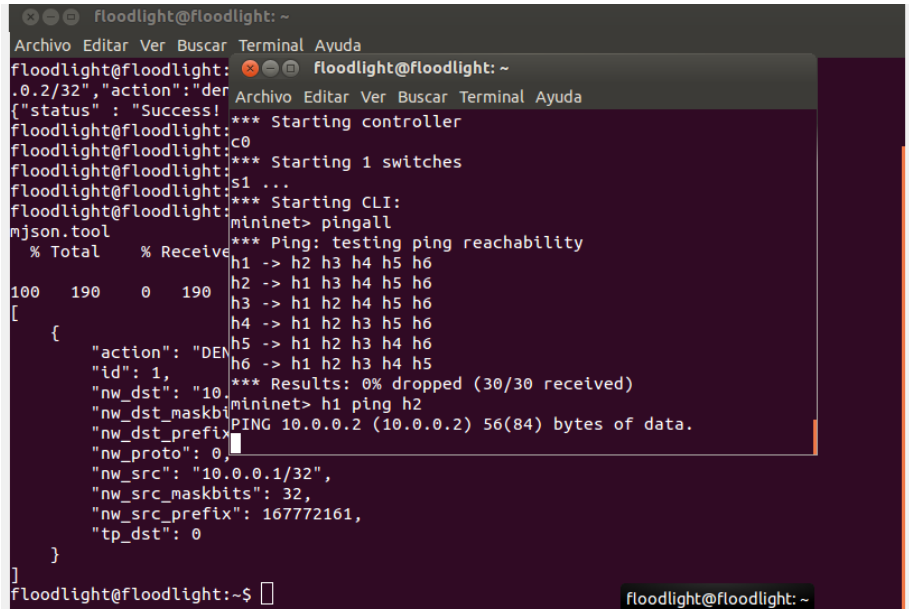

Fig. 24 **Ping denegado entre h1 y h2 [Fuente propia].**

Añadir nueva regla para denegar el ping entre h1 y h3 con el siguiente comando: **curl -X POST -d '{"src-ip":"10.0.0.1/32","dst-ip":"10.0.0.3/32","action":"deny"}' http://localhost:8080/wm/acl/rules/json**.

Donde **10.0.0.1/32** es la dirección *IP* del host 1 y **10.0.0.3/32** es la dirección *IP* del host 3. Después de esto, para verificar si la nueva regla se añadió correctamente, escribir nuevamente el comando que se muestra en la **Fig. 23** para observar las reglas creadas.

Como se observa en la **Fig. 25**, ahora hay dos reglas creadas en el controlador:

- Una para denegar la comunicación entre h1 y h2.

- La otra para denegar la comunicación entre h1 y h3.

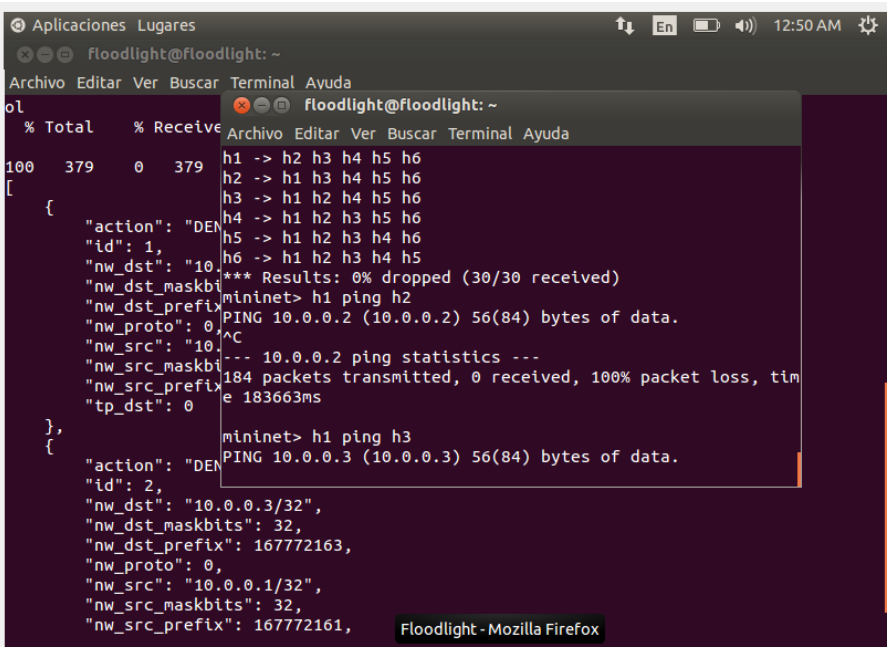

Fig. 25 **Ping entre h1 y h3 denegado [Fuente propia].**

Anteriormente se aprendió cómo añadir una regla al controlador, ahora procede a eliminar las reglas. Se puede eliminar una regla en específico o también eliminar todas las reglas existentes. Para eliminar una regla en específico utilizar el siguiente comando: **curl -X DELETE -d '{"ruleid":"1"}' http://localhost:8080/wm/acl/rules/json**.

Comprobar si la regla con id 1 se eliminó. En la **Fig. 26** se observa la eliminación de una sola regla, específicamente, la regla con *ID* 1:

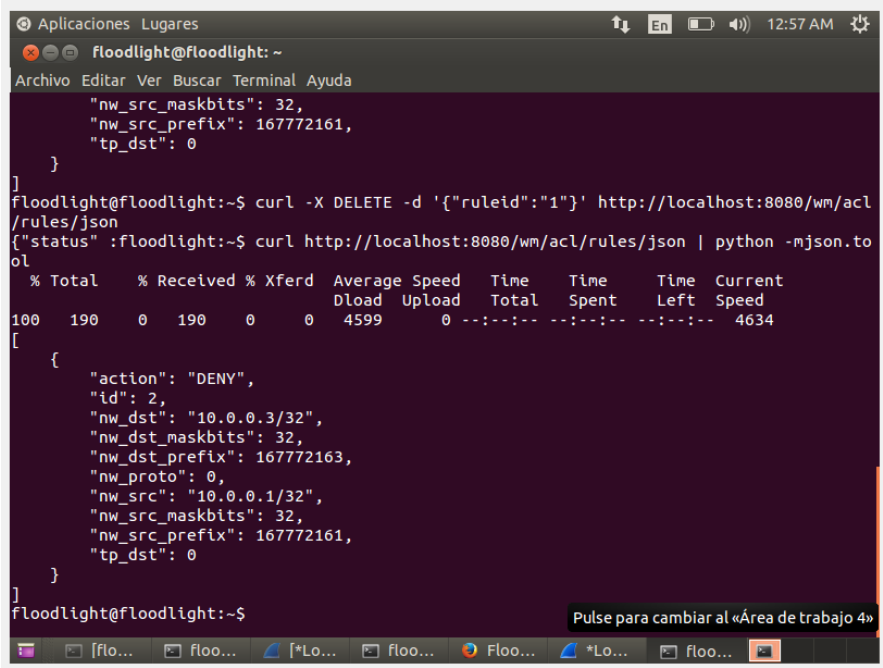

Fig. 26 **Regla específica eliminada [Fuente propia]**

Ya que la regla que impedía la comunicación entre el host 1 y el host 2 fue eliminada, hacer ping entre h1 y h2 para comprobar la comunicación:

Para eliminar todas las reglas creadas, lo primero sería añadir otra regla para eliminar ambas a la vez, luego comprobar que existan dos reglas creadas, como se tiene en la **Fig. 27**:

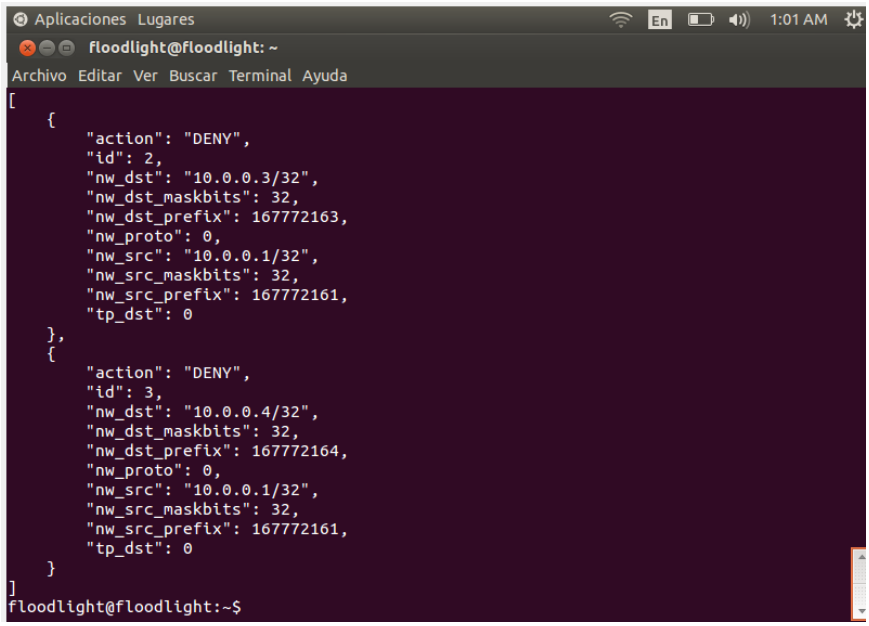

Fig. 27 **Nueva regla añadida [Fuente propia].**

Para eliminar todas las reglas creadas usaríamos el siguiente comando: **curl http://localhost:8080/wm/acl/clear/json**. En la **Fig. 28** se muestra la eliminación y la verificación de las reglas existentes:

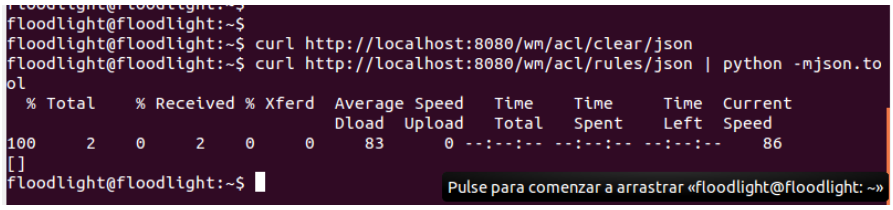

Fig. 28 **Eliminación y verificación de reglas [Fuente propia]**

Como se puede comprobar en la figura, primero se utilizó el comando para eliminar las reglas y después se utilizó el comando para verificar las reglas creadas, por ende, no se encontró ninguna regla creada.

Anexo E

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 5

RED DEFINIDA POR SOFTWARE UTLIZANDO HP VAN SDN CONTROLLER.

Objetivos:

- Configurar una red definida por software con el controlador SDN.
- Utilizar diferentes topologías utilizando los hosts con el controlador SDN.
- Crear y configurar las tablas de flujos.

Instrucciones:

- Instalar el sistema operativo Ubuntu 12.04.5 compatible con el controlador HP VAN SDN.
- Instalar el programa WinSCP.
- Descargar los archivos: del repositorio.
- Instalar el controlador SDN HP VAN SDN.

Desarrollo:

Iniciar la virtualización de la imagen ISO de Ubuntu 12.04 descargada del repositorio, esta versión del sistema operativo es compatible con el controlador, se indica esto debido a que en la versión 16.04 de Ubuntu se crean conflictos con el usuario sdnadmin motivo por el cual se utiliza esta versión de la distribución.

Reiniciar la máquina virtual y abrir la **terminal** de Ubuntu. Abrir la terminal de Ubuntu y obtener la dirección *IP* de la máquina virtual donde se instalará el controlador *SDN* HP con el comando **ifconfig**, como se muestra en la **Fig. 1**.

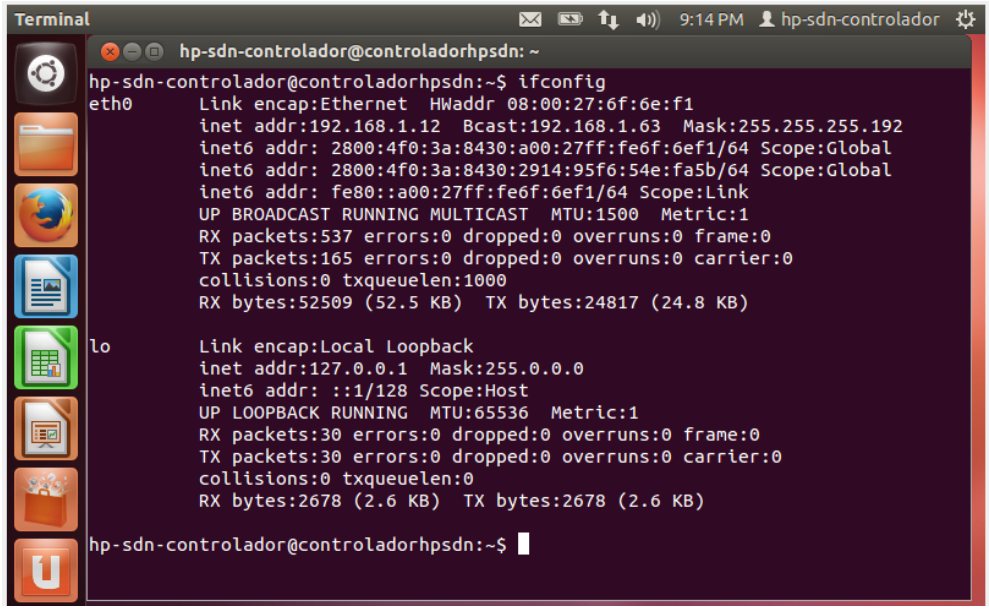

Fig. 1 **Dirección de las interfaces [Fuente propia]**

Instalar *SSH* para poder ingresar archivos utilizando *WinSCP* a la máquina virtual: **sudo apt-get install openssh-server**.

Iniciar el programa *WinSCP* desde una laptop o *PC* y conectarse utilizando la dirección *IP* de la máquina virtual donde se ha instalado Ubuntu, ingresar las credenciales de usuario y contraseña, como se muestra en la **Fig. 2**.

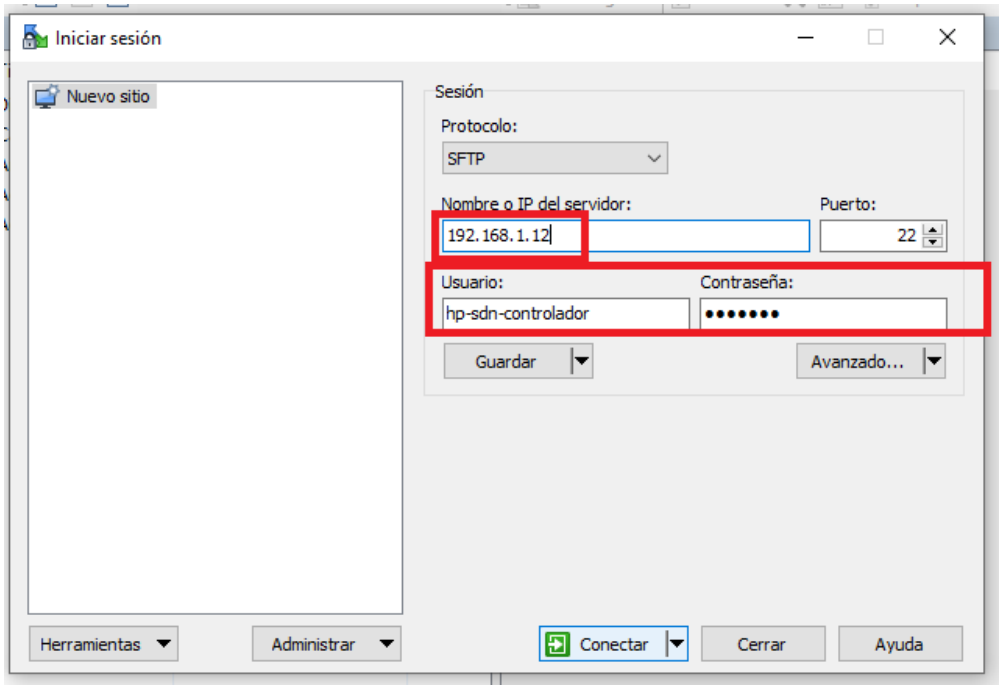

***Fig. 2* WinSCP [Fuente propia]**

Mover el archivo del controlador *SDN* HP 2.5.15 y el archivo de Flow Maker Deluxe 1.0.0 hacia el espacio de trabajo de la máquina con el sistema Ubuntu.

En la **Fig. 3** se visualizan los 2 archivos ya movidos al escritorio de la máquina virtual.

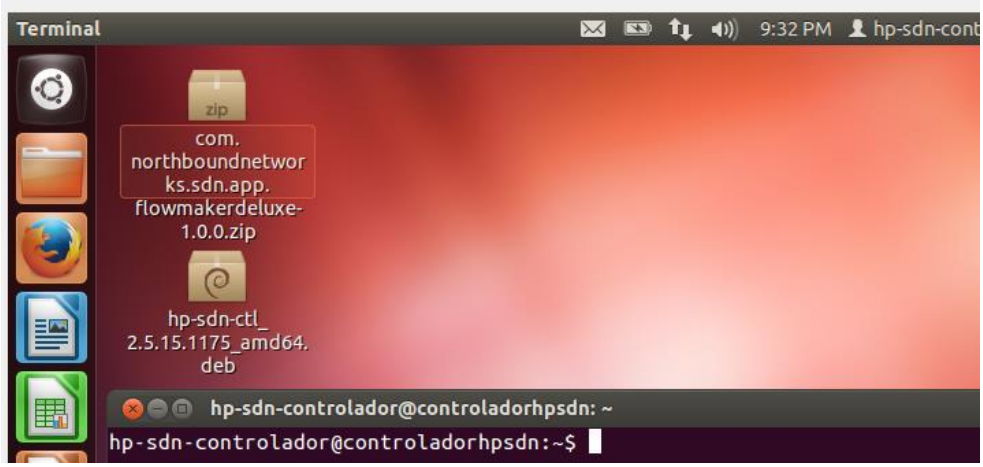

Fig. 3 **Archivos copiados en Ubuntu [Fuente propia]**

Se deben actualizar los paquetes del sistema antes de continuar con las instalaciones posteriores. **sudo apt-get update**.

Instalar el repositorio de Ubuntu con el siguiente comando: **sudo apt-get install ubuntu-cloud-keyring**.

Al instalar el repositorio de Ubuntu se debe añadir el archivo de la nube del cual se pueden descargar los ficheros para trabajar con el controlador. Para la versión 12.04.5 se debe utilizar icehouse, como se visualiza en la **Fig. 4**, el comando es el siguiente: **sudo add-apt-repository cloud-archive:icehouse**.

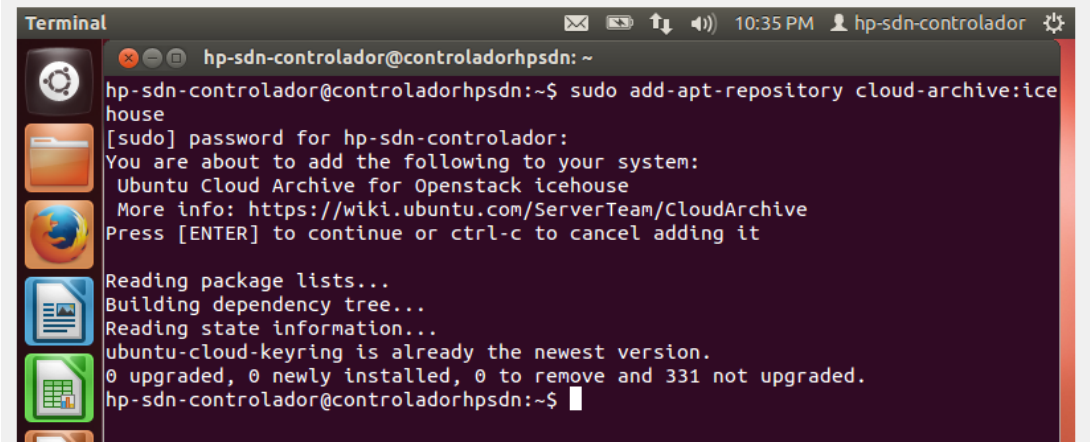

Fig. 4 **Repositorio de Ubuntu [Fuente propia]**

Del repositorio añadido en el paso anterior se obtienen los archivos para continuar con la instalación del controlador.

Keystone es un servicio OpenStack que proporciona autenticación de clientes *API*, para el descubrimiento de servicios y autorización distribuida de múltiples inquilinos mediante la implementación al ser instalado. En otras palabras, Keystone es el programa que funcionará como autentificador funciona con librerías *Python*. El comando es el siguiente: **sudo apt-get install keystone**.

Descomprimir el archivo de instalación del controlador SDN HP: **sudo dpkg --unpack hp-sdn-ctl_2.5.15.1175_amd64.deb**.

Al descomprimir el archivo de instalación del controlador, se denotan algunos errores que muestran que la máquina no posee los requerimientos mínimos del sistema para aquello se debe utilizar el comando touch. Los errores se muestran en la **Fig. 5**.

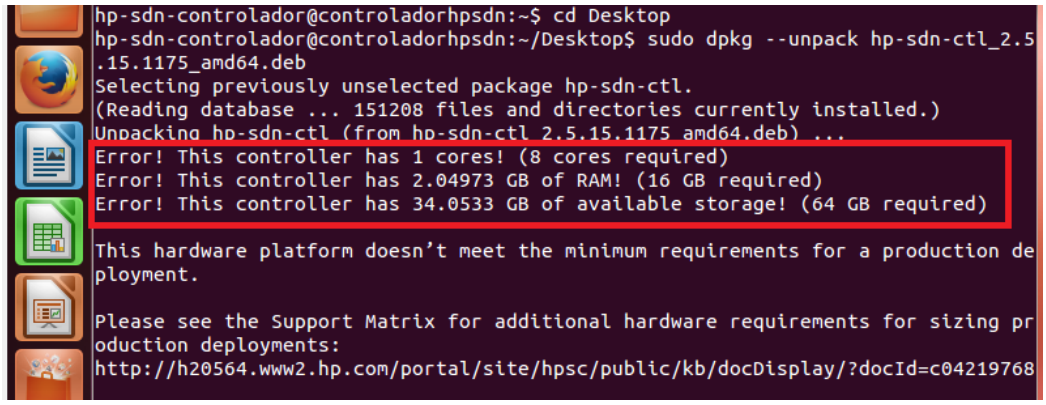

Fig. 5 **Error de descompresión [Fuente propia]**

El comando touch es la forma más fácil de crear archivos nuevos y vacíos. También se usa para cambiar las marcas de tiempo (es decir, fechas y horas del acceso y modificación más recientes) en los archivos y directorios existentes. **touch /tmp/override.txt** y descomprimir nuevamente el archivo del controlador, como se muestra en la **Fig 6**. **sudo dpkg --unpack hp-sdn-ctl_2.5.15.1175_amd64.deb**.

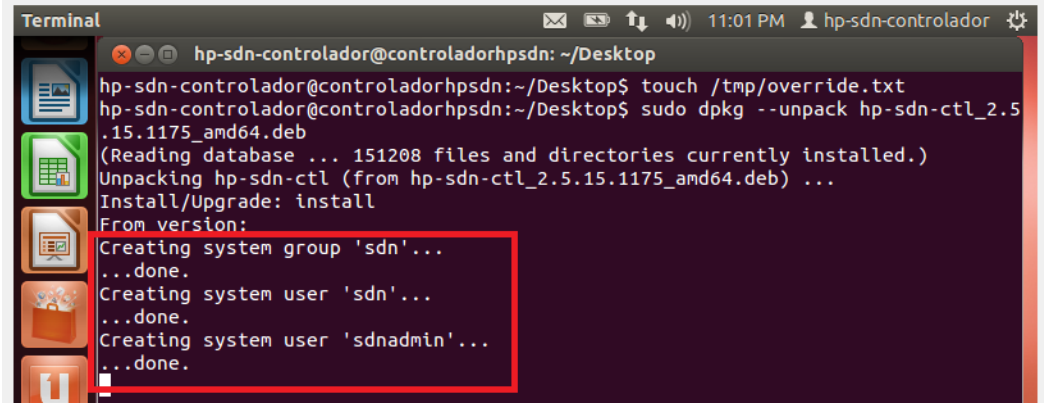

Fig. 6 **Descompresión del controlador [Fuente propia]**

Instalar los componentes descomprimidos del controlador como superusario. En la **Fig. 7** se muestra la instalación del controlador usando el comando: **sudo apt-get install -f**.

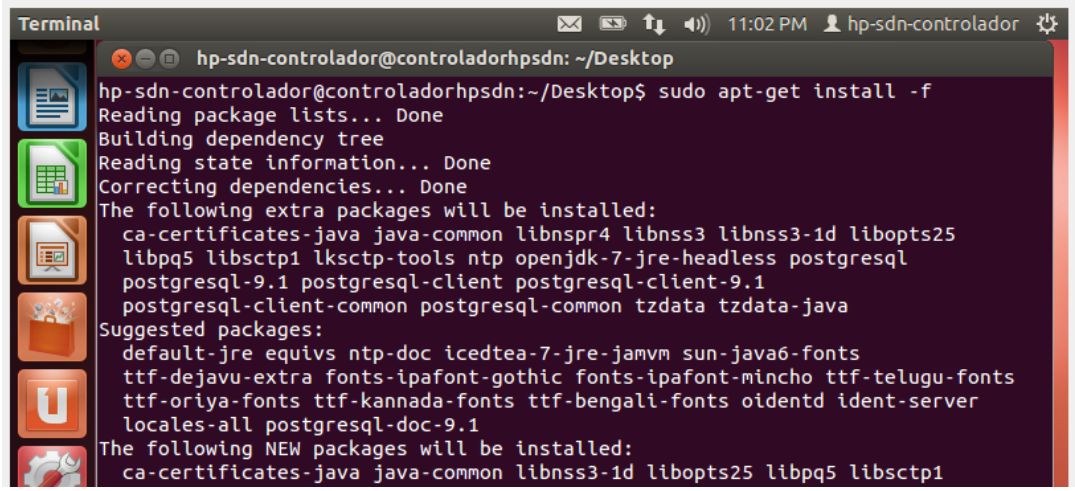

Fig. 7 **Instalación del controlador [Fuente propia].**

Se puede usar el comando **sudo dpkg -l hp-sdn-ctl** para verificar que el controlador fue instalado correctamente, debe observarse como en la siguiente imagen ya que significa que se instaló de manera correcta. Revisar el estado del servicio *SDN*: **sudo service sdnc status**.

Establecer las credenciales de usuario y contraseña del controlador. Iniciando el archivo de configuración de Keystone. En la **Fig. 8** se obtiene más información del comando efectuado. El comando a utilizar es: **sudo /opt/sdn/admin/config_local**.

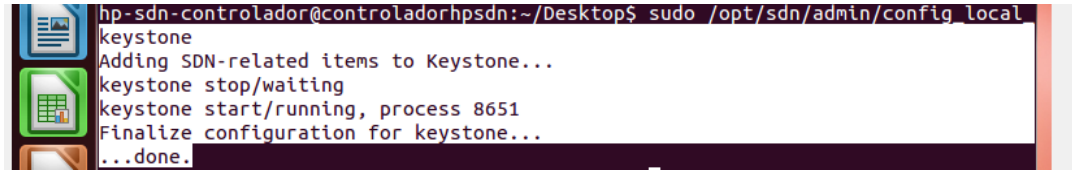

Fig. 8 **Ejecución de credenciales [Fuente propia]**

Usando el comando **sudo service sdnc status** se muestra el estado del servicio *SDN*, como se muestra en la **Fig. 9**.

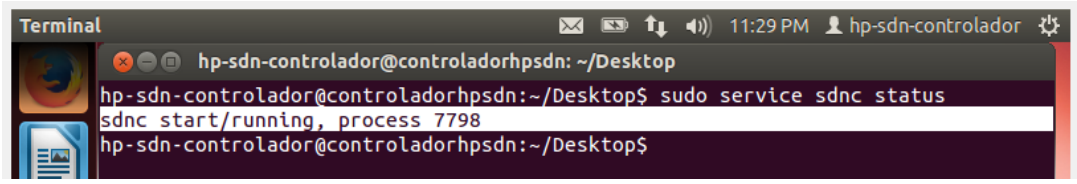

Fig. 9 Estado de SDN [Fuente propia]

Luego de iniciar las credenciales por default (defecto) con Keystone, las credenciales asignadas serán:

- **Usuario: sdn**
- **Contraseña: skyline**

Detener el servicio SDN del controlador, esto hará que cuando se inicie nuevamente el controlador las credenciales por defecto que fueron asignadas por Keystone puedan ser utilizadas para el ingreso por interfaz web. En la **Fig. 10** se muestra el uso del

comando “**sudo service sdnc stop**” y para iniciar se usa el comando: **sudo service sdnc start**.

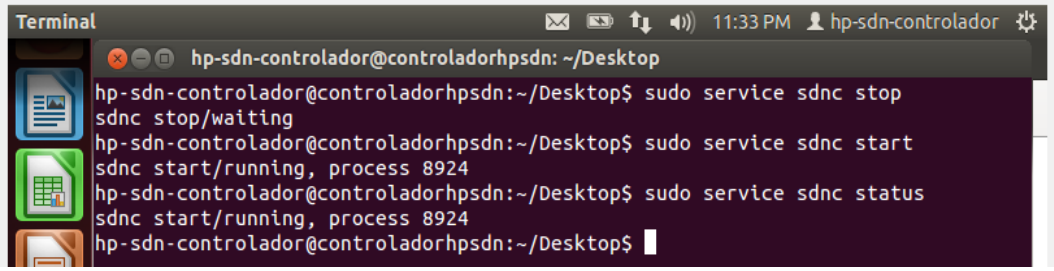

Fig. 10 **Detención e inicio del servicio SDN [Fuente propia]**

Para ingresar a las configuraciones del controlador se debe acceder utilizando el navegador web ya sea con la **dirección IP** de la máquina virtual donde se ha instalado el controlador o con **localhost** añadiendo el puerto de comunicación. Al no estar configurado el acceso a la interfaz web del controlador, debemos seleccionar añadir la excepción y confirmar la excepción de seguridad en el navegador web, como se muestra en la **Fig. 11**.

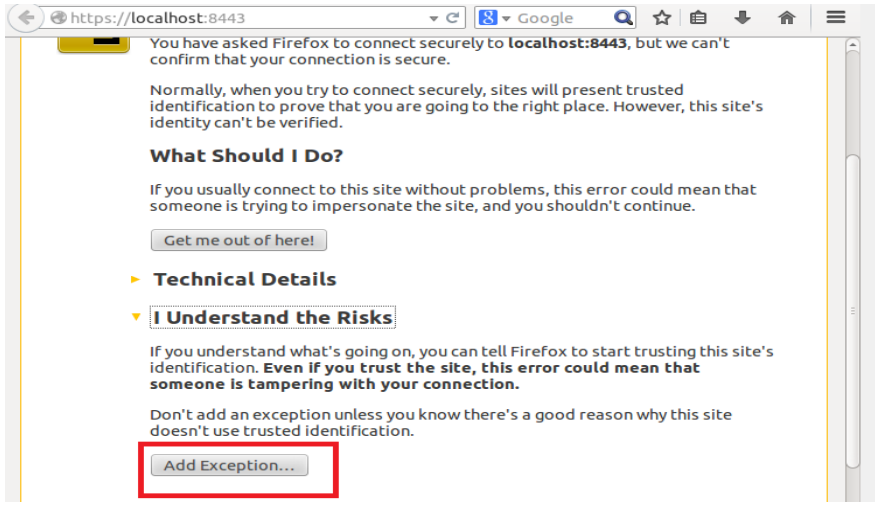

Fig. 11 **Excepción de seguridad web [Fuente propia]**

Luego de añadir la excepción se puede ingresar a la interfaz del controlador en el navegador. También se puede acceder al controlador desde otro computador accediendo con la dirección *IP* de este y añadiendo la excepción, como se muestra en la **Fig. 12**.

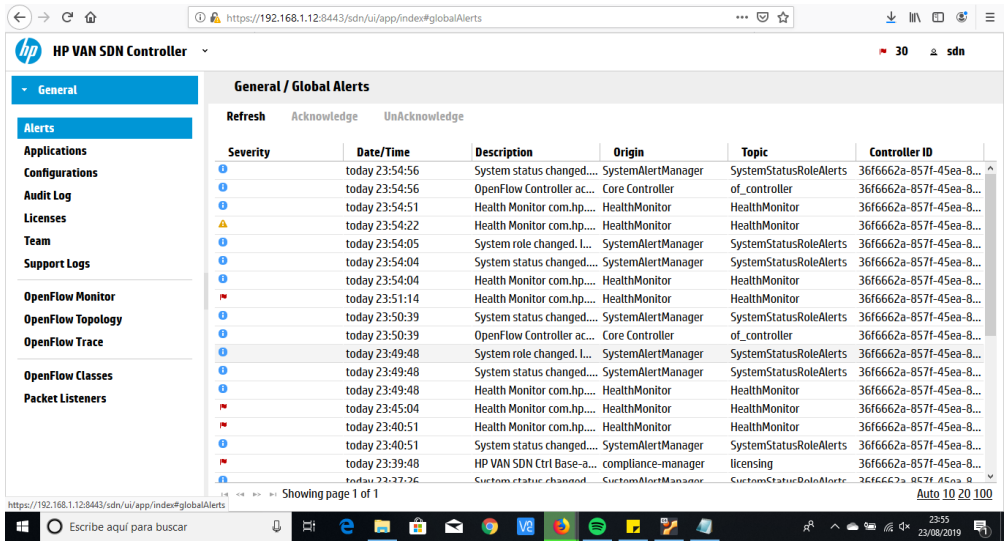

Fig. 12 **Interfaz web del controlador [Fuente propia]**

Modificar la configuración inicial del control seleccionando la opción **Modify**. En la **Fig. 13** se muestra más detalladamente.

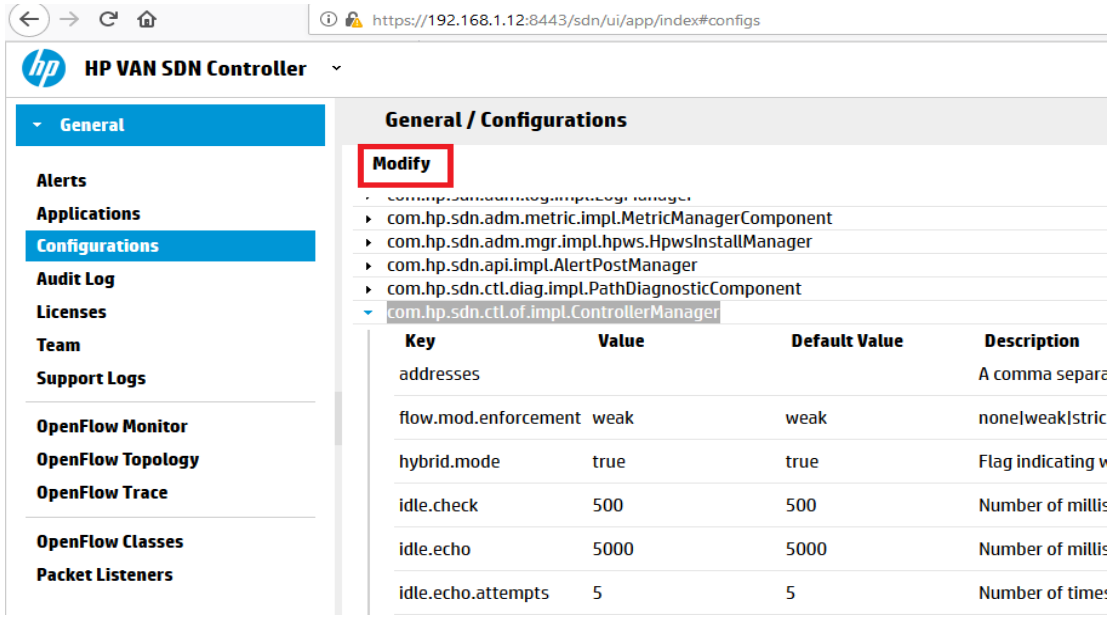

Fig. 13 **Configuración del controlador [Fuente propia]**

La configuración del modo híbrido determina qué decisiones de reenvío de paquetes son tomadas por los *switches OpenFlow* controlados y cuáles de estas decisiones las toma el controlador mismo.

- Si el modo híbrido está habilitado (la configuración predeterminada), el controlador delega el reenvío de paquetes normal a los conmutadores controlados, pero anula estos conmutadores para el reenvío de paquetes no estándar decisiones requeridas por las aplicaciones instaladas para tipos de paquetes específicos. En este modo, el controlador confía en los switches

controlados para resolver bucles y determinar rutas de reenvío mediante mecanismos de red tradicionales (como *STP*, *OSPF*).

- Si el modo híbrido está deshabilitado, el controlador toma las decisiones de reenvío para todos los paquetes en la red controlada por *OpenFlow*. En este estado, el controlador resuelve los bucles de red y determina los caminos de reenvío.

La gestión del modo híbrido incluye lo siguiente:

- Ver y (si es necesario) cambiar la configuración del modo híbrido.
- Coordinación del modo híbrido del controlador con la configuración del *switch OpenFlow*.

Establezca hybrid.mode en uno de los siguientes:

- Verdadero (predeterminado): habilita el modo híbrido. El controlador toma decisiones de reenvío de paquetes solo según lo requieran las aplicaciones instaladas.
- Falso: deshabilita el modo híbrido. El controlador toma todas las decisiones de reenvío.

Por lo que para esta práctica debemos cambiar el modo hibrido a falso, escribimos **false** en la casilla de **value**. En la **Fig. 14** se visualiza el cuadro que se debe de editar:

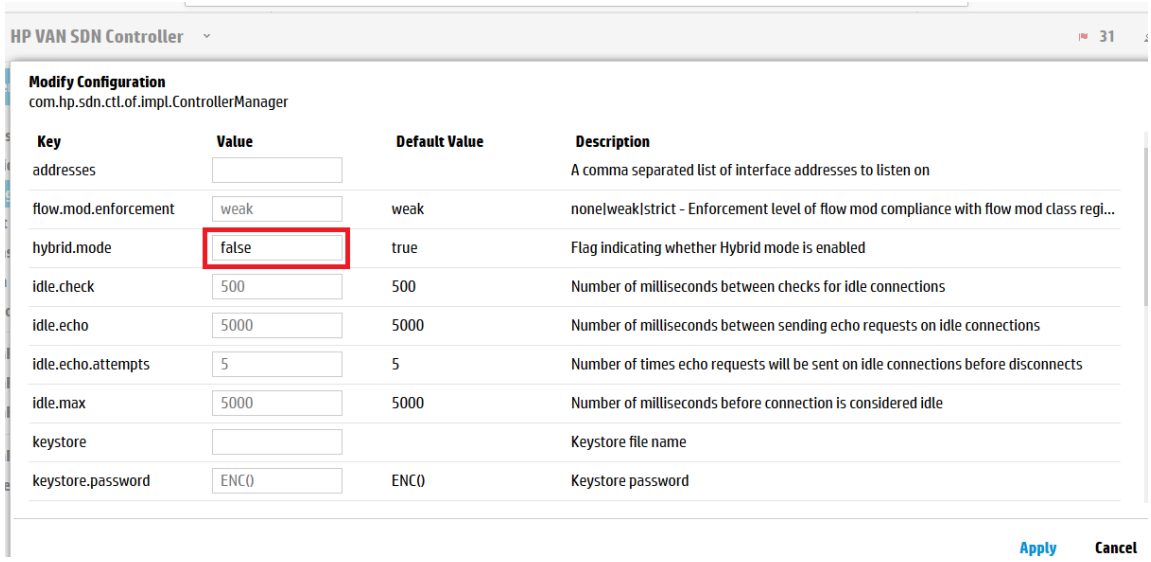

Fig. 14 **Cambio del modo hibrido [Fuente propia]**

Reiniciar el controlador como superusuario con el comando: **sudo service sdnc restart**.

El controlador *SDN* aplica la validación de firmas jar/zip de forma predeterminada para un ambiente experimental/desarrollo donde las aplicaciones sin firmar deben implementarse y la validación de jar/zip puede ser deshabilitada.

Debido a que el controlador aplica la validación mencionada, se debe de editar el archivo **dmk.sh** y deshabilitar la validación utilizando el editor “nano” para la modificación del archivo para poder instalar una aplicación en el controlador ya que, si

se intenta instalar una aplicación sin modificar el archivo, hay posibilidad de un error en la instalación, como se visualiza en la **Fig. 15**:

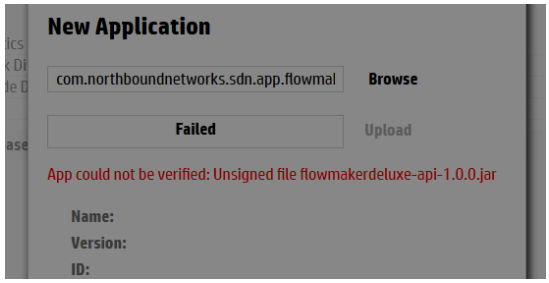

Fig. 15 **Error de instalación [Fuente propia]**

Detener el controlador con el comando "**sudo service sdnc stop**" y luego editar el archivo de configuración con el comando "**sudo nano /opt/sdn/virgo/bin/dmk.sh**" como se observa en la **Fig. 16**.

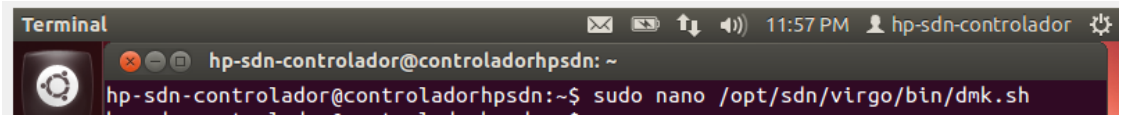

Fig. 16 **Edición de configuración [Fuente propia]**

Modificar el archivo /opt/sdn/virgo/bin/dmk.sh, como se observa en la **Fig. 17** y **Fig. 18**, para agregar la siguiente opción a la lista de JMX_OPTS: **-Dsdn.signedJar=none**. Por ejemplo:

- cd $KERNEL_HOME; exec $JAVA_EXECUTABLE \

- $JAVA_OPTS \
- $DEBUG_OPTS \
- $JMX_OPTS \
- -XX:+HeapDumpOnOutOfMemoryError \
- -XX:ErrorFile=$KERNEL_HOME/serviceability/error.log \
- -XX:HeapDumpPath=$KERNEL_HOME/serviceability/heap_dump.hprof \
- **-Dsdn.signedJar=none** \
- -Djava.security.auth.login.config=$AUTH_LOGIN \
- -Dorg.eclipse.virgo.kernel.authentication.file=$AUTH_FILE \

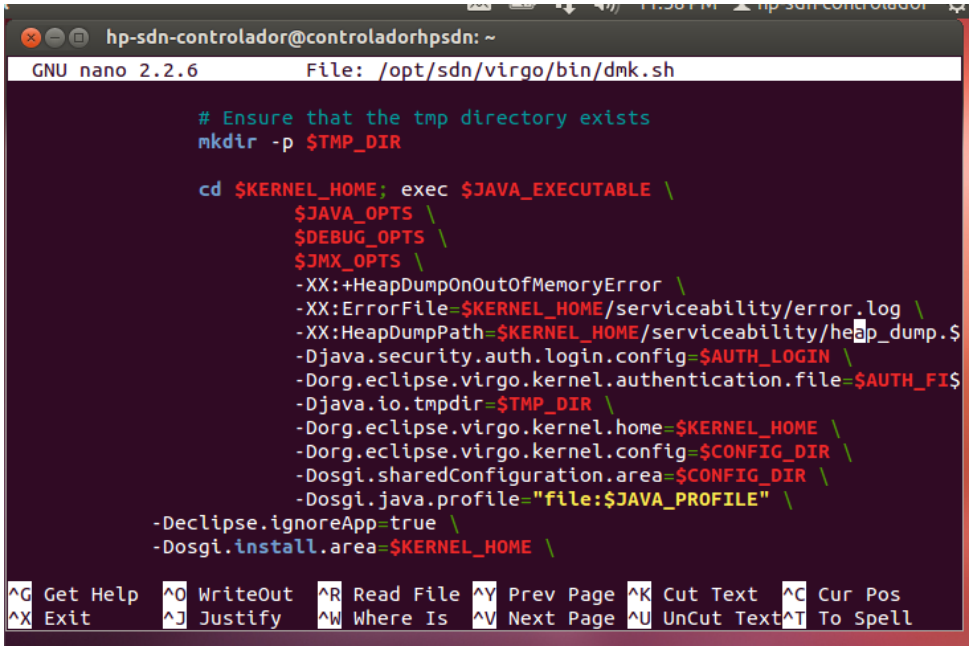

Fig. 17 **Archivo de configuración [Fuente propia]**

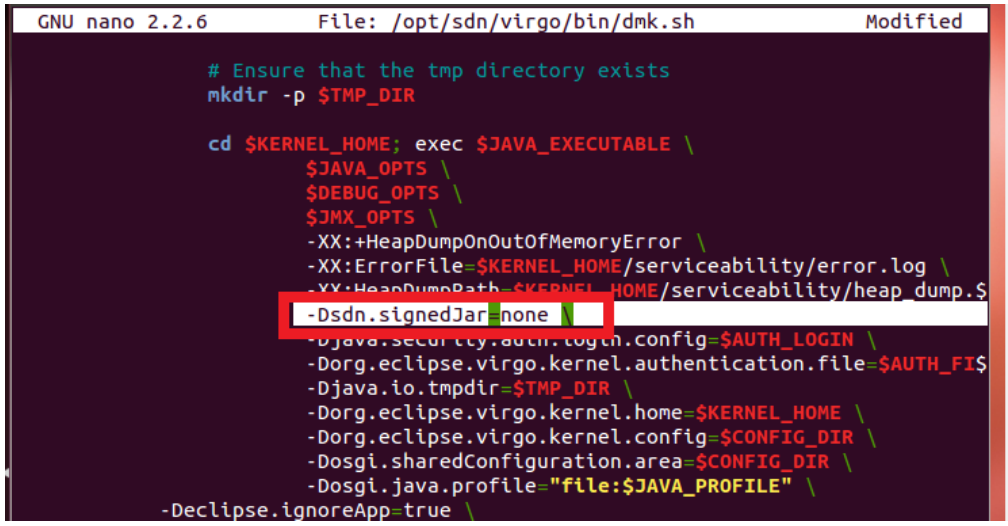

Fig. 18 **Cambio del archivo de configuración [Fuente propia]**

Guardar la configuración con la combinación de teclas **ctrl** + **o** y utilizar la combinación **ctrl** + **x** para salir del documento editado.

Luego de haber editado y guardado el archivo de configuración. Instalar la aplicación *Flow Maker* en el controlador, la cual es básicamente la forma más fácil de crear entradas de flujo directamente en el controlador y enviarlas a un *switch OpenFlow*. Todo se hace a través de la interfaz de usuario web, sin *API REST* y sin complicaciones.

Elegir la opción **Applications** y seleccionar **New** que permite instalar una nueva aplicación. La versión de la aplicación que se instalará es una versión Deluxe (pagada) esta versión es una opción más completa en comparación a la versión gratis de Flow Maker. Descargar la aplicación del repositorio e instalar como se visualiza en la **Fig. 19** y **Fig 20**.

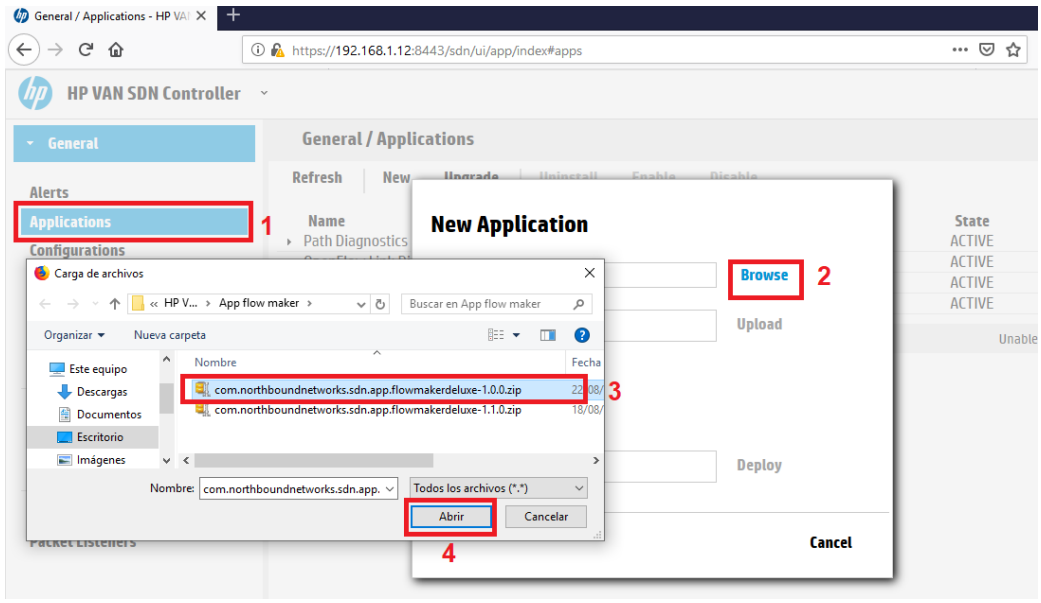

Fig. 19 **Instalación de la aplicación Flow Maker [Fuente propia]**

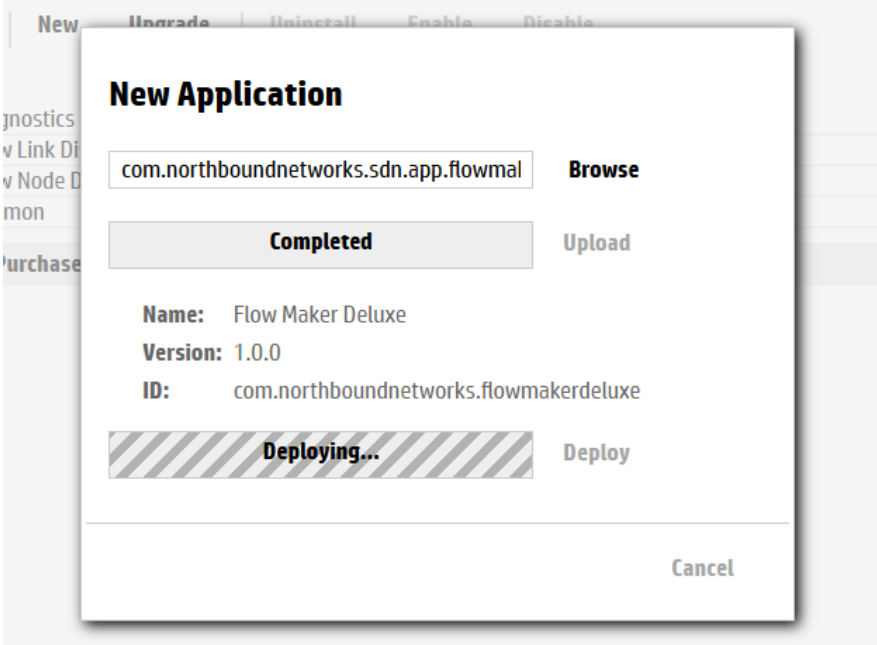

Fig. 20 **Instalación de Flow Maker validada [Fuente propia]**

Cuando haya finalizado la instalación de la aplicación, se debe de visualizar Flow Maker Deluxe entre las aplicaciones instaladas. En la **Fig. 21** se muestran las aplicaciones instaladas.

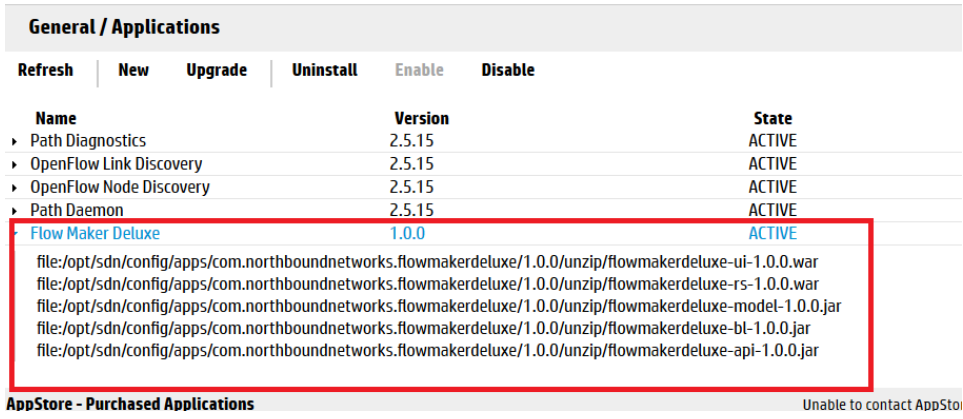

Fig. 21 **Aplicación Flow Maker Deluxe instalada [Fuente propia]**

Luego debemos conectar la tarjeta *Zodiac FX* al computador donde se encuentra instalada la máquina virtual por medio de cable *USB* y en el puerto # 4 de la tarjeta *Zodiac FX* conectarse por cable ethernet a la red en la cual estamos trabajando. Y abrimos el administrador de dispositivos.

En el administrador de dispositivos del equipo debemos revisar los puertos seriales conectados, estos se observan en la sección de Puertos (COM y LPT). En este caso la tarjeta *Zodiac FX* ha sido conectada en el puerto COM8. Abrimos el programa putty e ingresamos a la tarjeta *Zodiac FX* utilizando la conexión serial. En la **Fig. 22** se muestra la conexión serial a ingresar, el cual debe de ser el mismo que se evidencia en el administrados de dispositivos:

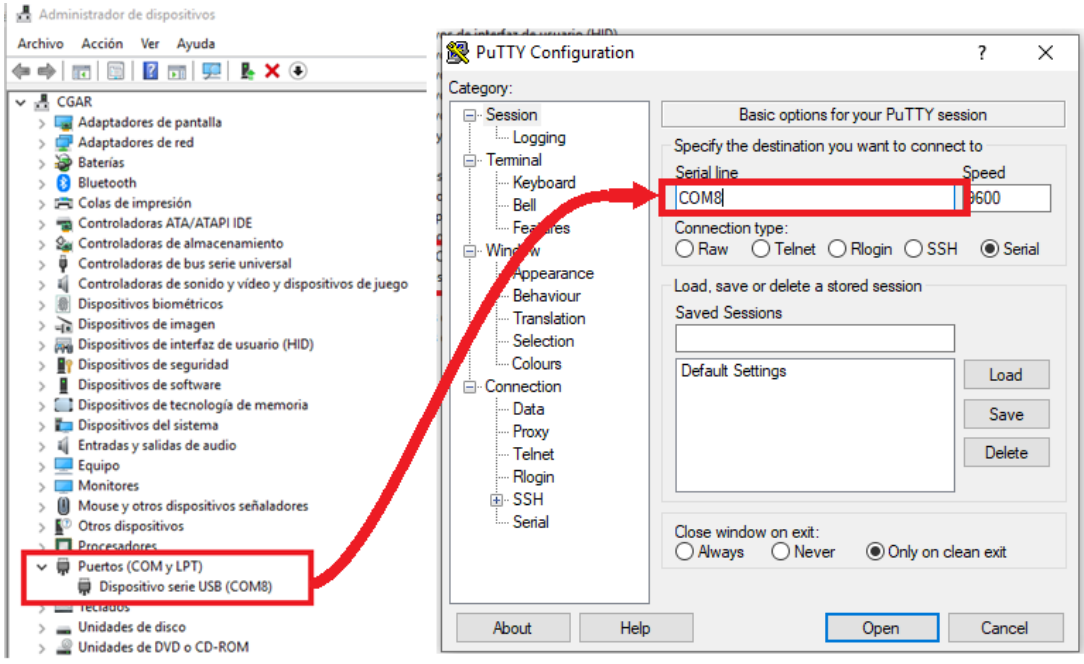

Fig. 22 **Acceso serial [Fuente propia]**

Como siguiente paso, restablecer los valores de fábrica para la tarjeta *Zodiac FX* como se hace en la **Fig. 23***:*

- Ingresar al modo de configuración de la tarjeta: **config**.

- Restablecer los valores: **factory reset**.

- Guardar la configuración: **save**.

- Reiniciar la tarjeta: **restart**.

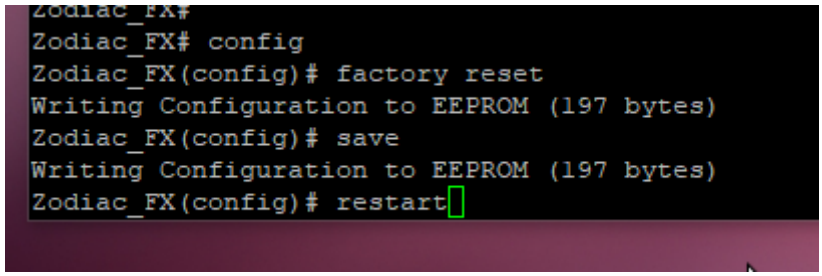

Fig. 23 **Restablecimiento de los valores de fábrica [Fuente propia]**

Desconectar el cable *USB* de la Tarjeta *Zodiac FX* del computador y volver a conectarlo e ingresar nuevamente a la interfaz de línea a de comando de la *Zodiac FX*. Configurar la máquina virtual con una dirección *IP* de manera manual, editar las conexiones de Ubuntu, colocar una dirección *IP* dentro del rango de direcciones en la red, añadir la máscara de red y la puerta de enlace. Todo esto se lo muestra en la **Fig. 24**.

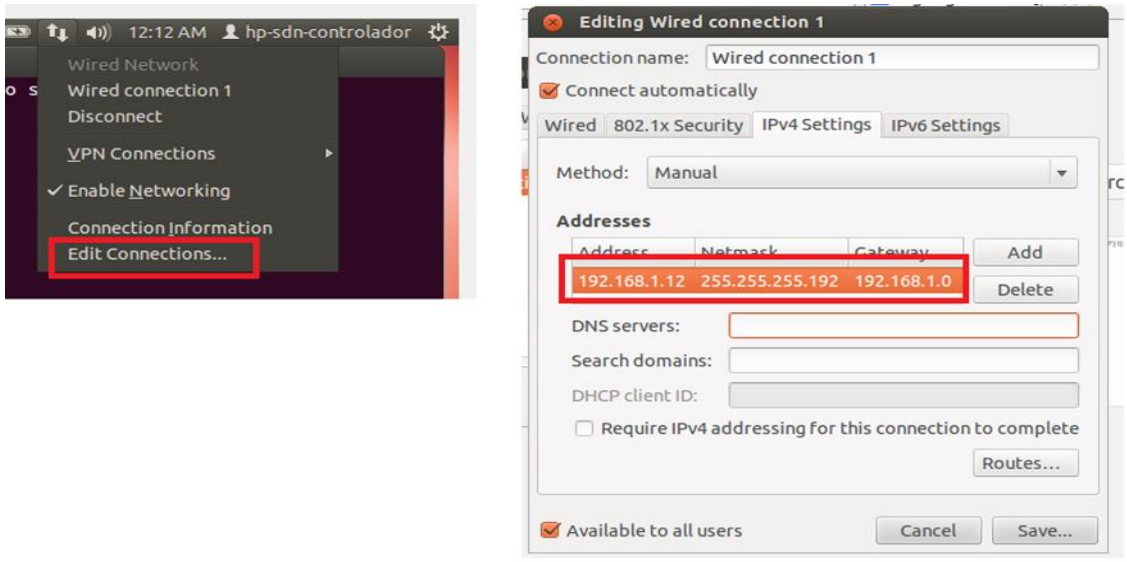

Fig. 24 **Configuración de red [Fuente propia]**

En la interfaz de línea de comando de la tarjeta *Zodiac FX* ingresar al modo de configuración y configurar la dirección *IP* del controlador para que sea detectada por la *Zodiac FX.* El comando a usar es el siguiente: **set of-controller (DIRECCIÓN IP DEL CONTROLADOR)**, como se muestra en la **Fig. 25**.

Guardar lo configurado y reiniciar.

```
Zodiac_FX# config
Zodiac_FX(config)# set of-controller 192.168.1.12
OpenFlow Server address set to 192.168.1.12
Zodiac_FX(config)# save
Writing Configuration to EEPROM (197 bytes)
Zodiac_FX(config)# restar
Unknown command
Zodiac_FX(config)# restart
```

Fig. 25 **Configuración OpenFlow de la tarjeta Zodiac FX [Fuente propia]**

Ingresar nuevamente con el programa putty a la interfaz de comando y verificar el cambio realizado dentro del modo de configuración con el comando **show config**.

A continuación, configurar y asignar una dirección *IP* a la tarjeta *Zodiac FX*, su puerta de enlace y la máscara de red. En la **Fig. 26** se muestra un ejemplo de esto. Los comandos a usar son:

- *IP* de la tarjeta Zodiac: **set ip-addresss (DIRECCION IP QUE SE VA ASIGNAR A LA TARJETA ZODIAC FX)**.

- Puerta de enlace: **set Gateway (DIRECCION IP DE LA PUERTNA DE ENLACE QUE SE VA ASIGNAR A LA TARJETA ZODIAC FX)**.

- Mascara de red: **set netmask (DIRECCION IP DE LA MASCARA DE RED QUE SE VA ASIGNAR A LA TARJETA ZODIAC FX)**.

- Guardar configuración: **save**.

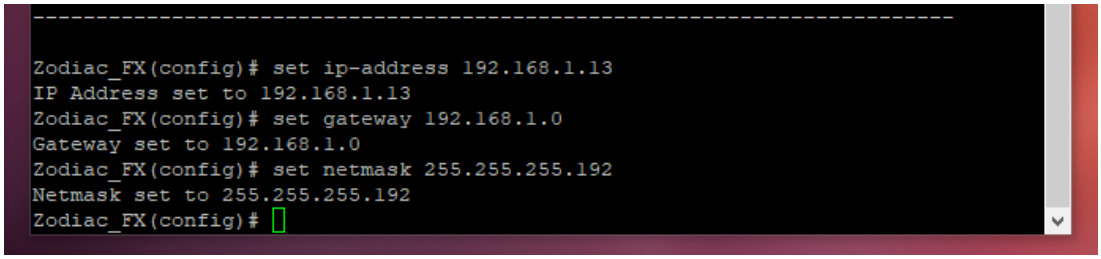

Fig. 26 **Configuración de red la tarjeta Zodiac FX [Fuente Propia]**

Verificar que los cambios realizados fueron guardados ingresando al modo de configuración y verificar con el comando **show config**.

En la interfaz del controlador observamos la dirección MAC de la tarjeta Zodiac FX, lo cual significa que la conexión se establece correctamente. En la **Fig. 27** se detalla lo mencionado:

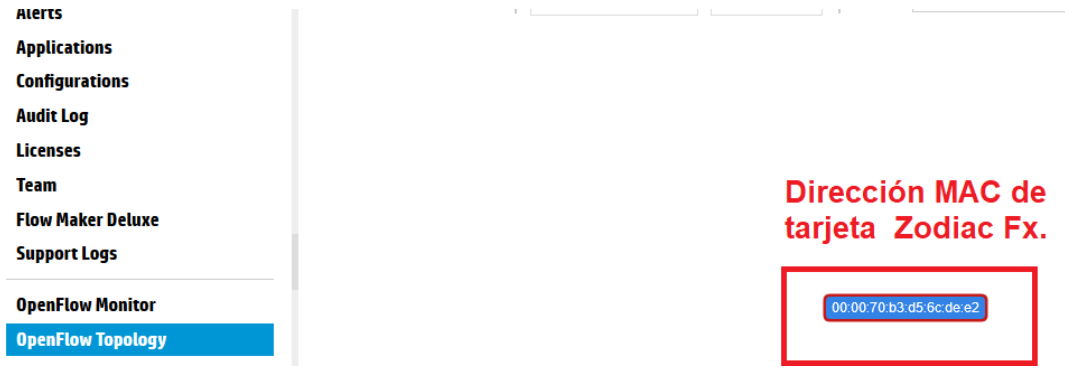

Fig. 27 **Zodiac FX detectada por el controlador [Fuente propia]**

Luego de haber revisado que los cambios realizados fueron guardados, se debe ingresar al controlador para corroborar que la tarjeta *Zodiac FX* fue detectada por el controlador con la opción de *OpenFlow* Monitor que muestra los puertos, flujos y grupos para la visualización de la ruta de datos para la ruta de datos seleccionada e identifica y detecta un *switch OpenFlow*.

Luego de haber sido detectada la tarjeta *Zodiac FX* por el controlador, se procede a crear o modificar flujos para rutas seleccionadas con la aplicación instalada Flow Maker Deluxe.

Debido a que se cambió el modo hibrido del controlador a falso las acciones en la tabla de flujo de datos se establecen con salida hacia el controlador.

Se muestra en la aplicación *Flow Maker Deluxe*: **Apply Actions**: **Output: Controller** y en la tarjeta *Zodiac FX* también se observa la configuración realizada, en la **Fig. 28** se demuestra lo mencionado anteriormente.

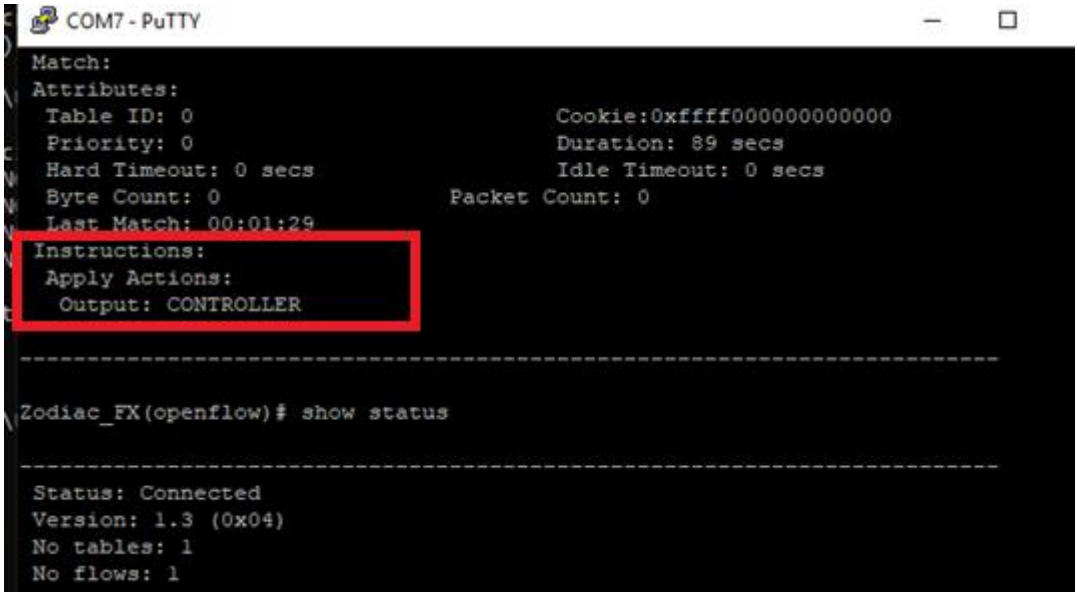

Fig. 28 **Configuración efectuada [Fuente propia]**

Luego de haber instalado los sistemas operativos en las *Raspberry Pi* que se utilizan como host. Conectar una *Raspberry Pi* al puerto # 1 de la tarjeta *Zodiac FX* y comprobar la dirección *IP* que fue asignada por la tarjeta *Zodiac FX* con el comando **ifconfig**.

Luego, hacer ping a la dirección *IP* de la *Raspberry Pi* que está conectada en el puerto # 1 de la tarjeta *Zodiac FX* para empezar a generar tráfico y aprender la funcionalidad de las tablas de flujo de datos a escribirse. Al hacer ping se observa un mensaje que indica **error en la transmisión. Error general**, lo cual se puede observar en la **Fig. 29**. Este error se debe a que al utilizar un controlador *SDN* la red debe ser programable por lo cual la escritura de flujos debe ser programada.

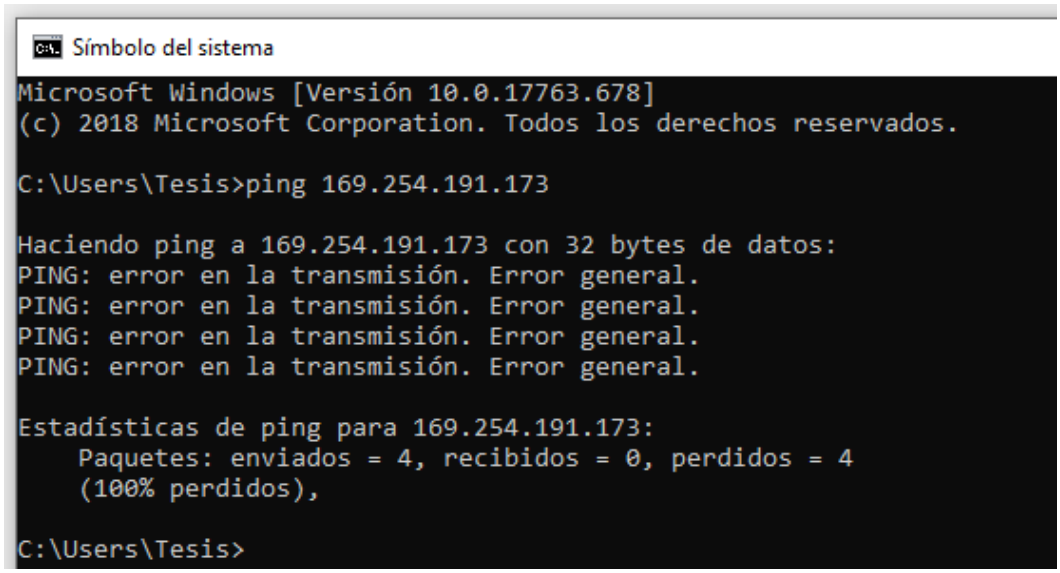

Fig. 29 **Error en la transmisión [Fuente propia]**

Sin embargo, aunque los mensajes no hayan sido intercambiados se genera tráfico en la red. Y al mostrar el estado en la interfaz de línea de comando en la tarjeta *Zodiac FX* ingresando al modo **OpenFlow** y utilizando el comando **show status**, se puede observar el tráfico generado con el intercambio de mensajes fallido.

En el puerto 1 se encuentra conectada la *Rapsberry Pi*. En la **Fig. 30** se visualizan los dispositivos conectados, formando así, la topología de red.

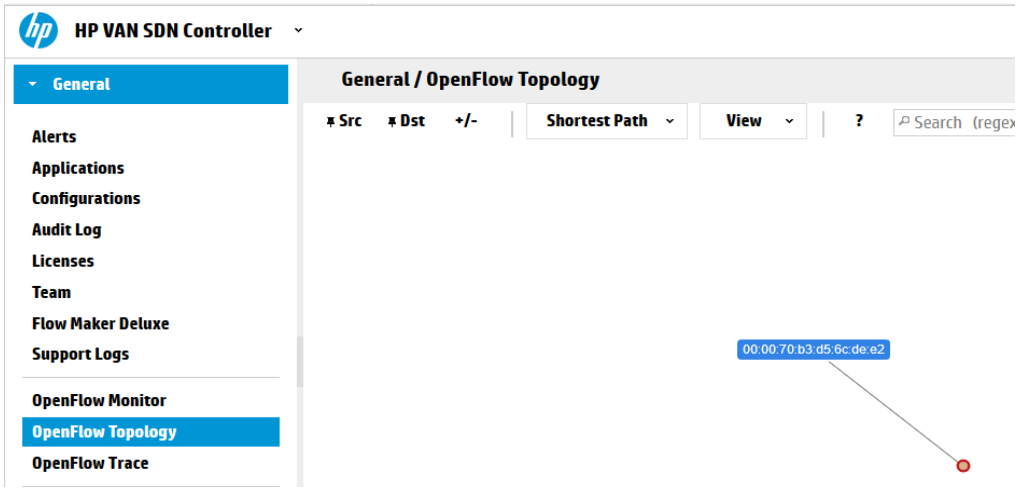

Fig. 30 **Topología de red [Fuente propia]**

Conectar el puerto # 3 a internet. Y observar como en la topología aparecen los equipos dentro de la red. En la **Fig. 31** se pueden observar más dispositivos en la topología debido a que se procedió a conectar el puerto # 3 de la *Zodiac FX* a un *router.*

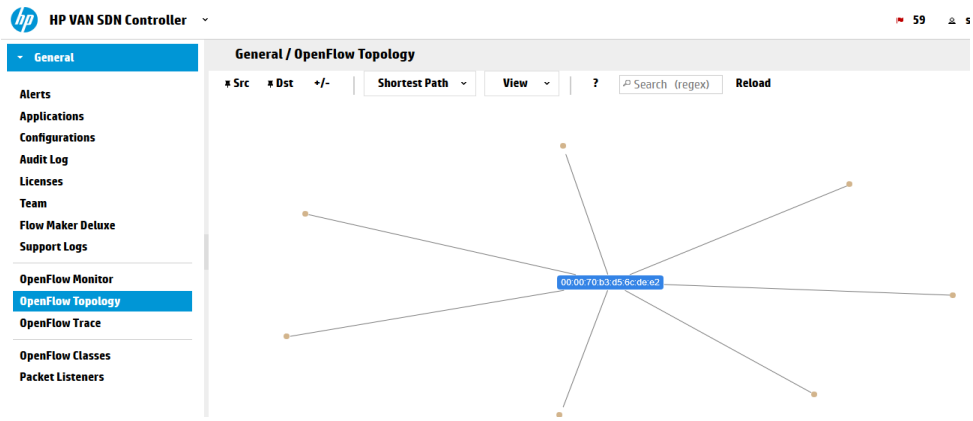

Fig. 31 **Zodiac FX conectada al router [Fuente propia]**

A continuación, conectar en el puerto # 2 de la tarjeta *Zodiac FX* la otra *Raspberry Pi* que actuara como segundo host y obtener la dirección *IP* asignada por la tarjeta *Zodiac FX.* En la **Fig. 32** se muestra el uso del comando **ifconfig** para verificar las direcciones *IP*.

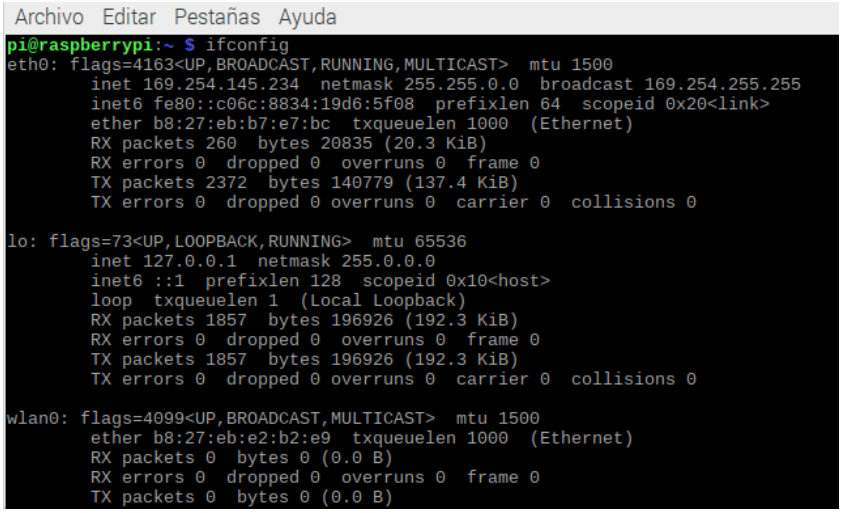

Fig. 32 **Raspberry Pi conectada a la tarjeta Zodiac FX [Fuente propia]**

Para poder tener comunicación entre ambas *Raspberry Pi* conectadas se deben configurar los flujos con la aplicación “Flow Maker Deluxe” de un puerto a otro. Abrir la aplicación y seleccionar la opción **Add** para añadir una configuración. Configuración de los campos son las siguientes:

- **Table ID**: Se escribe una identificación para la tabla sobre la cual se trabajará.

- **Priority**: El valor de la prioridad que se le asignará.

- **In Port**: El puerto en el cual se quiere realizar la configuración o programación.

- **Action**: Es la acción que se quiere ejecutar para una configuración entre las opciones esta la salida de un puerto (Output to Port).

- **Value**: Es el campo donde se especificará el valor de la acción que se quiere ejecutar por ejemplo el número de un puerto.

Seguir el ejemplo de la **Fig. 33**:

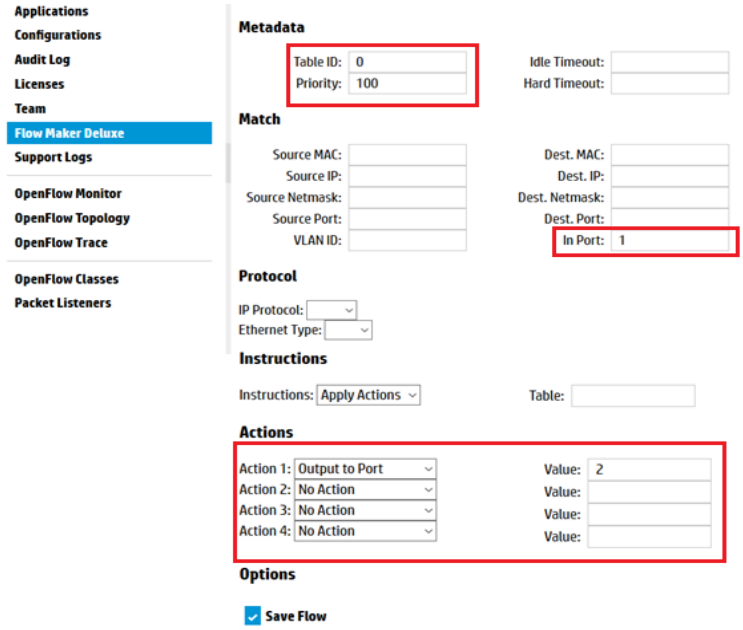

Fig. 33 **Configuración de una tabla de flujo de datos [Fuente propia]**

En la **Fig. 34** se muestra la tabla de flujo ya creada.

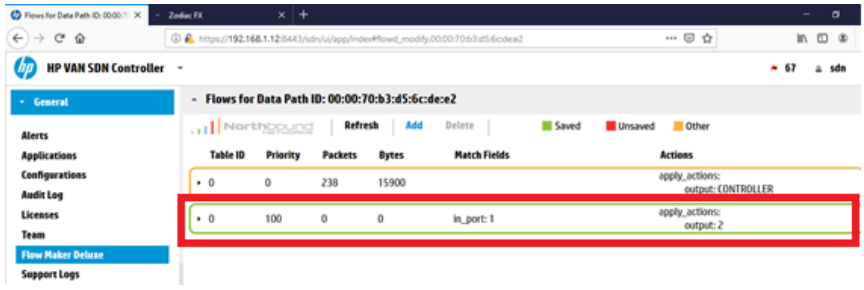

Fig. 34 **Tabla de flujo de datos creada [Fuente propia]**

En el siguiente paso de se debe realizar ping desde la *Raspberry Pi 3 Model B* conectada en el puerto 2 de la tarjeta *Zodiac FX* hacia la que está conectada en el puerto 1. Se muestra que al solo tener creada la configuración de comunicación desde el puerto

1 hacia el puerto 2, no hay posibilidad realizar el intercambio de mensajes y se visualiza un mensaje en el que muestra que el destino al que se hace ping es inalcanzable.

Para realizar ping al host 1 desde el host 2 se debe crear la configuración de comunicación desde el puerto 2 hacia el puerto 1.

Para visualizar que la creación de la configuración establece la comunicación entre ambos puertos dejar que se sigan enviado mensajes *ICMP* y solicitudes de mensajes como siguiente paso crear la configuración de comunicación. En la **Fig. 35** se muestran los mensajes cuando la comunicación se encuentra configurada y cuando no se encuentra configurada.

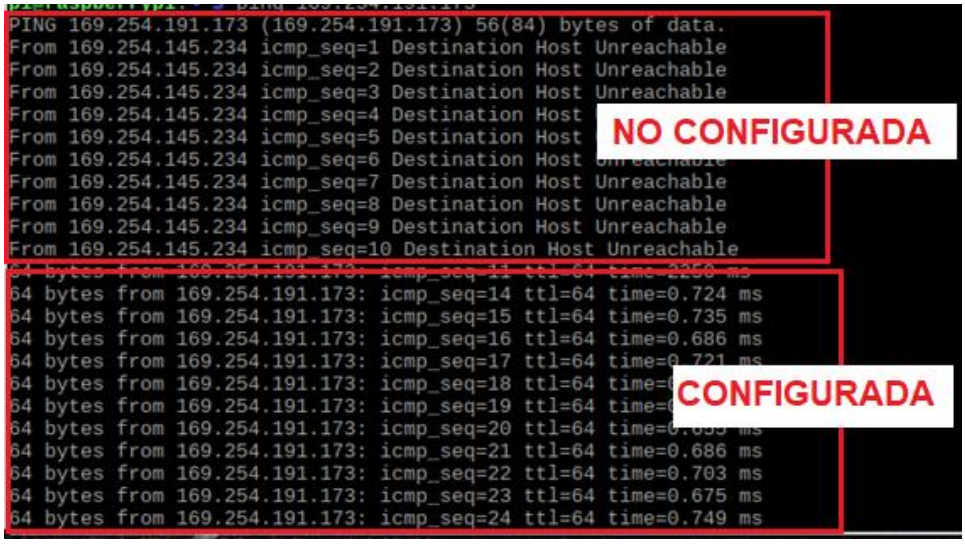

Fig. 35 **Red configurada y red no configurada [Fuente propia]**

Seguir los mismos pasos de la configuración anterior y llenar los mismos campos. Conectar a internet el puerto 3 de la tarjeta *Zodiac FX*. En la **Fig. 36** se tienen todos los puertos de la tarjeta Zodiac FX en el controlador.

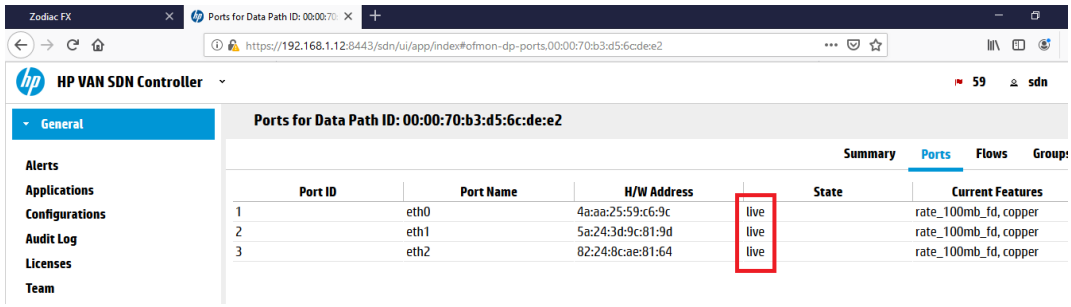

Fig. 36 **Puertos activos de la tarjeta Zodiac FX [Fuente propia]**

Como en los pasos anteriores crear 3 configuraciones de tabla de flujo para establecer comunicación:

- Desde el puerto 1 hacia el puerto 3.
- Desde el puerto 2 hacia el puerto 3.
- Desde el puerto 3 hacia los puertos 1 y 2.

Al crear estas configuraciones se tiene acceso a internet a través del puerto 3 de la tarjeta *Zodiac FX*. Luego de crear las configuraciones acceder al mismo tiempo desde

los hosts 1 y 2 a cualquier página web y anotar el resultado. En la **Fig. 37** se muestran las configuraciones creadas.

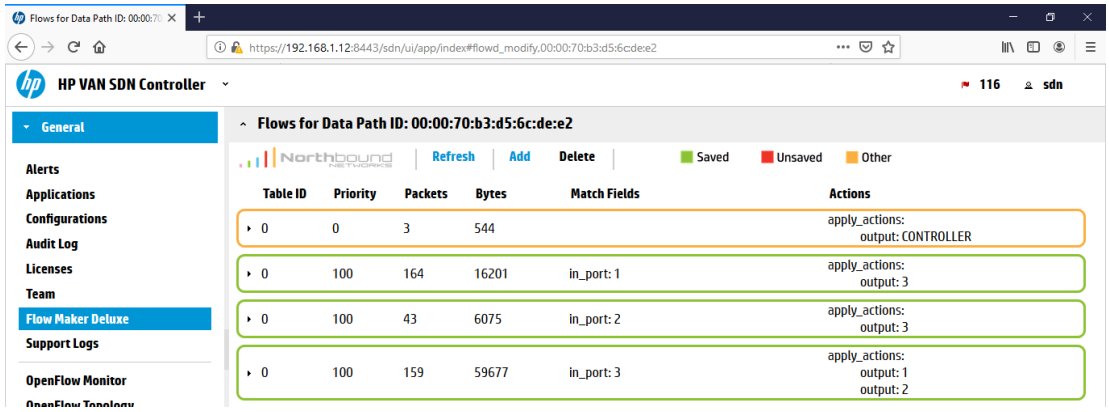

Fig. 37 **Configuraciones creadas [Fuente propia]**

Anexo F

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 6

RED DEFINIDA POR SOFTWARE UTILIZANDO OPENDAYLIGHT.

Objetivos:

- Realizar la conexión de la tarjeta *Zodiac FX* con el controlador OpenDaylight.
- Comprobar la comunicación entre los hosts conectados.

Instrucciones:

- Crear una máquina virtual con al menos 1 GB de RAM y 16 GB de disco para el óptimo funcionamiento del controlador.
- Habilitar en la configuración de la máquina virtual creada, en el apartado de redes, el Adaptador Puente.
- Instalar y configurar los programas necesarios para generar topologías.

Desarrollo:

Como dato curioso, al momento de instalar las librerías de *ODL* (OpenDaylight), no todas tiene soporte para la arquitectura *ARM*, por lo que se procedió a desarrollar la práctica mediante una máquina virtual. *ARM* es la arquitectura en la que está desarrollado el procesador de la *Raspberry Pi*.

Para esta práctica se debe utilizar la imagen ISO de Ubuntu Mate 16.04. Instalar el sistema operativo Ubuntu y configurar sus interfaces de red para que pueda conectarse por cable de red o Ethernet a la tarjeta *Zodiac FX* con dirección una dirección *IP* estática asignada y también tener conexión *WiFi* utilizando una dirección *IP* dinámica.

Luego de finalizar la instalación de Ubuntu. Seleccionar el apartado de red en la configuración de la máquina virtual.

Configurar el Adaptador 1 y 2 como un adaptador puente, seleccionar la conexión que provee la máquina. En la **Fig. 1** se muestra de una forma detallada.

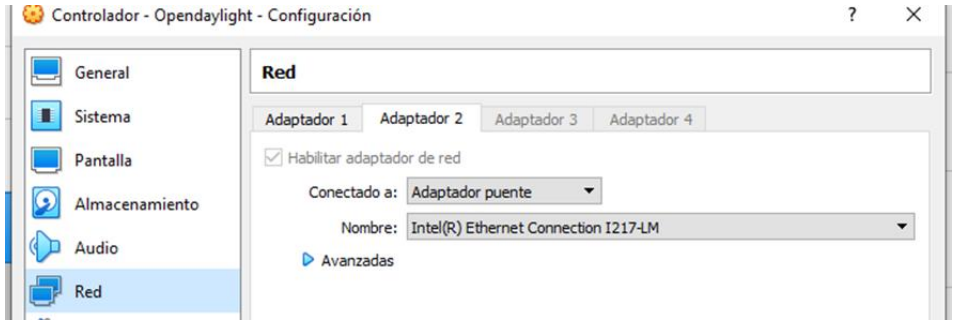

***Fig. 1* Configuración del adaptador puente [Fuente propia]**

Conectar un cable de red en el puerto 4 de la tarjeta *Zodiac FX* hacia el puerto ethernet del computador que contiene la máquina virtual. Ingresar al archivo de configuración de las interfaces de red mediante el siguiente comando: **sudo nano /etc/network/interfaces**.

Modificar y configurar las interfaces de red. Donde enp0s3 se refiere a la red en el adaptador 1 y enp0s8 se refiere a la red en el adaptador 2. Se especifica que la interfaz de red enp0s8 sea configurada de manera estática con una dirección *IP* en el mismo rango que la dirección *IP* de la tarjeta *Zodiac FX* y la interfaz de red enp0s3 es configurada por medio del protocolo *DHCP* el cual asignará una dirección *IP* dinámica a Ubuntu. En la **Fig. 2** se visualiza la configuración. Lo que se debe de agregar es lo siguiente:

- auto lo
- iface lo inet loopback
- auto enp0s3
- iface enp0s3 inet dhcp
- auto enp0s8
- iface enp0s8 inet static
- address 10.0.1.8

- netmask 255.255.255.0

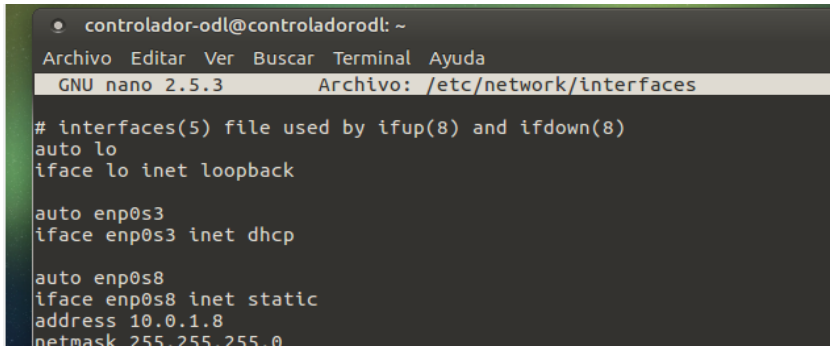

Fig. 2 **Archivo de configuración de redes [Fuente propia]**

Guardar con **Ctrl** + **o** y salir con **Ctrl** + **x**. Cuando se ha realizado la configuración, se debe reiniciar la máquina virtual para que los cambios guardados funcionen. En la **Fig. 3** se denota el uso del comando "**ifconfig**" para verificar las interfaces de red.

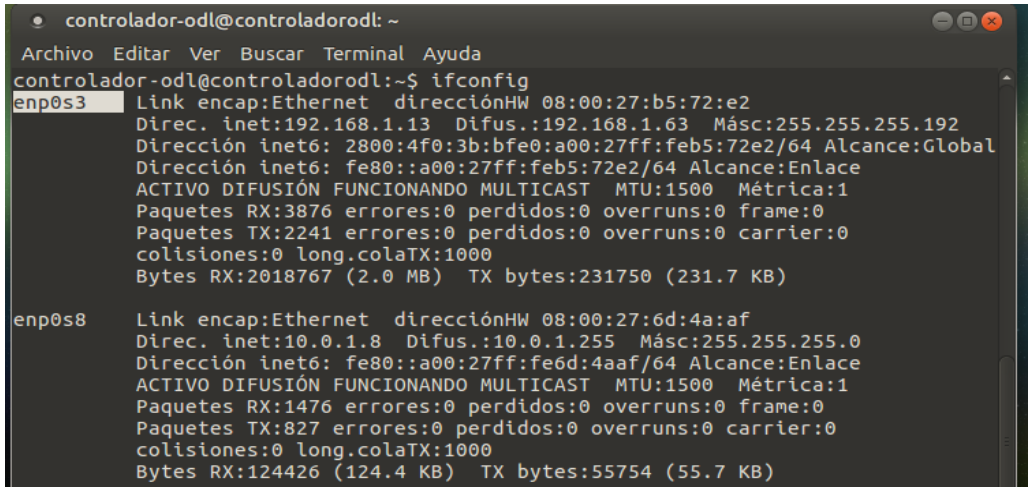

Fig. 3 **Interfaces de red [Fuente propia]**

Instalar el servidor *SSH* para poder hacer la conexión *SSH*, ya que en Ubuntu Mate no viene instalado *SSH* por defecto; usar el siguiente comando: **sudo apt-get install openssh-server**.

La distribución de *ODL* compatible con la tarjeta *Zodiac FX* es la distribución Lithium.

Obtener la dirección *IP* dinámica asignada, con el comando **ifconfig**. Descargar el archivo de Lithium, ubicado en el repositorio, desde un equipo externo y transferir el archivo descargado utilizando el programa *WinSCP* utilizando las credenciales configuradas. El archivo a descargado es de tipo “tar.gz”, de tamaño “278797452”. Este ya está optimizado para que funcione junto con la tarjeta *Zodiac FX*, actualizar los paquetes del sistema: **sudo apt-get update**.

Ya actualizados los paquetes, proceder a instalar los archivos necesarios para el funcionamiento del controlador: **sudo apt-get install -y bash-completion software-properties-common pythonsoftware-properties sudo curl ssh git**. En la **Fig. 4** se muestra el proceso de instalación de los archivos.

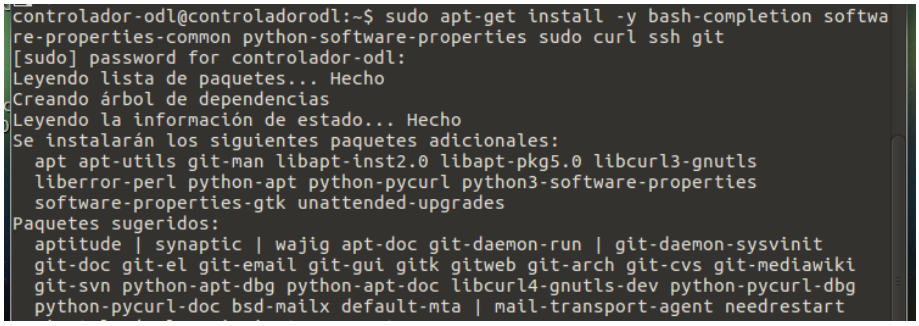

Fig. 4 **Instalación de archivos para ODL [Fuente propia]**

Ingresar al archivo de configuración de *SSH* para cambiar de estado la línea “PermitRootLogin” a “Yes” como se observa en la **Fig. 5**. acceder al archivo utilizando el comando: **sudo nano /etc/ssh/sshd_config**.

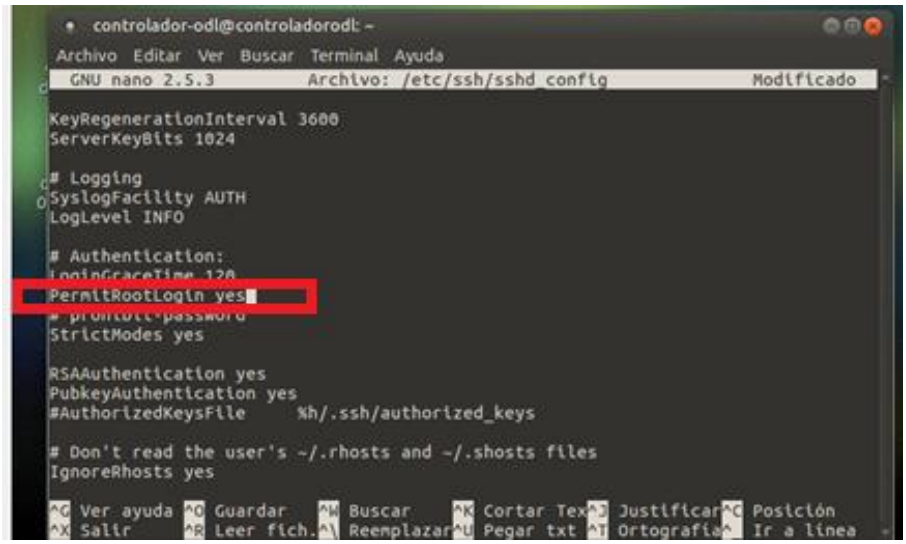

Fig. 5 Archivo de configuración de SSH [Fuente propia]

Guardar la configuración y salir, luego iniciar el servicio *SSH* con la siguiente línea de comandos: **sudo service ssh start** y luego **sudo ssh-keygen -t rsa -P ""**. Esto se verifica en la **Fig. 6**.

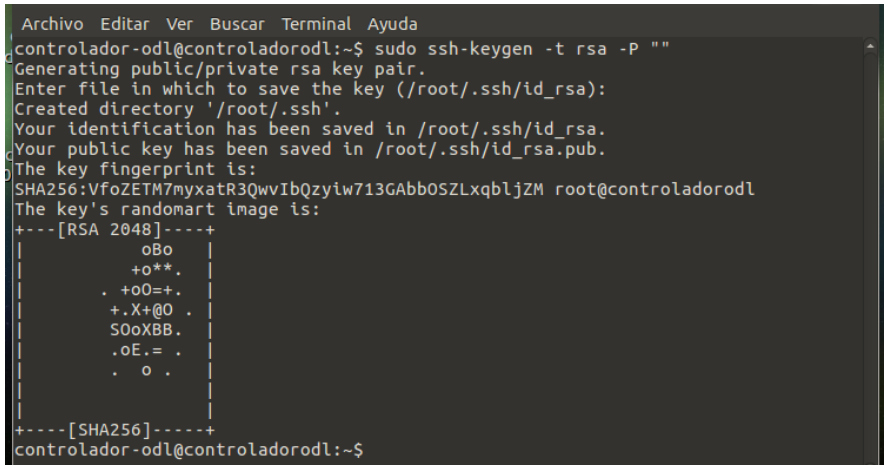

Fig. 6 **Contraseña generada [Fuente propia]**

Realizados los pasos anteriores, agregar el repositorio que contiene la versión de Java (en este caso Java en su versión 8) con la que se puede utilizar el controlador ODL: **sudo add-apt-repository ppa:webupd8team/java**. En la **Fig. 7** se visualiza la adición del repositorio.

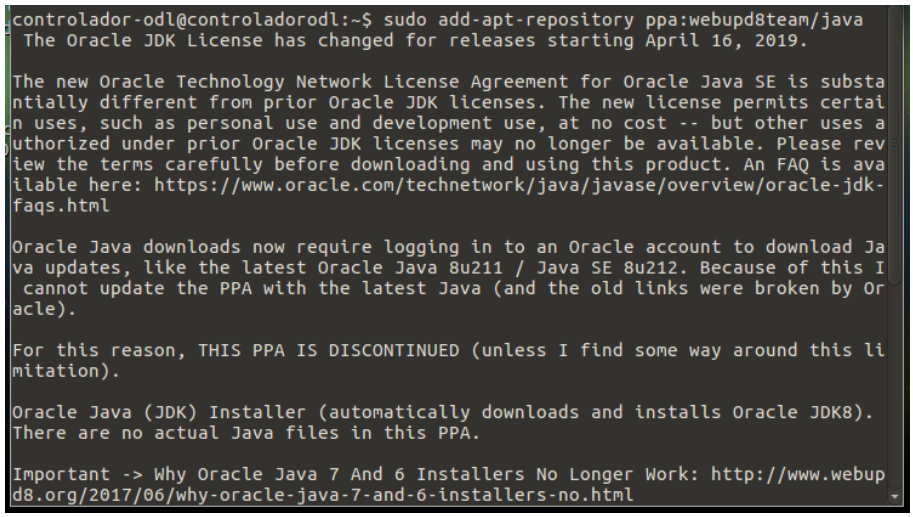

Fig. 7 **Repositorio de Java [Fuente propia]**

Realizar nuevamente una actualización de los paquetes con: **sudo apt-get update**.

Proceder a la instalación de Java en su versión 8, como se visualiza en la **Fig. 7**, el repositorio en el que se alojaba Java fue actualizado, motivo por el cual, no se puede agregar Java desde la ruta agrega anteriormente. Entonces, proceder a instalar la versión *JDK* (Java Development Kit) 8 de Java con el siguiente comando: **sudo apt-get -y install openjdk-8-jre**. Después, ingresar la ruta del directorio personal para poder exportar la variable de entorno de Java: **sudo nano ~/.bashrc**.

Como se muestra en la **Fig. 8**, en la parte final del texto del directorio se procede a la declarar la variable de entorno de java necesaria para el funcionamiento de ODL, lo recomendable es buscar manualmente la carpeta en la que se aloja el archivo JAVA_HOME.

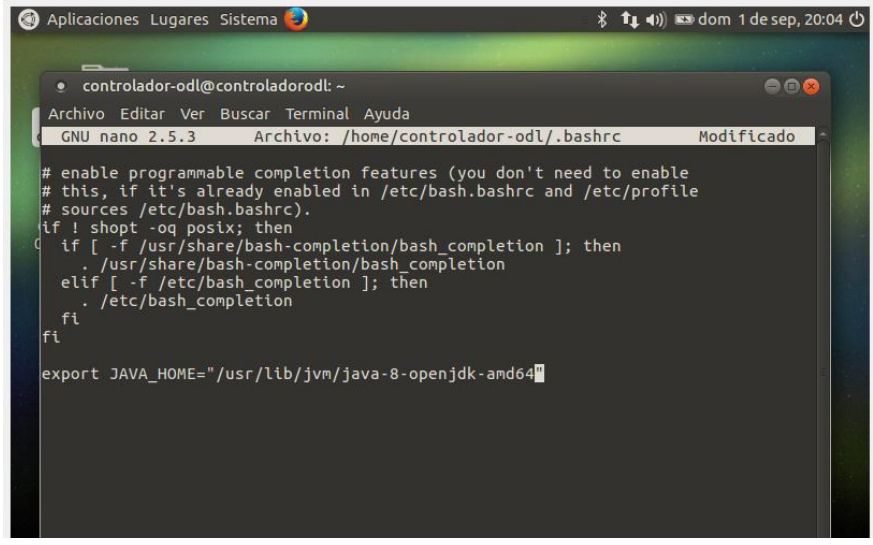

Fig. 8 **Directorio personal de la máquina virtual [Fuente propia]**

Guardar y salir del directorio, inmediatamente verificar la configuración realizada, como se tiene en la **Fig. 9**, con los siguientes comandos: **. ~/.bashrc** y después **echo $JAVA_HOME**.

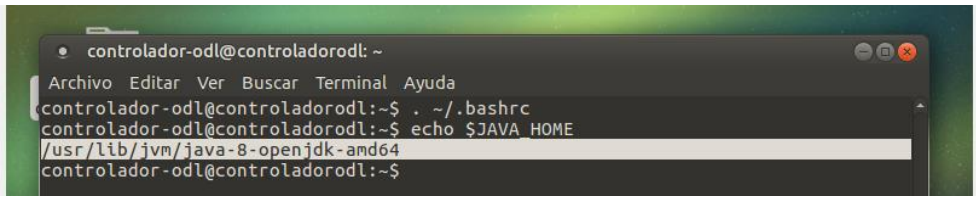

Fig. 9 **Variable de entorno JAVA_HOME declarada en el sistema [Fuente propia]**

Todo está preparado para proceder a instalar el controlador. Abrir la terminal, situarse en el directorio del escritorio donde se encuentra el archivo descargado del repositorio **cd Escritorio** y luego ejecutar el siguiente comando para descomprimir dicho archivo: **sudo tar zxvf distribution-karaf-0.3.0-Lithium.tar.gz**.

Realizada la descompresión del archivo, ingresar a la carpeta donde se descomprimió el archivo: **cd distribution-karaf-0.3.0-Lithium**. Ejecutar el controlador para que se muestre el logo del controlador, como se tiene en la **Fig. 10**, con la siguiente línea de comando: **./bin/karaf**.

Fig. 10 **Controlador OpenDaylight [Fuente propia]**

Esperar un momento para que se ejecute el controlador y debe aparecer el logo del controlador *OpenDaylight*, esto significa que está correctamente instalado y ejecutado, incluyendo todas sus dependencias, pero por el momento *OpenDaylight* no cuenta con ningún complemento instalado. En la **TABLA I** se indica todos los complementos que deben ser instalados.

TABLA I

Complementos necesarios de ODL [Fuente propia]

Nombre	Descripción
APIs de REST	Hace referencia a la instalación de un grupo de pasos que autorizan la comunicación entre el controlador y las aplicaciones de red.
Plugin de OpenFlow	Afirma el sustento del protocolo OpenFlow en sus versiones más recientes. Para el caso de ODL, en la versión Lithium, se asegura el soporte desde la versión 1.0 hasta la 1.3 (las cuales son versiones estables) y un soporte adicional a la versión de prueba 1.4. Además, el plugin de OpenFlow le revela al controlador el puerto TCP/UDP de comunicación con el dispositivo de red, denominado puerto OpenFlow. En el caso de ODL, sería el puerto 6633.
L2 switch	Testifica la comunicación con cualquier tipo de conmutador de capa 2, precisamente autorizado para OpenFlow.

SAL	El servicio de abstracción de capa (SAL) se usa para extirpar información de red que, notificada por el protocolo OpenFlow, no pertenece a instrucciones del conmutador. Con este servicio es viable que el controlador extirpe estadísticas sobre el estado de la red y características de esta, como la topología.
Herramientas de Yang	Son un conjunto de herramientas y bibliotecas que toleran aplicaciones, que están basadas en Java, en el controlador ODL. Fundamentalmente, es el complemento que permite la comunicación entre el controlador y las aplicaciones de red.
D-Lux	Provee la interfaz gráfica web donde se observa la configuración del controlador e información sobre estadísticas recopiladas.

Para instalar los complementos que se utilizarán, incluyendo la interfaz web de ODL, ingresar los siguientes comandos: **feature:install odl-restconf-all odl-openflowplugin-all odl-l2switch-all odl-mdsal-all odl-yangtools-common odl-dlux-all**.

En la **Fig. 11** se observan loa complementos de *ODL* necesarios para que el controlador funcione:

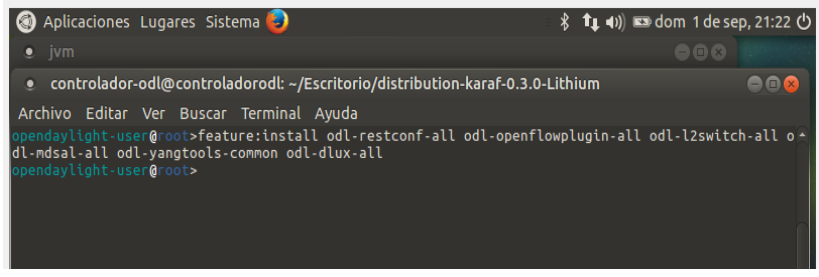

Fig. 11 **Instalación de complementos de ODL [Fuente propia].**

Esperar a que todos los complementos se instalen y después ingresar a la interfaz web del controlador.

Para ingresar, se debe de escribir la siguiente dirección *URL* en el navegador: **http://localhost:8181/index.html#/login**.

Las credenciales por defecto son:

- Usuario: admin

- Contraseña: admin

Por lo general en el primer ingreso aparece un error, pero no significa que hay error en la instalación significa que aún se están cargando los complementos hay que esperar e intentar nuevamente. En la **Fig. 12** se muestra la página de autenticación del controlador y en la **Fig. 13 se** observa la página de inicio de la interfaz gráfica del controlador *ODL*:

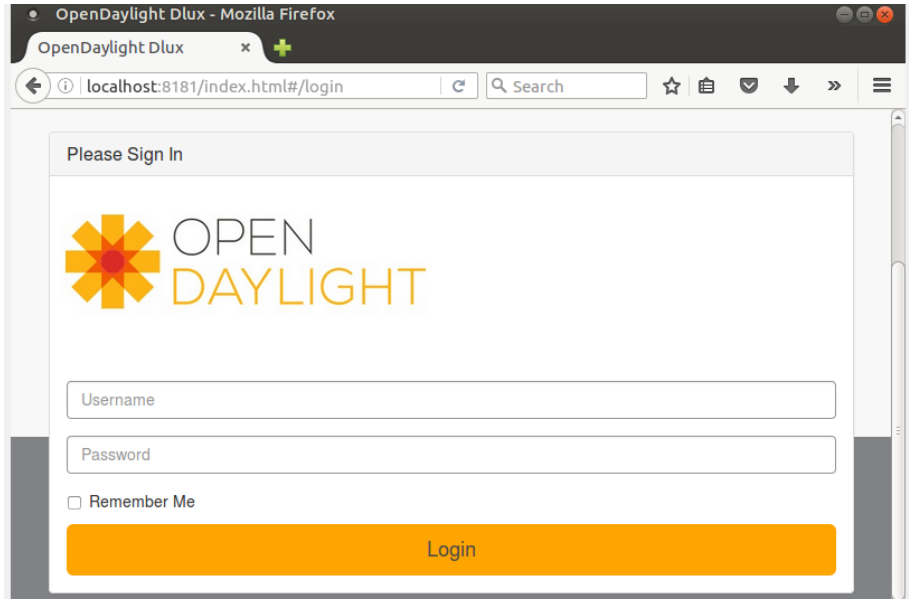

Fig. 12 **Página de autenticación ODL [Fuente propia]**

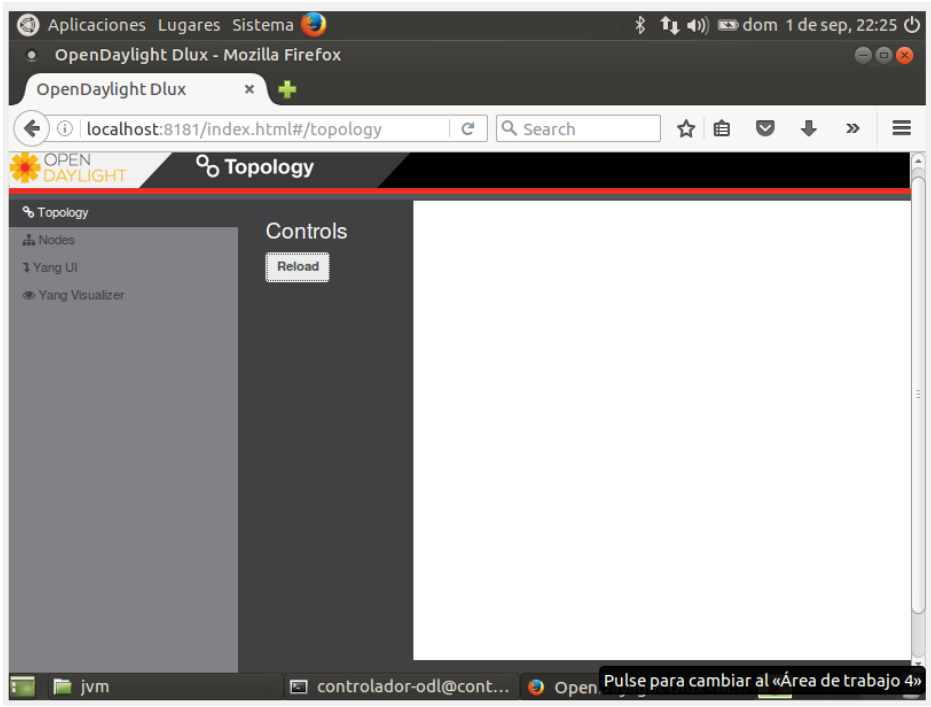

Fig. 13 **Interfaz gráfica web de ODL [Fuente propia].**

En la interfaz web se deben utilizar los dos primeros apartados, los cuales son “Topology” que sirve para visualizar la topología de la red o plano de datos y “Nodes” muestra los dispositivos que están conectados al controlador, junto con las estadísticas de los mismos.

Proceder a instalar ciertas funciones que se deben de usar más adelante. Se descarga “git” con el comando: **sudo apt-get install git**.

Instalar el sistema de gestión de paquetes “npm” con el siguiente comando: **sudo apt-get install npm**.

Instalar “nodejs-legacy”, Node.js es una plataforma basada en los archivos JavaScript de Chrome para crear fácilmente aplicaciones de red rápidas y escalables: **sudo apt-get install nodejs-legacy**.

En la **Fig. 14** se visualiza la clonación de archivos para el controlador. Clonar los archivos de la aplicación OpenDaylight-OpenFlow-app desde el siguiente repositorio de esta forma: **sudo git clone https://github.com/CiscoDevNet/OpenDaylight-OpenFlow-App**.

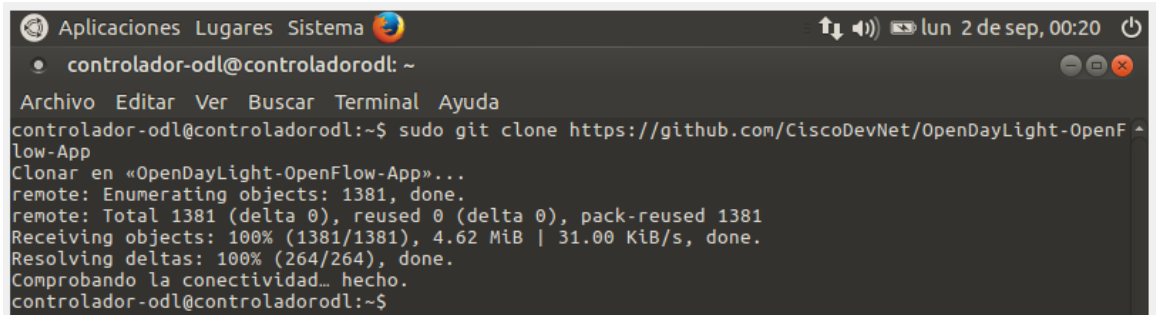

Fig. 14 **Clonación desde repositorio [Fuente propia].**

Acceder a la carpeta de OpenDaylight-OpenFlow-App e instalar lo siguiente con el comando: **sudo npm install -g grunt-cli**.

Ingresar al siguiente archivo para cambiar ciertas configuraciones: **sudo nano ./ofm/src/common/config/env.module.js**.

Dentro del archivo cambiar la opción de “baseURL” por la dirección *IP* dinámica del controlador como se muestra en la **Fig. 15**:

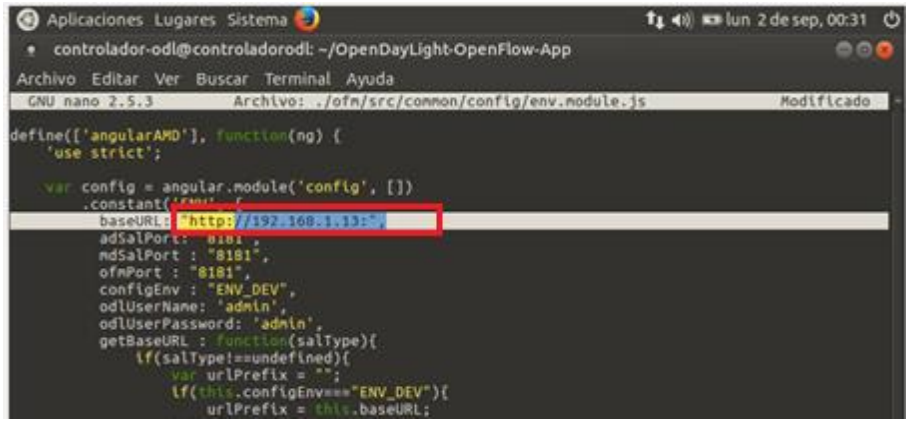

Fig. 15 **Edición del archivo [Fuente propia]**

Guardar y salir, después ejecutar la aplicación de la forma como se visualiza en la **Fig. 16**:

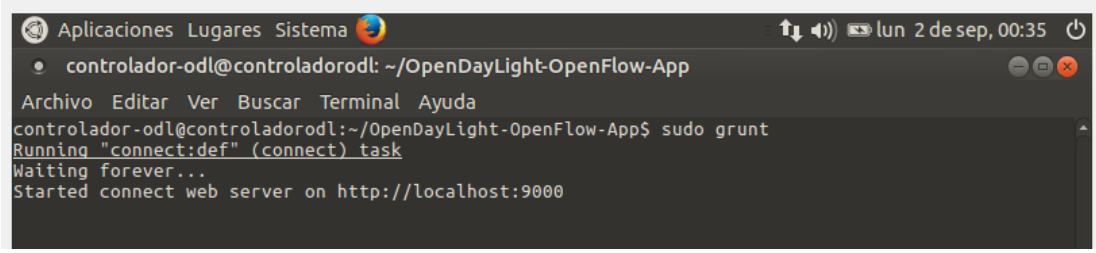

Fig. 16 **Ejecución OpenFlow Manager [Fuente propia]**

Al finalizar se muestra un enlace al que se debe ingresar para acceder a la interfaz web de OpenFlow Manager, pero antes que eso instalar Mininet para crear topologías: **sudo apt-get install Mininet**.

Y ejecutar *Mininet* con el comando: **sudo mn –controller=remote,ip=192.168.1.13**. Donde la dirección *IP* ingresada es la dirección *IP* dinámica asignada, verificar con el comando **“ifconfig”**. En la **Fig. 17** se tiene la creación de una topología para la ejecución de *Mininet*:

Fig. 17 **Ejecución de Mininet [Fuente propia]**

Ingresar con el link proporcionado anteriormente a la interfaz web de *OpenFlow* Manager para verificar la topología, aparecerá la tarjeta *Zodiac FX* sí está conectada y el controlador, como se observa en la **Fig. 18**:

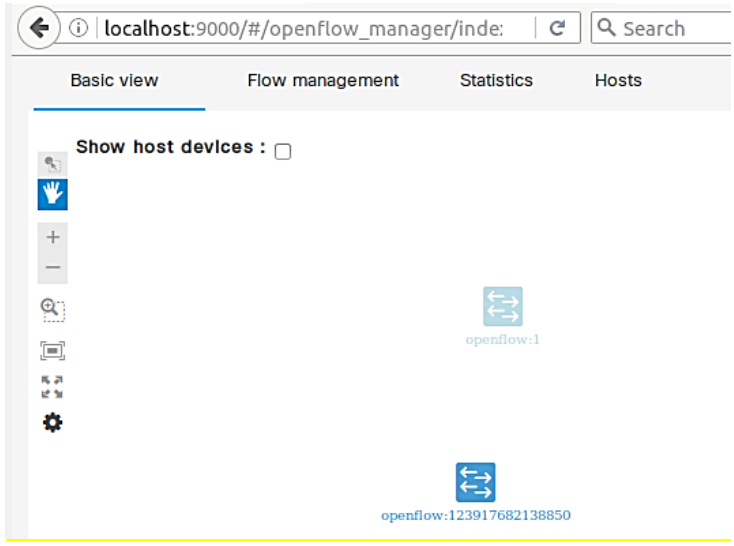

Fig. 18 **OpenFlow Manager [Fuente propia]**

Hacer ping para que se realice el envío de paquetes de un host a otro y recargar la interfaz web de *OpenFlow* Manager para observar la topología. El comando que se usa es **pingall**, el cual se demuestra en la **Fig. 19**:

```
mininet> ping all
*** Unknown command: ping all
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Fig. 19 **Ping entre hosts [Fuente propia]**

Se deben visualizar los hosts de *Mininet* en la aplicación, como se tiene en la **Fig. 20:**

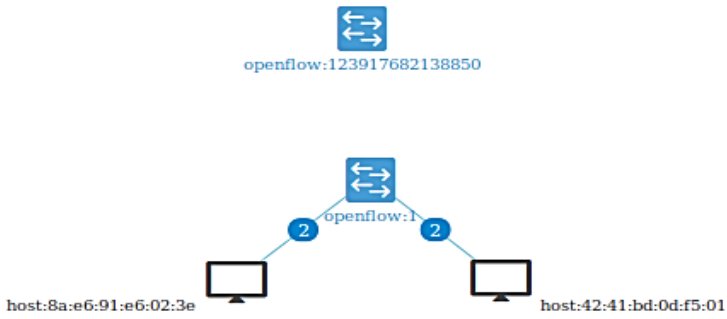

Fig. 20 **Topología de red en OpenFlow Manager [Fuente propia]**

También debe aparecer la misma topología en la interfaz web de ODL, como se señala en la **Fig. 21**, para ello recargar la página de la interfaz e ingresar al apartado de topología:

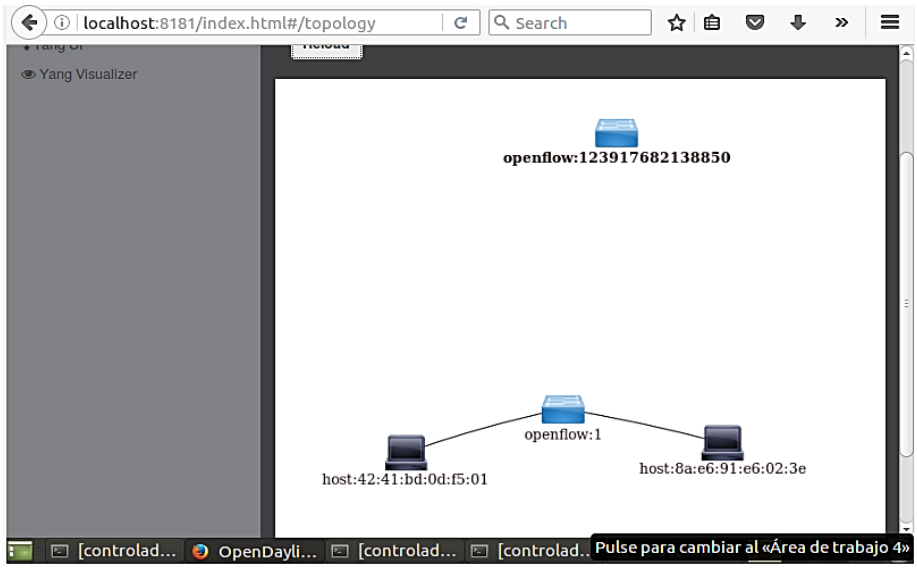

Fig. 21 **Topología de red en OpenDaylight [Fuente propia].**

Conectar una *Raspberry Pi* en uno de los puertos de la tarjeta *Zodiac FX*, debe de aparecer una topología extra, la cual es la de la *Raspberry Pi* conectada a la *Zodiac FX.* Esto se lo muestra en la **Fig. 22** y **Fig. 23***:*

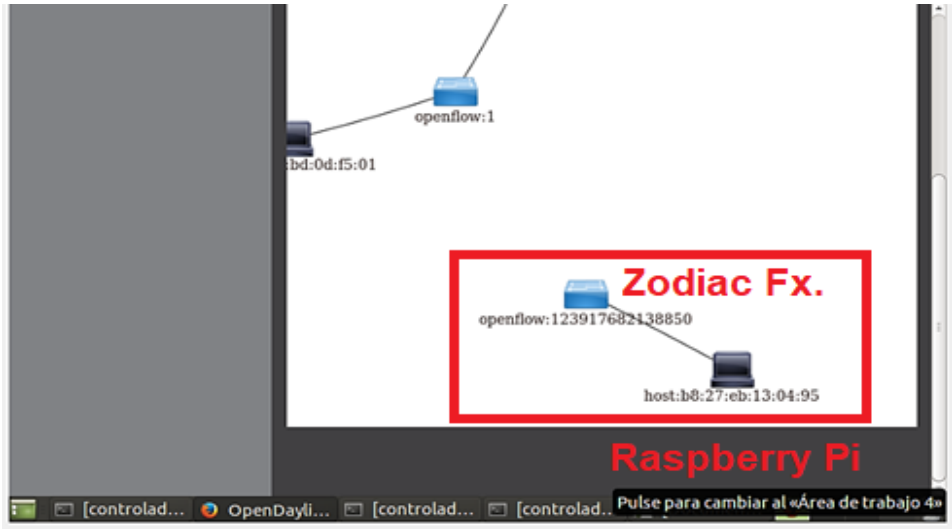

Fig. 22 **Topología de red física en interfaz web de ODL [Fuente propia]**

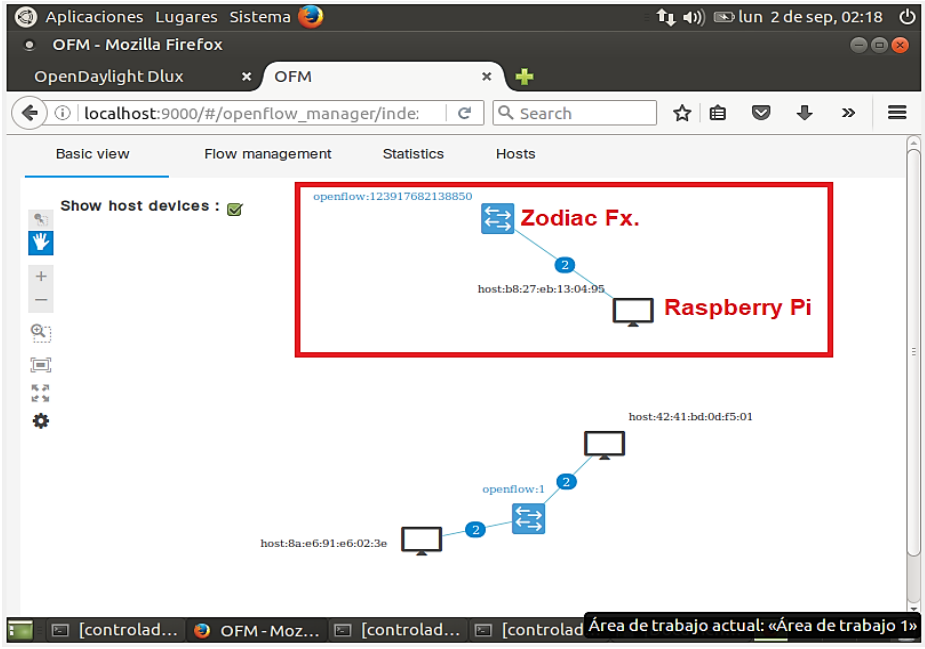

Fig. 23 **Topología de red física en OpenFlow Manager [Fuente propia]**

Luego conectarse a la tarjeta *Zodiac FX* utilizando el programa putty y restablecer la configuración de fábrica, mostrado en la **Fig. 24**:

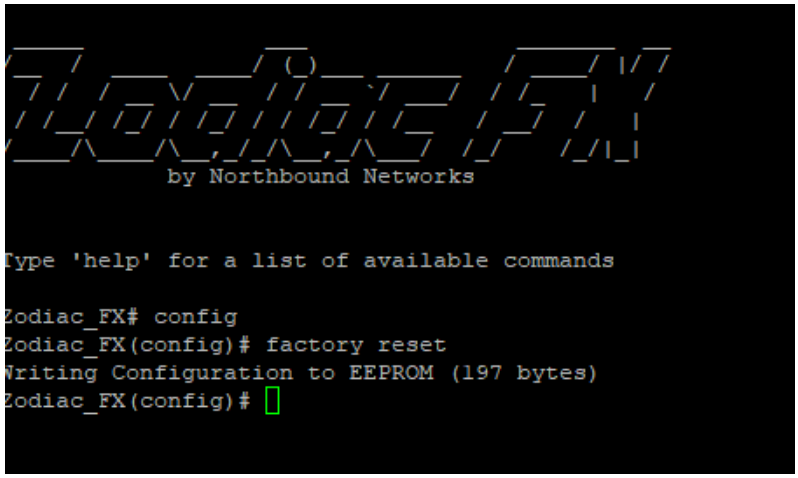

Fig. 24 **Restablecimiento de fábrica de Zodiac FX [Fuente propia]**

Conectar en el puerto 1 de la tarjeta *Zodiac FX* otra *Raspberry Pi*, desactivar la conexión *WiFi*. Luego el puerto 3 de la tarjeta *Zodiac FX,* conectarlo por cable de red a internet. Se debe asignar una dirección *IP* dentro de la red al realizar este paso, como se muestra en la **Fig. 25** y **Fig. 26:**

Fig. 25 **Dirección IP asignada [Fuente propia]**

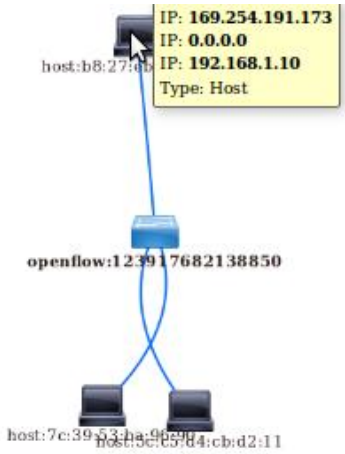

Fig. 26 **Dirección IP desde la interfaz de ODL [Fuente propia]**

Se conectó el puerto 2 de la tarjeta *Zodiac FX* por cable de red al computador que contiene la máquina virtual del controlador. Para la práctica utilizar otra de las *Raspberry Pi* implementada en el módulo y desactivar la conexión *WiFi*.

Realizar una prueba de conectividad a la *Raspberry Pi* conectada en el puerto 1 de la tarjeta *Zodiac FX*, es decir, hacer ping a la *Raspberry Pi* conectada al puerto 1, como se expone en la **Fig. 27**:

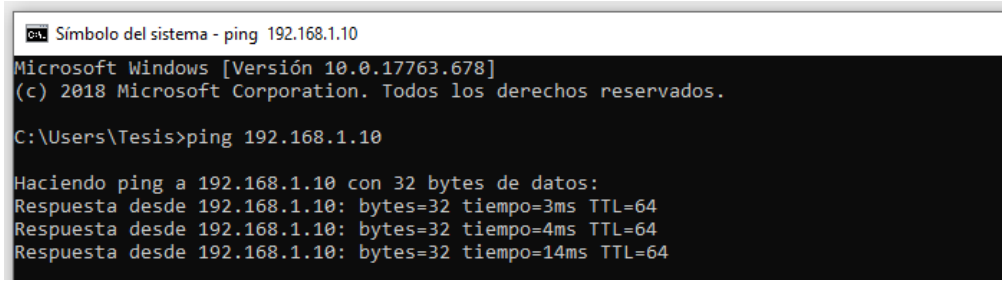

Fig. 27 **Ping entre dispositivos en la red [Fuente propia]**

Realizar una prueba de conectividad ahora desde la *Raspberry Pi* a la dirección *IP* del computador donde se encuentra ejecutándose la máquina virtual. Dicha prueba se muestra en la **Fig. 28**.

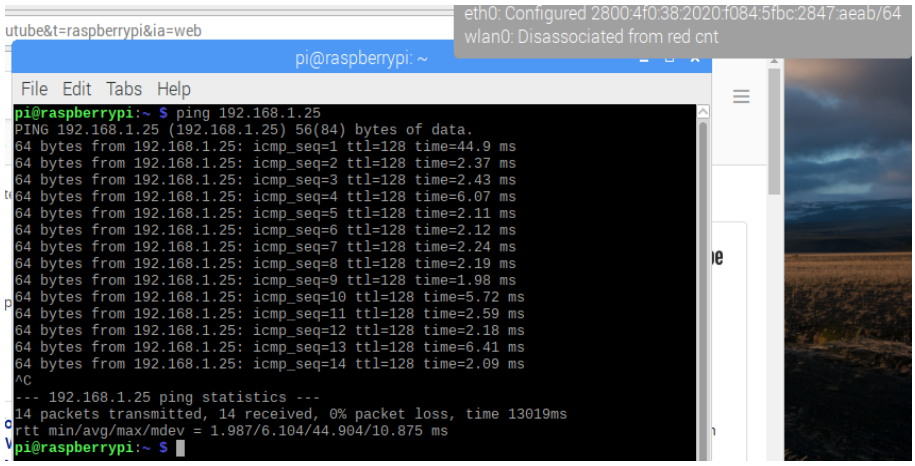

Fig. 28 **Prueba de comunicación entre dispositivos [Fuente propia]**

Configurar tabla de flujos para el controlador *OpenDaylight* desde la aplicación OpenFlow Manager en el apartado Flow Management y después seleccionar crear un nuevo flujo, el cual es el ícono señalado en la **Fig. 29**:

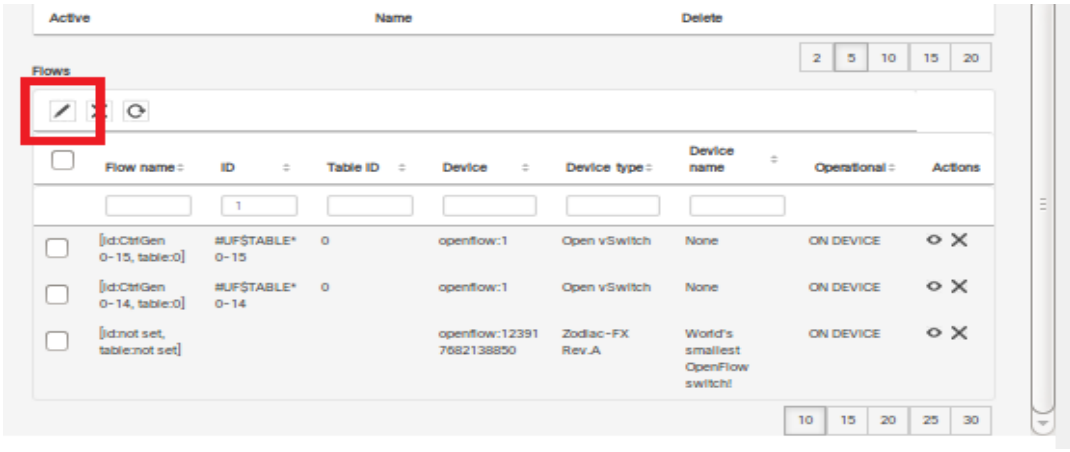

Fig. 29 **Creación de un nuevo flujo [Fuente propia].**

Configurar las opciones **TABLE**, **ID**, **PRIORITY, MATCH IN PORT**. Estos campos se muestran en la **Fig. 30** y especifican en que tabla se debe de agregar el flujo creado, su identificación, la prioridad y la opción “Match” que permite personalizar en que puerto se ejecutan estas acciones.

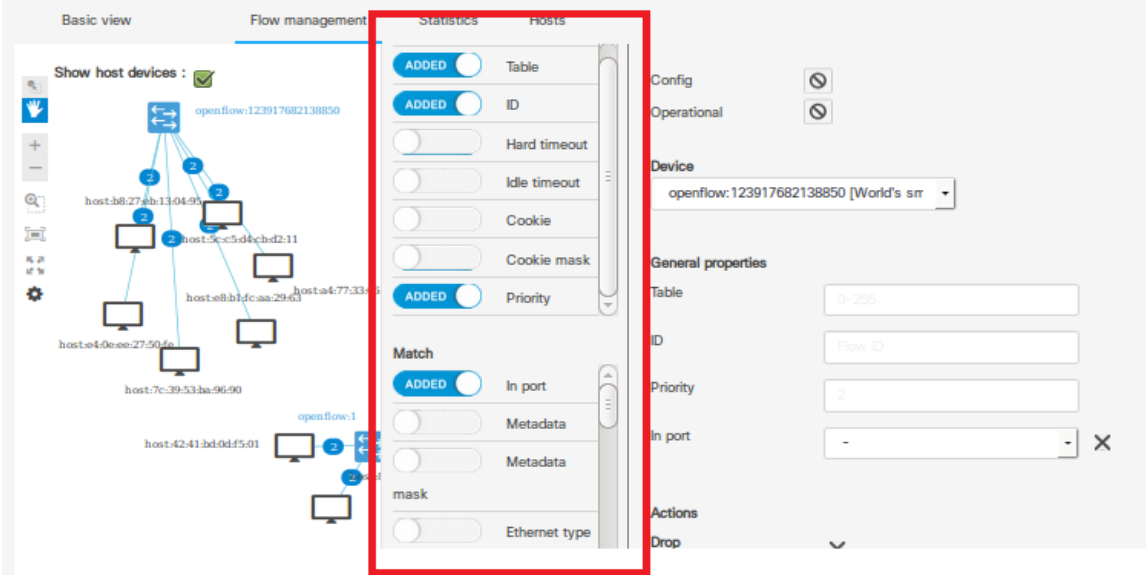

Fig. 30 **Configuración de flujos [Fuente propia]**

Seleccionar el puerto a configurar, para este caso el puerto 1 como se señala en la **Fig. 31**:

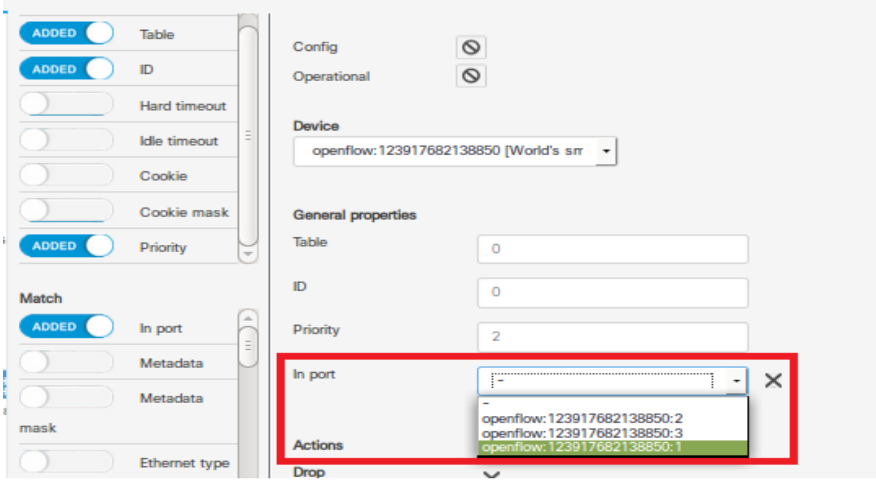

Fig. 31 **Configuración de puertos [Fuente propia]**

Seleccionar la **Acción**: **DROP**. Esta opción no permite el paso de paquetes para este puerto y después se guarda con **SEND REQUEST**. Lo mencionado se detalla en la **Fig. 32**:

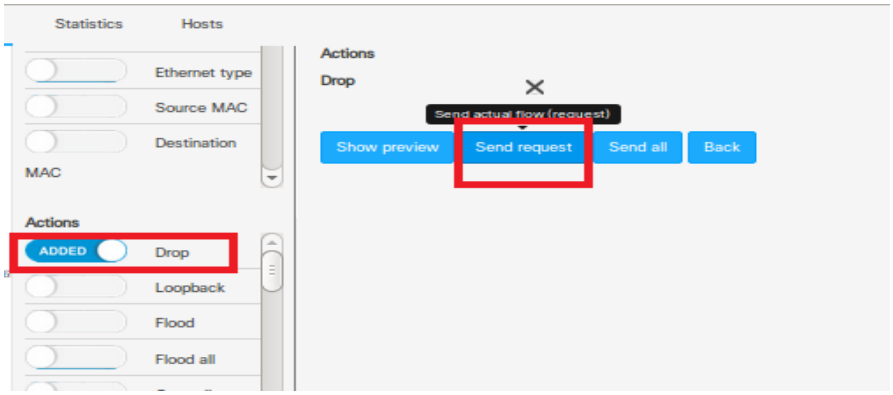

Fig. 32 **Configuración de paquetes del puerto [Fuente propia]**

En la **Fig. 33** se muestra el flujo creado en la lista del apartado “Flow management”:

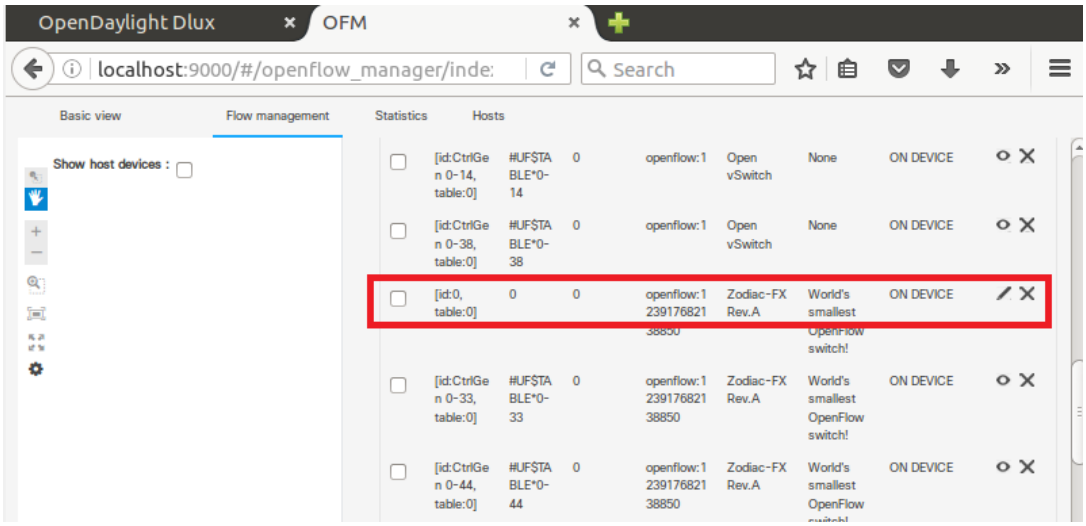

Fig. 33 **Flujo creado [Fuente propia]**

Al realizar una conexión *SSH* utilizando el programa WinSCP hacia la *Raspberry Pi* conectada al puerto 1 de la tarjeta *Zodiac FX*, se debe rechazar la conexión debido a la configuración realizada en el puerto 1 a través de la aplicación del controlador. En la **Fig. 34** se visualiza el rechazo de la solicitud para la conexión:

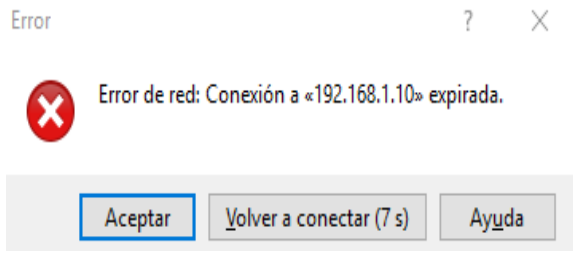

Fig. 34 **Solicitud rechazada [Fuente propia]**

Anexo G

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 7

IMPLEMENTACIÓN DE UN FIREWALL MEDIANTE EL CONTROLADOR SDN RYU.

Objetivos:

- Iniciar el controlador Ryu como un firewall.
- Comprobar la comunicación entre los hosts de Mininet.
- Crear y eliminar reglas para la comunicación entre los hosts.

Instrucciones:

- Agregar un SO (Sistema Operativo) en la *Raspberry Pi*, en este caso se usó Raspbian Stretch que se podrá hacer la debida descarga del siguiente repositorio.
- Descargar Python con todas sus librerías para su correcto funcionamiento.
- Descargar e instalar Ryu para poder ejecutarlo como un firewall.

- Descargar *xterm* para poder abrir un terminal de los distintos dispositivos de la red virtual.

Desarrollo:

Como de costumbre, el primer paso es actualizar el contenido del sistema raspbian instalado en la *Raspberry Pi* con el comando: **sudo apt-get update.** Lo siguiente es instalar Python mediante es sistema de gestión de paquetes: **sudo apt-get install python-pip**.

Instalar “git”, el cual sirve para clonar contenido de una dirección *URL* específica del cual se quiere obtener la información: **sudo apt-get install git**. Después realizar la instalación de las librerías de *Python* para su correcto funcionamiento ya que el controlador se instala y utiliza la arquitectura de la versión del *Python* instalada junto con todas sus librerías: **sudo apt-get install gcc python-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev**. Este proceso se visualiza en la **Fig. 1**:

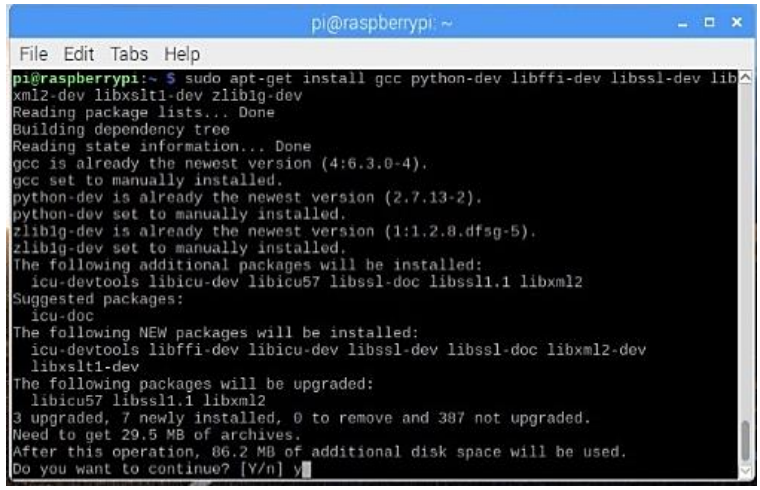

Fig. 1 Instalación de las librerías de Python [Fuente propia]

Se necesitan instalar otras librerías, lo cual se hace con el siguiente comando: **sudo apt-get install python-eventlet python-routes python-webob python-paramiko**. La instalación se muestra en la **Fig. 2**:

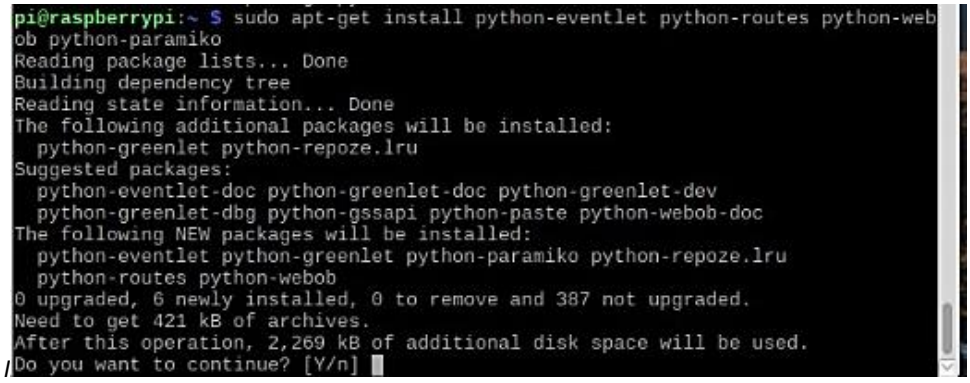

***Fig. 2* Continuación de la instalación de las librerías de Python [Fuente propia]**

Otras librerías con las siguiente: **sudo apt-get install python-pbr python-py python-six python-oslo.config python-lxml python-netaddr python-sphinx**. Esto se muestra en la **Fig. 3**:

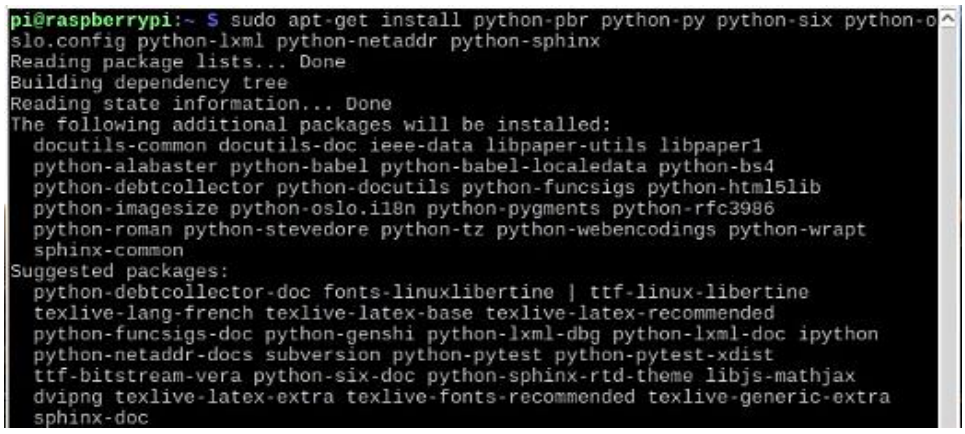

Fig. 3 **Librerías de Python [Fuente propia]**

Se necesitan muchas librerías de *Python* para su debido funcionamiento, otras librerías, mostradas en la **Fig. 4**, se instalan con el comando: **sudo apt-get install build-essential python-stdeb python-all-dev**.

Fig. 4 **Últimas librerías de Python necesarias [Fuente propia]**

Luego, hay que clonar la información de un repositorio de GitHub para poder instalar el controlador *Ryu* con git, programa instalado anteriormente. Se encuentra en GitHub debido a que el controlador es OpenSource y todos pueden acceder a este ya que es una herramienta de código abierto y no se necesitan licencias para su funcionamiento, se lo realiza con el comando: **sudo git clone git://github.com/osrg/ryu.git**. La clonación se la muestra en la **Fig. 5**:

***Fig. 5* Clonación de Ryu desde GitHub [Fuente propia]**

Ya que el controlador se ha instalado en la *Raspberry Pi*, lo siguiente es ingresar a la ruta del controlador *Ryu* mediante el comando cd. El comando cd permite cambiar la ubicación actual en la terminal y permite situarse en una carpeta o directorio en específico, en la parte inferior de la Ilustración anterior se puede observar el proceso: **cd ryu**.

Una vez que se ha cambiado la ruta de la terminal, lo siguiente es instalar otras librerías de *Python* mediante pip, en este caso se instalará “six --upgrade” que es una librería que es compatible tanto con *Python* 2 y *Python* 3, es de utilidad porque combina las diferencias entre las versiones de *Python* con la finalidad de que todo código escrito en *Python* sea compatible con ambas versiones, con --upgrade se instala la versión más reciente de la librería six: **sudo pip install six –upgrade**.

Después, dentro de la misma ruta de *Ryu* se instalará la versión más reciente librería de *Python* “eventlet”, la cual es una librería de red concurrente que permite cambiar cómo se ejecuta un código, pero no cómo se lo escribe. Como se puede observar, en la ilustración anterior se realiza la instalación de la librería con el siguiente comando: **sudo pip install eventlet –upgrade**.

Realizar la instalación de la librería de *Python* “msgpack”, este es un formato de serialización binario eficiente que permite intercambiar datos entre múltiples idiomas. Esta librería se instala mediante “271slo.config”, 271slo.config es parte de una biblioteca *OpenStack* para examinar las opciones de configuración desde la línea de comandos y los archivos de configuración: **sudo pip install 271slo.config msgpack-python**. La instalación de la librería se la señala en la **Fig. 6**:

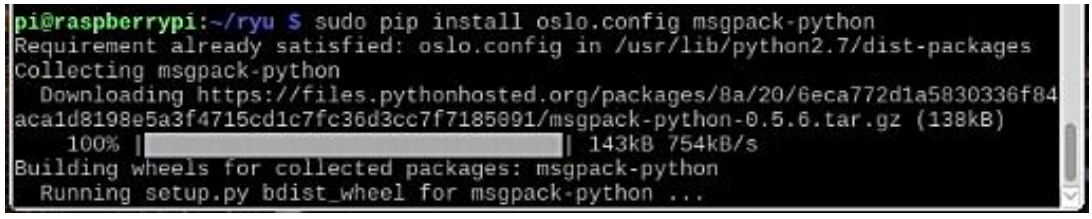

Fig. 6 **Instalación de librería mediante oslo.config [Fuente propia]**

Instalar las herramientas pip-requires, como se muestra en la **Fig. 7**. Esto sirve para solo instalar las dependencias que se mencionan en el archivo pip-requires con el siguiente comando: **sudo pip install -r tools/pip-requires**.

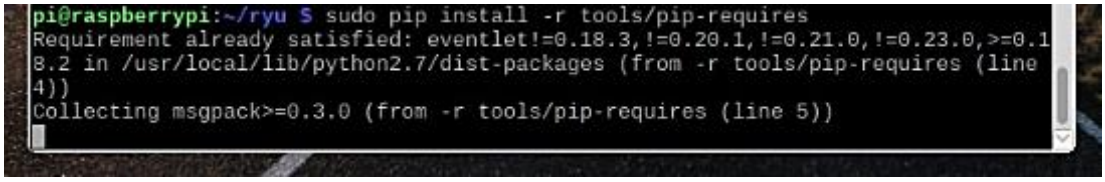

Fig. 7 **Instalación de herramientas de pip-requires [Fuente propia]**

Después de realizar la instalación de pip-requires, se debe de instalar **./setup.py**. Setup.py es un archivo de *Python*, que generalmente indica que el módulo o el paquete que está a punto de instalarse ha sido empaquetado y distribuido con Distutils, que es el estándar para distribuir módulos de *Python*, esto le permite instalar fácilmente paquetes de *Python*. La instalación se la realiza en la misma ruta del controlador *Ryu* con el siguiente comando: **sudo python ./setup.py install**.

Desde otro terminal, como se hará uso de *Mininet* para la virtualización de una red con todos sus elementos, se lo instala con lo siguiente: **sudo apt-get install Mininet**.

Instalaremos “xterm” para poder abrir terminales en cada elemento de la red que se virtualizan en *Mininet:* **sudo apt-get install xterm**.

Con el siguiente comando se crea de una topología simple de 3 hosts mediante *Mininet:* **sudo mn --topo single,4 --mac --switch ovsk --controller remote.** Al instante se abren los terminales de cada uno de los dispositivos de la red virtual creada con *Mininet*, es decir, un terminal para el controlador, para el *switch* y terminales para cada host, por el momento como en el comando se especificó con 4 hosts.

Como primer paso se deben configurar las *VLAN ID* de cada uno de los hosts, mostrado en la **Fig. 8**. Se configura lo siguiente en el host 1:

- **ip addr del 10.0.0.1/8 dev h1-eth0**
- **ip link add link h1-eth0 name h1-eth0.2 type vlan id 2**
- **ip addr add 10.0.0.1/8 dev h1-eth0.2**
- **ip link set dev h1-eth0.2 up**

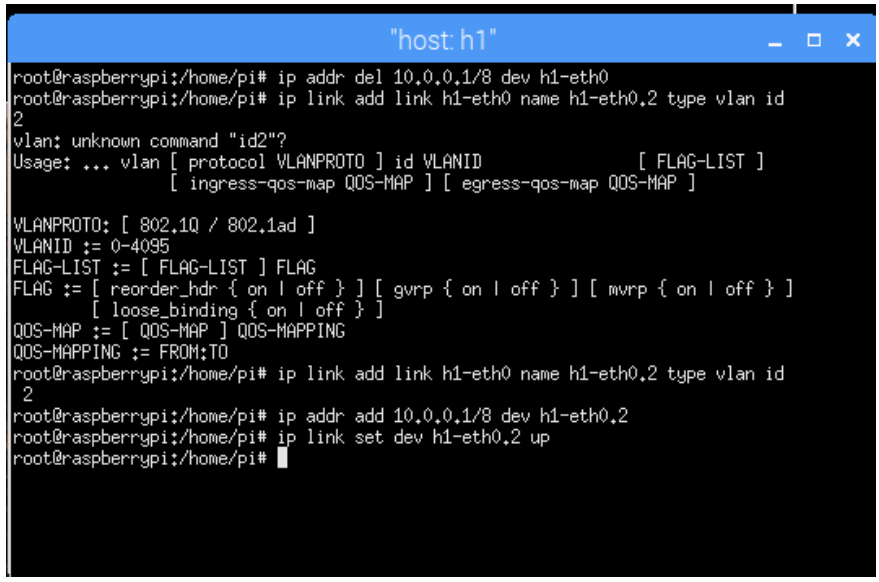

Fig. 8 Configuración de VLAN ID en host # 1 [Fuente propia]

En el host 2:

- **ip addr del 10.0.0.2/8 dev h2-eth0**

- **ip link add link h2-eth0 name h2-eth0.2 type vlan id 2**
- **ip addr add 10.0.0.2/8 dev h2-eth0.2**
- **ip link set dev h2-eth0.2 up**

En el host 3:

- **ip addr del 10.0.0.3/8 dev h3-eth0**
- **ip link add link h3-eth0 name h3-eth0.110 type vlan id 110**
- **ip addr add 10.0.0.3/8 dev h3-eth0.110**
- **ip link set dev h3-eth0.110 up**

En el host 4:

- **ip addr del 10.0.0.4/8 dev h4-eth0**
- **ip link add link h4-eth0 name h4-eth0.110 type vlan id 110**
- **ip addr add 10.0.0.4/8 dev h4-eth0.110**

- **ip link set dev h4-eth0.110 up**

En el terminal del *switch* especificaremos la versión del protocolo *OpenFlow* con la que se trabaja en cada host, como se muestra en la **Fig. 9**. El comando a usar es el siguiente: **ovs-vsctl set Bridge s1 protocols=OpenFlow13**.

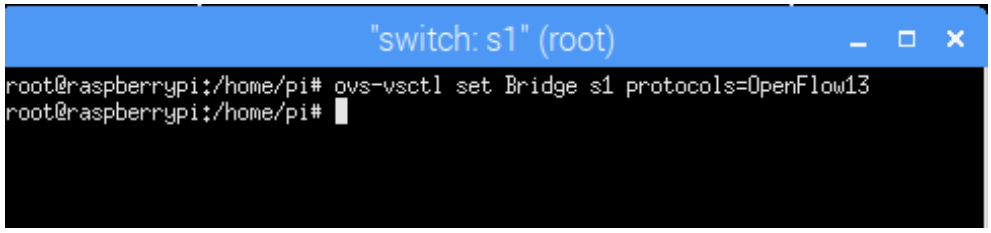

Fig. 9 **Versión de OpenFlow especificada [Fuente propia]**

Iniciamos *Ryu* como un firewall en el terminal del controlador (C0) con el siguiente comando: **ryu-manager ryu.app.rest_firewall**. En la **Fig. 10** se muestra lo mencionado anteriormente:

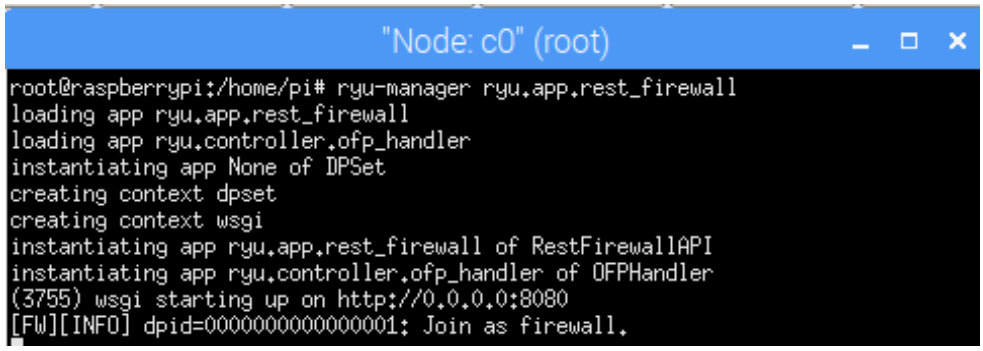

Fig. 10 **Inicio de Ryu como Firewall [Fuente propia]**

Aparece este mensaje en el terminal como muestra de que se inició exitosamente la aplicación, como se presenta en la **Fig. 11**:

Fig. 11 **Inicio de la aplicación exitoso [Fuente propia]**

Donde el “dpid” es la dirección MAC del switch virtualizado, hay que tener en cuenta esta dirección MAC ya que se la utilizará en diversas ocasiones. Inmediatamente después de iniciar el firewall, en el terminal c0 (controlador), se debe de cambiar el estado de este a habilitado ya que por defecto viene deshabilitado para cortar toda comunicación, ya que ya hay un terminal C0 corriendo el controlador, se procede a abrir otro en el terminal donde se está ejecutando Mininet con el comando: **xterm c0**.

En el terminal recién abierto se ejecuta el comando que sirve para cambiar el estado del firewall, el comando es el siguiente: **curl -X PUT http://localhost:8080/firewall/module/enable/0000000000000001**. El uso del comando se lo muestra en la **Fig. 12**:

Fig. 12 **Cambio de estado del firewall [Fuente propia]**

Donde el “**0000000000000001**” es la dirección MAC del switch. Para confirmar el estado se usa el comando: **curl http://localhost:8080/firewall/module/status**. Como se observa en la **Fig. 13**:

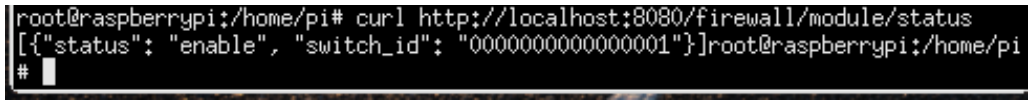

Fig. 13 **Confirmación del estado del firewall [Fuente propia]**

Seguido de esto, se expone mediante un ping la comunicación entre h1 y h2, como aún no hay reglas escritas y guardadas, la comunicación está bloqueada. Un mensaje, en el terminal donde se está corriendo el firewall, debe de confirmar bloqueo de comunicación, como se indica en la **Fig. 14**:

Fig. 14 **Comunicación bloqueada [Fuente propia]**

Para que haya comunicación entre los dos hosts, se debe de crear una regla, como se muestra en la **Fig. 15**. Primero se agrega la regla para la comunicación ente h1 a h2. Como en el host 1 y en el host 2 su VLAN ID es 2, se escribe la regla de la siguiente

forma: **curl -X POST -d '{"nw_src": "10.0.0.0/8", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/2**.

Fig. 15 **Regla para la comunicación de h1 a h2 [Fuente propia]**

Después se escribe otra regla cambiando el "nw_src" a "nw_dst": **curl -X POST -d '{"nw_dst": "10.0.0.0/8", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/2**.

Se puede visualizar el uso del comando en la **Fig. 16**:

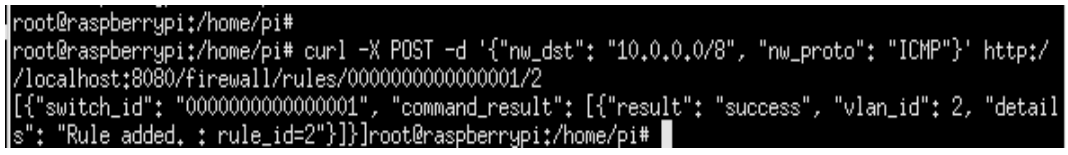

Fig. 16 **Comunicación entre h1 y h2 [Fuente propia]**

Para corroborar que las reglas se hayan guardado, en el terminal donde corre el firewall debe de aparecer un mensaje de que la regla ha sido añadida correctamente, como se puede observar en la **Fig. 17**:

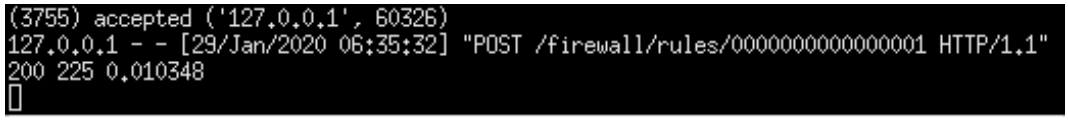

Fig. 17 **Reglas agregadas correctamente [Fuente propia]**

Si se desea observar toda la información sobre las reglas creadas, en el terminal C0 se debe de ingresar el siguiente comando: **curl http://localhost:8080/firewall/rules/0000000000000001/all.** En la **Fig. 18** se exhiben las reglas creadas:

Fig. 18 **Información de las reglas creadas [Fuente propia]**

Para confirmar que la regla esté correctamente configurada, se realiza un ping entre ambos hosts para detallar el funcionamiento de dicha regla. En la **Fig. 19** se denota la comunicación entre el host 1 y el host 2:

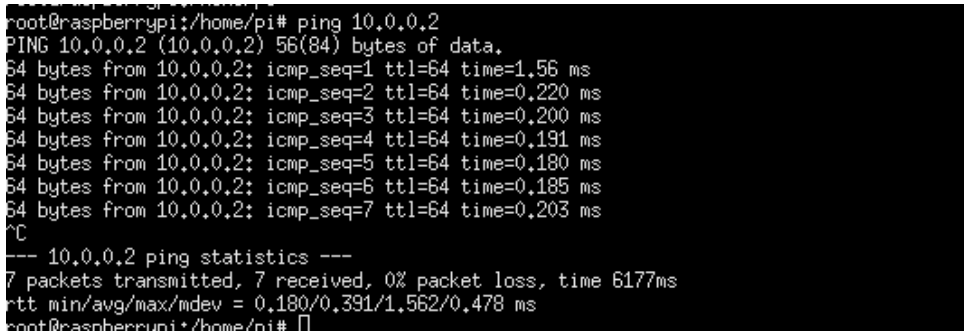

Fig. 19 **Comunicación desde h1 a h2 [Fuente propia]**

Como no hay alguna regla que permita la comunicación entre h3 a h4 o viceversa, para corroborar esto se debe de hacer ping de h3 a h4, como se observa en la **Fig. 20**, debe de aparecer el siguiente mensaje en el terminal que está ejecutando Ryu como firewall:

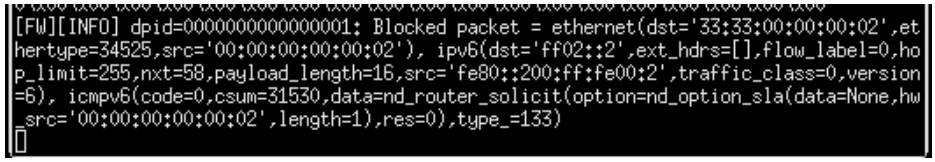

Fig. 20 **Bloqueo en la comunicación entre h3 y h4 [Fuente propia]**

Anexo H

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 8

IMPLEMENTACIÓN DE UN SWITCH SIMPLE MEDIANTE EL CONTROLADOR SDN RYU.

Objetivos:

- Ejecutar la aplicación Simple Switch de Ryu para agregar los flujos a la *Zodiac FX*.
- Comprobar la comunicación entre los hosts conectados a la *Zodiac FX*.
- Verificar la topología formada con la aplicación Flow manager.

Instrucciones:

- Agregar un SO (Sistema Operativo) en la *Raspberry Pi 3 Model B*, en este caso se usó Raspbian Stretch que se podrá hacer la debida descarga en el repositorio acordado.

- Instalar Apache para no tener inconvenientes al usar la aplicación Flow Manager.
- Descargar Python con todas sus librerías para su correcto funcionamiento.

Desarrollo:

Como primer paso hay que actualizar el contenido del sistema de la *Raspberry Pi 3 Model B* con el comando, para ello se deberá abril la terminal e ingresar la siguiente línea de comando: **sudo apt-get update**.

Después proceder a realizar la debida instalación de Apache en su segunda versión para poder visualizar el contenido de Flow manager ingresando el comando: **sudo apt-get install apache2**.

Para esta práctica también se requiere de la instalación de todos los paquetes de Python de igual forma que en la práctica # 7 (**Anexo G**), para ello se debe de usar los siguientes comandos:

- **sudo apt-get install python-pip**
- **sudo apt-get install git**
- **sudo apt-get install gcc python-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev**

- **sudo apt-get install python-eventlet python-routes python-webob python-paramiko**

- **sudo apt-get install python-pbr python-py python-six python-oslo.config python-lxml python-netaddr python-sphinx**

- **sudo apt-get install build-essential python-stdeb python-all-dev**

Una vez se haya instalado las librerías de Python, se deberá de clonar Ryu desde su repositorio en GitHub con el siguiente comando: **sudo git clone git://github.com/osrg/ryu.git**

Se debe de seguir instalando paquetes de Python para el debido funcionamiento de Ryu, estos paquetes se los instala dentro de la carpeta donde se encuentre Ryu de la siguiente forma: **cd ryu**.

De igual forma que en la práctica # 7 (**Anexo G**), se deben de instalar los siguientes paquetes con sus respectivos comandos:

- **sudo pip install six –upgrade**

- **sudo pip install eventlet –upgrade**

- **sudo pip install oslo.config msgpack-python**

- **sudo pip install -r tools/pip-requires**

- **sudo python ./setup.py install**

Una vez instaladas todas las librerías de *Python*, se procede a clonar la información de otro repositorio de GitHub para poder instalar una aplicación del controlador Ryu, la cual es Flow manager.

Flow manager es una aplicación de código abierto desarrollada para ser compatible con el controlador *Ryu,* la cual permite el control manual sobre las tablas de flujo en una red que haya sido implementada bajo *OpenFlow*. Se pueden crear, modificar o eliminar flujos directamente desde la aplicación de una manera más sencilla y gráfica. También permite monitorear los *switches OpenFlow* y ver sus estadísticas. Es ideal para aprender cómo funciona *OpenFlow* en un entorno de laboratorio, o en conjunto con otras aplicaciones para ajustar el comportamiento de los flujos de red en un entorno de producción. El comando a usar es el siguiente: **sudo git clone https://github.com/martimy/flowmanager**. En la **Fig. 1** se visualiza el proceso de clonación:

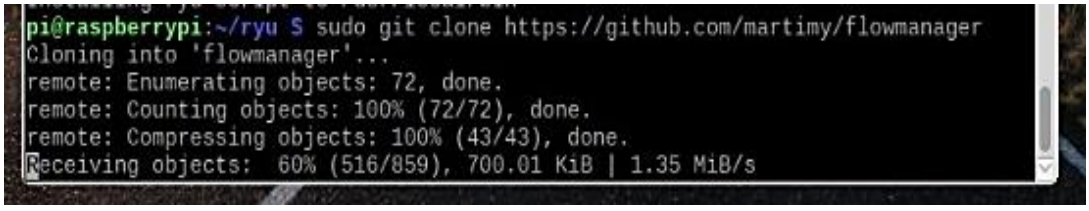

Fig. 1 **Clonación de Flow manager desde GitHub [Fuente propia]**

Para poder tener acceso a una red *WiFi* en la *Raspberry Pi* que actúa como controlador y realizar la conexión por Ethernet (cable de red o RJ45) al puerto 4 de la tarjeta *Zodiac FX*, se deben seguir los siguientes pasos. En primer lugar, es necesario tener Raspbian en la *Raspberry Pi* y además tener activada la conexión *SSH*. Para editar el archivo de configuración de las interfaces de red se utiliza el siguiente comando: **sudo nano /etc/network/interfaces**.

Una vez que se haya ingresado al archivo de configuración, lo siguiente es ingresar los parámetros de la red Ethernet y de la WLAN, primero listaremos la configuración para la red Ethernet:

- auto eth0
- iface eth0 inet static
- address 10.0.0.2 (puede ser la dirección IP que se desee)
- netmask 255.255.255.0

Para la configuración de la WLAN:

- allow-hotplug wlan0
- iface wlan0 inet dhcp

- wpa-ssid "Nombre_Red_WiFi"

- wpa-psk "Contraseña_WiFi"

Donde “Nombre_Red_WiFi” es la red a la que se desea acceder y “Contraseña_WiFi” es la contraseña de dicha red. Guardar las modificaciones y salir, en la **Fig. 2** se tiene la configuración de las interfaces. Reiniciar la *Raspberry Pi 3 Model B* para que la configuración se establezca.

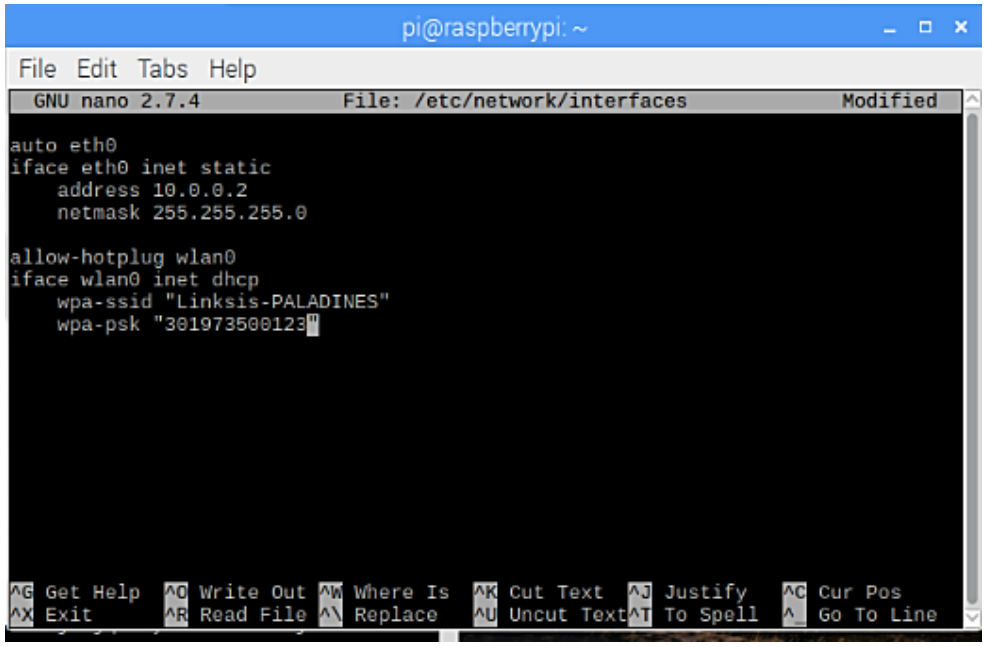

***Fig. 2* Configuración de interfaces [Fuente propia]**

Configurar la tarjeta *Zodiac FX* ingresando la dirección *IP* Ethernet ingresada anteriormente, esta dirección *IP* se la debe de ingresar como la *IP* del controlador

OpenFlow en la configuración de la tarjeta *Zodiac FX*, para ello descargar la aplicación "putty" mediante la siguiente línea de comando: **sudo apt-get install putty**.

Ejecutar el programa putty con el comando: **sudo putty**.

En la **Fig. 3** se muestra la ventana del programa putty:

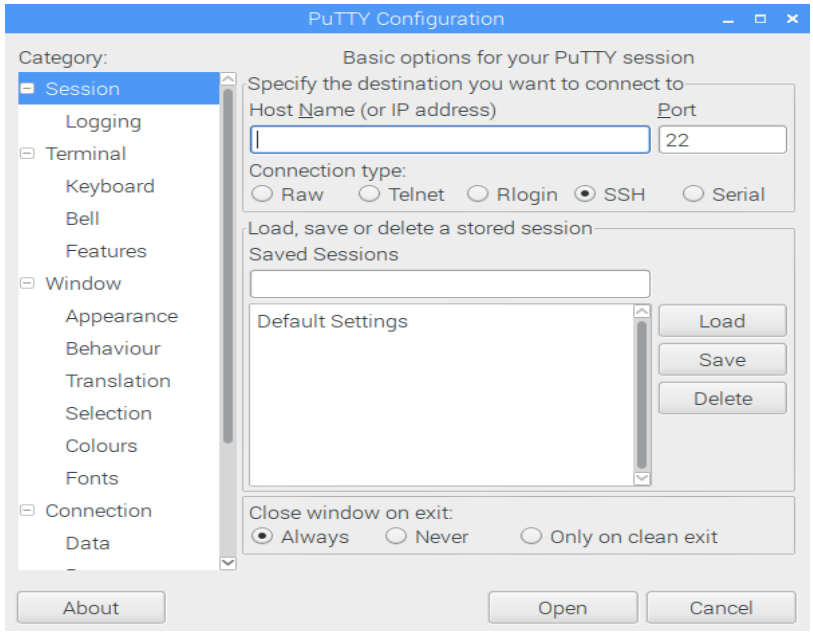

***Fig. 3* Putty [Fuente propia]**

Una vez ejecutado putty proceder a seleccionar la opción "Serial" en lugar de "*SSH*" (opción establecida por defecto) y seleccionar el puerto al que se hace la conexión con la tarjeta *Zodiac FX* (en estos casos, el puerto serial al que se conecta la tarjeta *Zodiac FX* es el puerto **ACM0**).

Cuando se activa el tipo de conexión serial hay que cambiar el puerto, se visualiza por defecto “ttyCOM0” u otros, se debe cambiar por “ttyACM0” para poder ingresar a la interfaz de línea de comandos de la tarjeta *Zodiac FX,* la cual se presenta en la **Fig. 4**, dar enter o clic en Open y se procede a configurar, al principio la ventana no muestra ningún tipo de información, presionar la tecla enter para que muestre información:

Fig. 4 **Ventana de terminal de la Zodiac FX [Fuente propia]**

Escribir el comando “help” para observar la lista de comandos que funcionan en la tarjeta *Zodiac FX* para su configuración. Con respecto a la configuración, se muestra la siguiente **Fig. 5**:

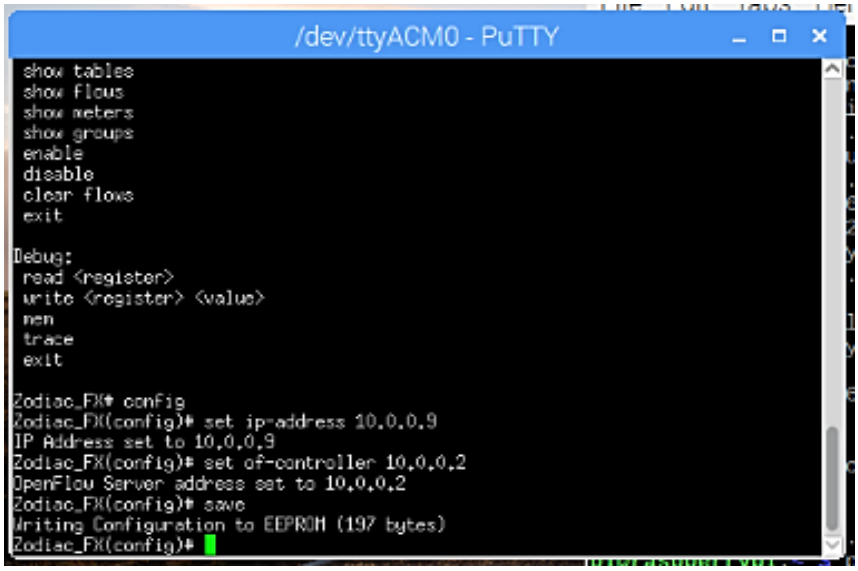

***Fig. 5* Configuración de la Zodiac FX [Fuente propia]**

En la interfaz de línea de comando se puede configurar una dirección *IP* la tarjeta *Zodiac FX*, la máscara de red, la dirección *IP* de la puerta de enlace (Gateway), dirección *IP* del controlador *OpenFlow*, puerto *OpenFlow*, etc.

En éste caso, se establecen todos los valores predeterminados, excepto la dirección *IP* del controlador *OpenFlow* y el puerto OpenFlow. Ingresar al modo de configuración "**config**", y establecer la dirección *IP* del controlador *OpenFlow* se utiliza: **set of-controller (dirección IP del controlador OpenFlow)**.

Establecer el puerto en el cual *OpenFlow* funciona predeterminadamente, el cual es el puerto *TCP* 6633: **set of-port 6633**.

Guardar la configuración con el comando **save** y cerrar la interfaz de línea de comando de la tarjeta *Zodiac FX*. Una vez que se haya configurado todo lo necesario

para el correcto funcionamiento de la tarjeta *Zodiac FX*, ejecutar la aplicación Flow manager junto con la aplicación simple_switch_13, la cual es una aplicación predeterminada de *Ryu* que permite que la tarjeta *Zodiac FX* funcione como un *switch* para que en esta tarjeta procese los paquetes *OpenFlow*, cabe recalcar que para ejecutar estas aplicaciones, primero deben de estar conectados los dispositivos que funcionan como hosts, caso contrario, no se reconocen los dispositivos; ingresar el siguiente comando: **ryu-manager --verbose ~/flowmanager/flowmanager.py ryu.app.simple_switch_13**. En la **Fig. 6** se señala la ejecución de Flow Manager:

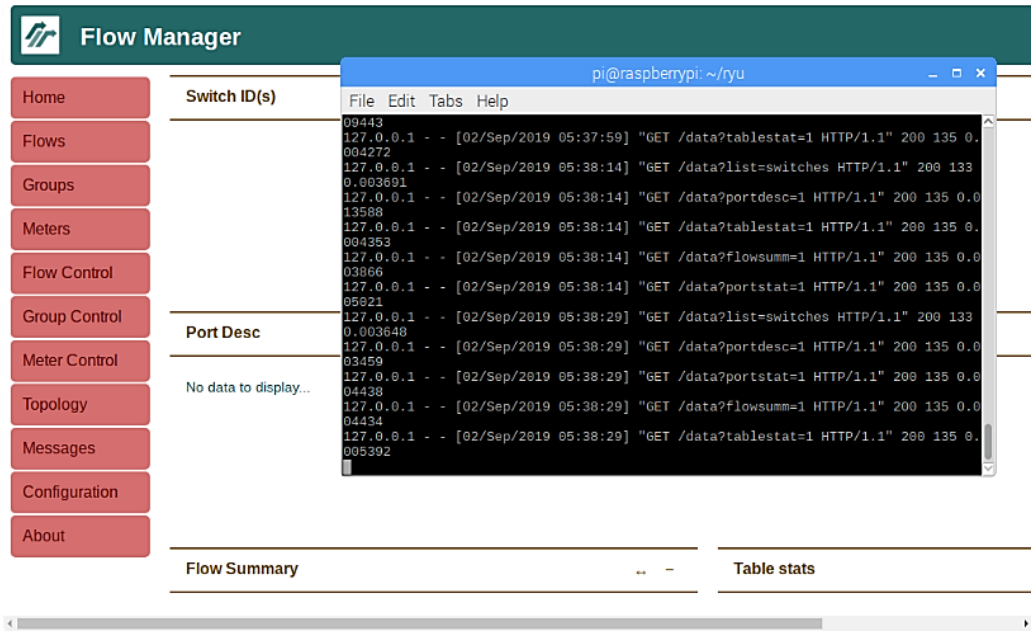

Fig. 6 **Ejecución de Flow manager [Fuente propia]**

Esperar unos segundos para que se establezcan los flows (flujos) dentro de la aplicación Flow manager y proceder a ingresar a ella. La forma de ingresar a la interfaz web de Flow manager es por medio de un navegador de internet con la siguiente

dirección *URL*, se sugiere utilizar Google Chrome: **http://localhost:8080/home/index.html**.

Una vez de dentro de la interfaz web, se pueden observar algunas pestañas de opciones de configuración, para este caso puntual se utilizarán las pestañas de home, flows, topology y messages.

En el desarrollo se utilizaron dos laptops para la debida conexión (para el desarrollo de la práctica se deben utilizar las *Raspberry Pi* implementadas en el módulo), ambas direcciones *MAC* deberían de aparecer, indicando con esto que sí están conectadas a la tarjeta *Zodiac FX*. En la **Fig. 7** se tiene la página de inicio de Flow Manager:

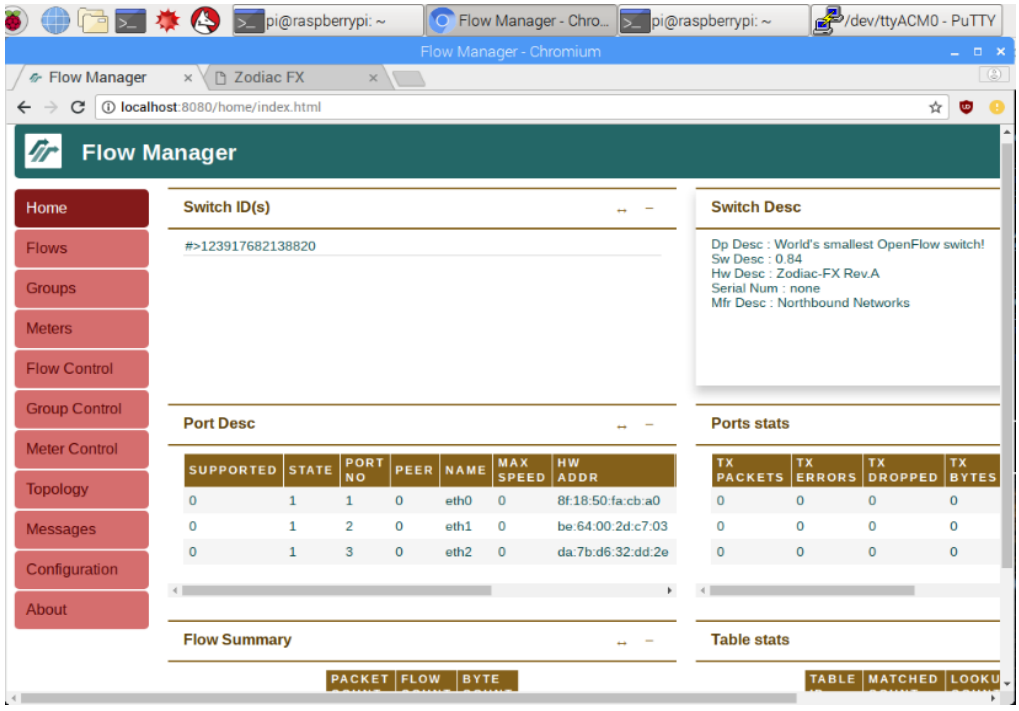

***Fig. 7* Página de inicio de Flow manager [Fuente propia]**

Para observar la topología creada mediante la aplicación “simple_switch_13” se debe de ingresar a la pestaña de “topology” y comprobar lo siguiente que se observa en la **Fig. 8**:

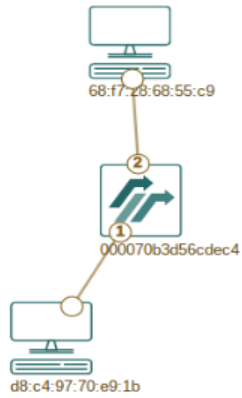

Fig. 8 **Topología de red [Fuente propia]**

Se pueden observar que fueron dos dispositivos los que se conectaron a la tarjeta *Zodiac FX*, también se observa que ambas direcciones *MAC* y el puerto en el que fueron conectados todos los dispositivos. Para tener más información con respecto a la conexión de los dispositivos, junto a la opción “graph” está la opción “tables”, se selecciona la opción “tables” y se tendrán los respectivos datos de los dispositivos, como se expone en la **Fig. 9**:

Fig. 9 **Tablas de información de dispositivos [Fuente propia]**

En este apartado se puede observar el puerto, dirección *MAC*, dirección *IPv4* y también la dirección *IPv6* de cada dispositivo que están conectándose a la tarjeta. Si se desean ver los paquetes recibidos y enviados en cada puerto de la tarjeta *Zodiac FX*, se pueden revisar en la interfaz web de la tarjeta *Zodiac FX*. Se accede escribiendo en la *URL* la dirección *IP* establecida de la *Zodiac FX.* En la **Fig. 10** se demuestra el apartado de los “Port Information”:

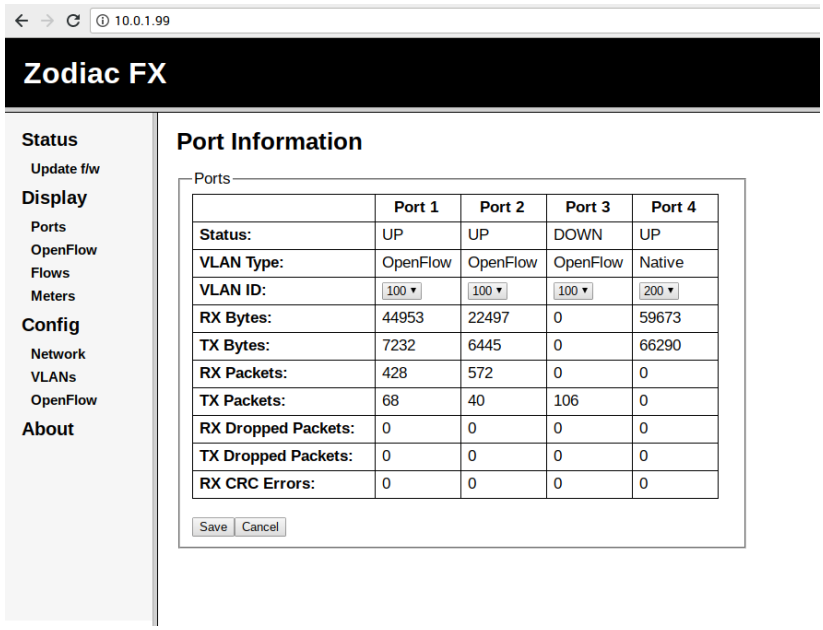

Fig. 10 **Información de los puertos de la Zodiac FX [Fuente propia]**

Revisar los flujos o flows en el apartado de “Flows”, como se indica en la **Fig. 11 y Fig. 12**:

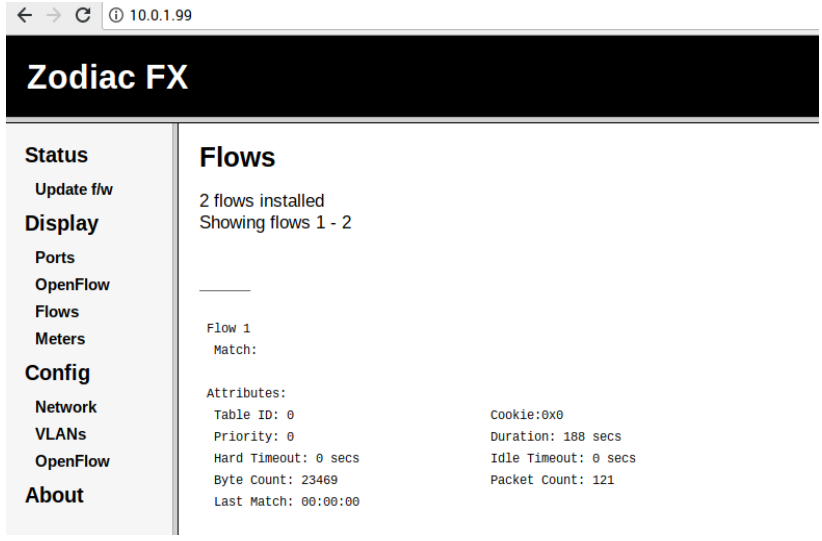

Fig. 11 **Flujo 1 instalado [Fuente propia]**

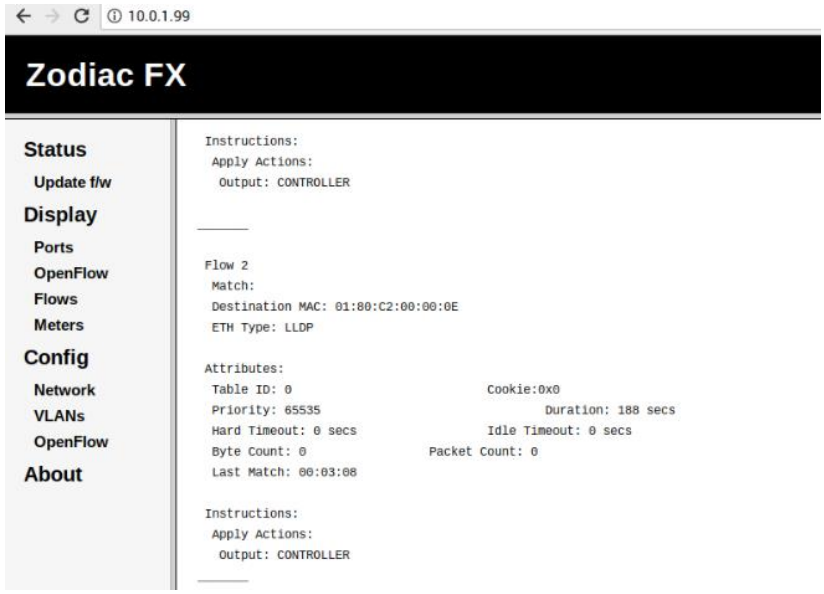

Fig. 12 **Flujo 2 instalado [Fuente propia]**

Dentro de la interfaz de Flow manager, se pueden ver los mensajes enviados por cada paquete. Esto se lo demuestra en la **Fig. 13**:

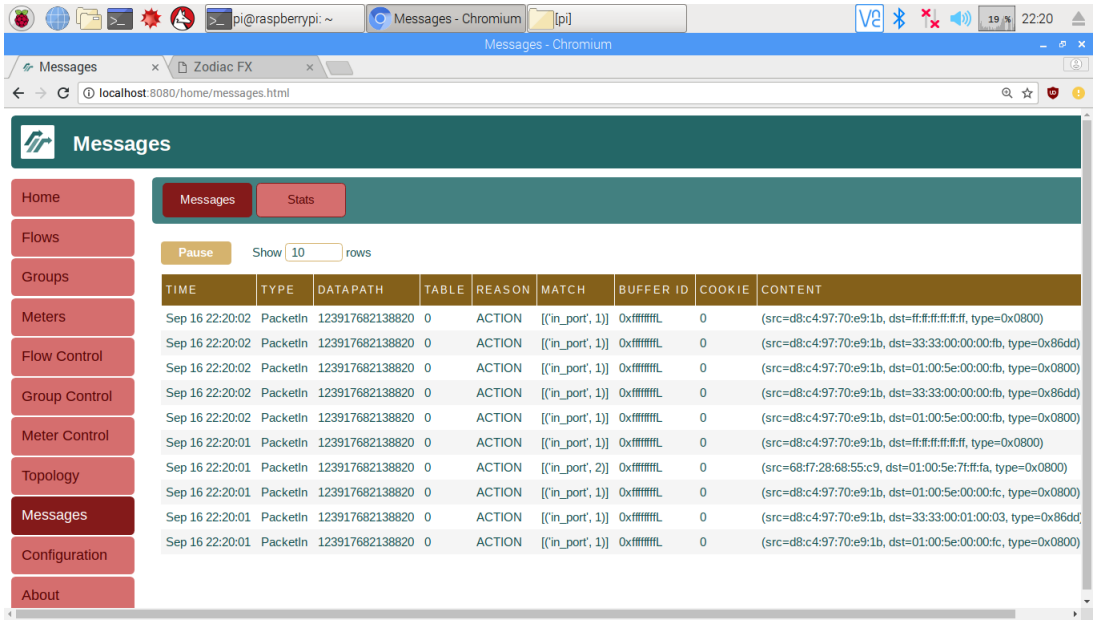

Fig. 13 **Detalles de mensajes de los puertos [Fuente propia]**

Para verificar si los hosts tienen comunicación, realizar ping de uno a otro, tal cual se muestra en la **Fig. 14**:

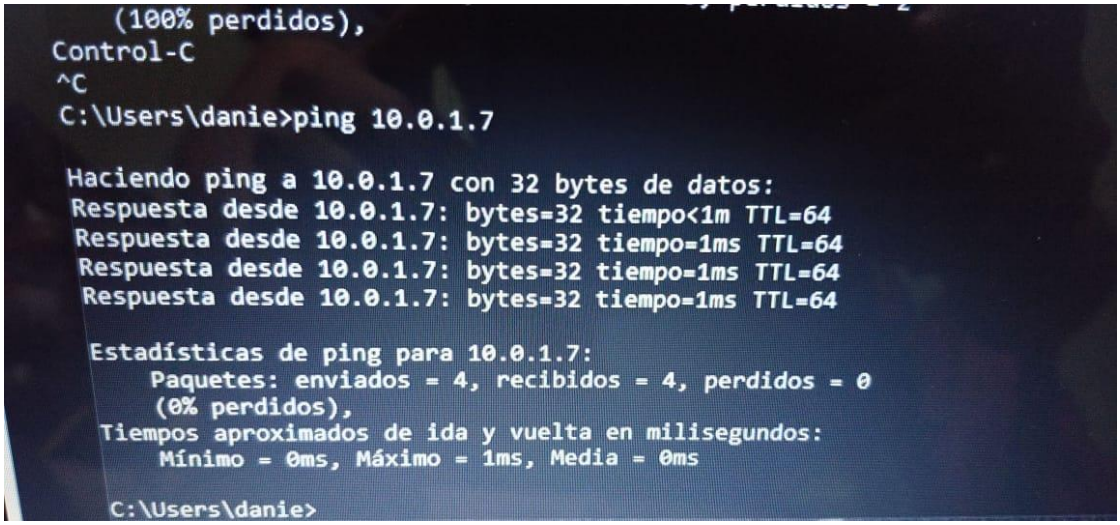

Fig. 14 **Comunicación entre hosts exitosa [Fuente propia]**

Anexo I

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 9

IMPLEMENTACIÓN DE UN IDS MEDIANTE EL CONTROLADOR SDN RYU.

Objetivos:

- Establecer la integración entre un controlador *SDN* y un *IDS*.
- Crear reglas que permitan el monitoreo del sistema
- Visualizar alertas configuradas en el controlador *SDN*.

Instrucciones:

- Crear una nueva interfaz de red Ethernet para el controlador *Ryu*.
- Instalar y configurar *Snort*.
- Establecer los parámetros correctos para el funcionamiento de Pigrelay.

Desarrollo:

Después de realizar la configuración y ejecución de controlador *Ryu* en la práctica # 7 y práctica # 8, se realizará nuevamente la configuración de las interfaces de red de la *Raspberry Pi* que funciona como controlador *SDN* para que se pueda establecer la comunicación entre el controlador mencionado y los hosts conectados a la tarjeta *Zodiac FX* con el comando: **sudo nano /etc/network/interfaces**. En la **Fig. 1** se muestra la creación de una nueva interfaz de red:

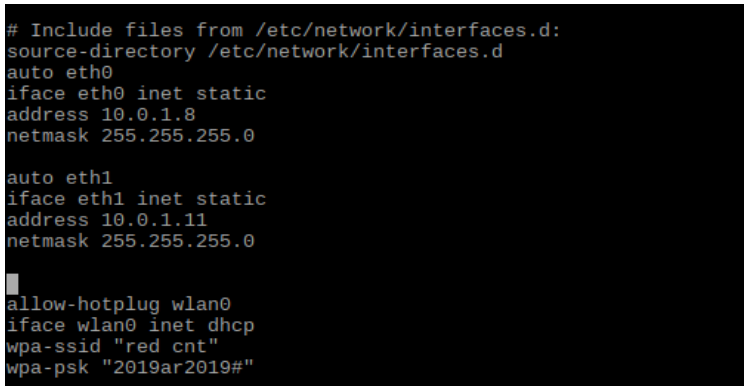

Fig. 1 **Creación de una nueva interfaz de red [Fuente propia]**

La interfaz de red **eth0** es la dirección IP del controlador y la interfaz **eth1** es aquella que se utiliza para establecer comunicación entre el controlador y el *IDS Snort*. Se especifica que se debe conectar el adaptador de red *USB* a ethernet desde la *Raspberry Pi* que actúa como controlador al puerto #3 de la tarjeta *Zodiac FX*, luego conectar el cable de red que va desde el puerto ethernet de la Raspberry Pi al puerto #4 de la tarjeta *Zodiac FX*, como se indica en la **Fig. 2**:

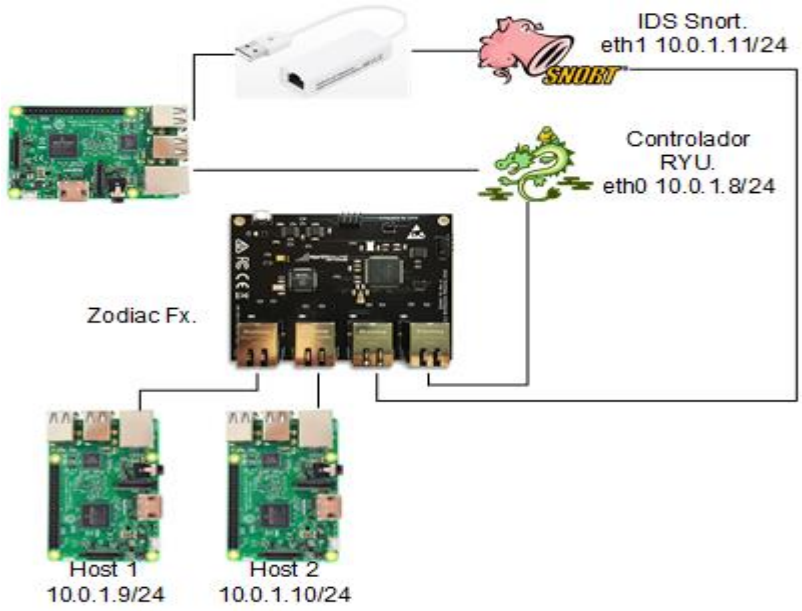

Fig. 2 **Topología a implementar [Fuente propia]**

Establecer las interfaces de red en los hosts 1 y 2 como en la **Fig. 3**:

```
# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto eth0
iface eth0 inet static
address 10.0.1.9
netmask 255.255.255.0

allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-ssid "red cnt"
wpa-psk "2019ar2019#"
```

Fig. 3 **Configuración de la interfaz Ethernet en el host 1 [Fuente propia]**

Se debe instalar Bison y Flex dos programas que funcionan en conjunto; Bison es un generador de analizadores sintácticos (compiladores) y Flex es una herramienta que

reconoce patrones léxicos en un fichero como en el caso de esta práctica donde se efectúa la lectura de textos por parte de estos programas. **sudo apt-get install bison flex**.

Instalar las dependencias de Snort como **libpcap-dev** que permite capturar paquetes en la capa de red, **libdumbnet-dev** una librería de red y **libpcre3-dev** para admitir expresiones regulares este paquete contiene los archivos de desarrollo, encabezados y documentación:

- **sudo apt-get install libpcap-dev**
- **sudo apt-get install libpcre3-dev**
- **sudo apt-get install libdumbnet-dev**

Realizar la instalación de sistema de detección de intrusos de nombre Snort, éste programa de código abierto cuenta con un lenguaje que permite la creación de reglas para monitorizar un sistema, se instala con el siguiente comando: **sudo apt-get install snort.** Después establecer la red que va a ser monitoreada por Snort, dentro del rango de la red se encuentran los hosts conectados en la tarjeta *Zodiac FX*, como se muestra en la **Fig. 4**:

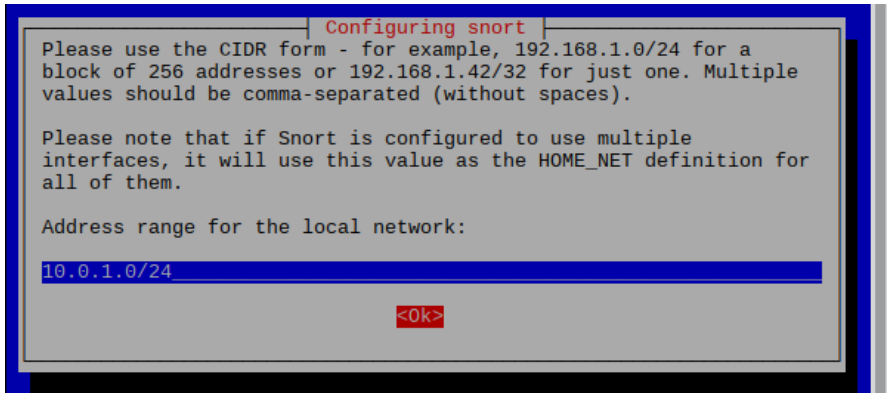

Fig. 4 **Red monitoreada por el IDS Snort [Fuente propia]**

Utilizar el comando **dpkg-reconfigure** para restablecer la configuración por defecto que se estableció en la instalación de Snort: **sudo dpkg-reconfigure snort.** Después elegir el modo manual y configurar interfaz con la que interactuará Snort la cual es **eth1**. Estos pasos se pueden observar en las **Fig. 5**, **Fig. 6**, **Fig.7**, **Fig. 8**, **Fig. 9** y **Fig. 10**:

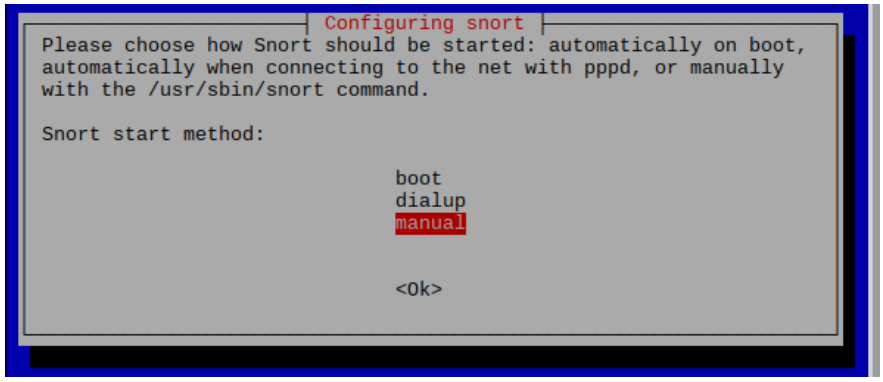

Fig. 5 **Elección del modo de inicio para Snort [Fuente propia]**

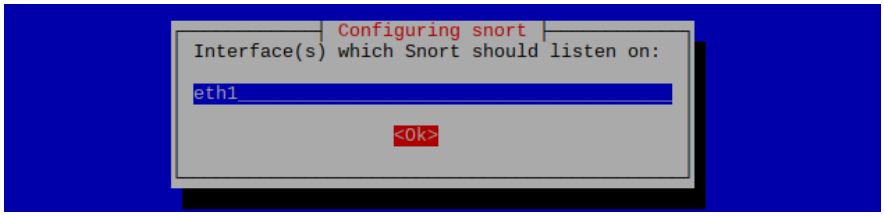

Fig. 6 **Configuración de Snort [Fuente propia]**

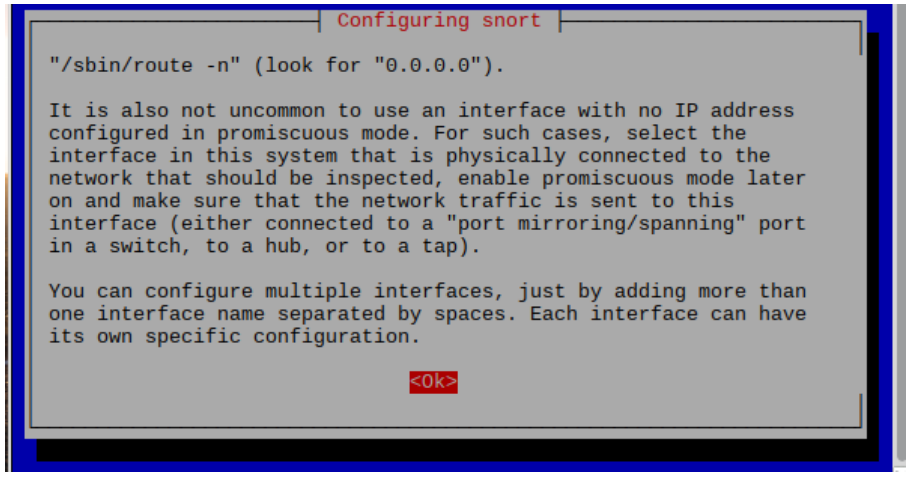

Fig. 7 **Configuración de la interfaz [Fuente propia]**

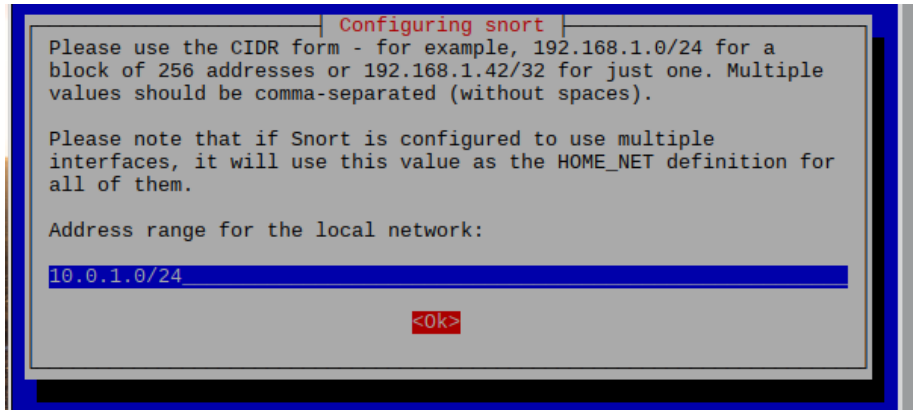

Fig. 8 **Establecimiento de la configuración [Fuente propia]**

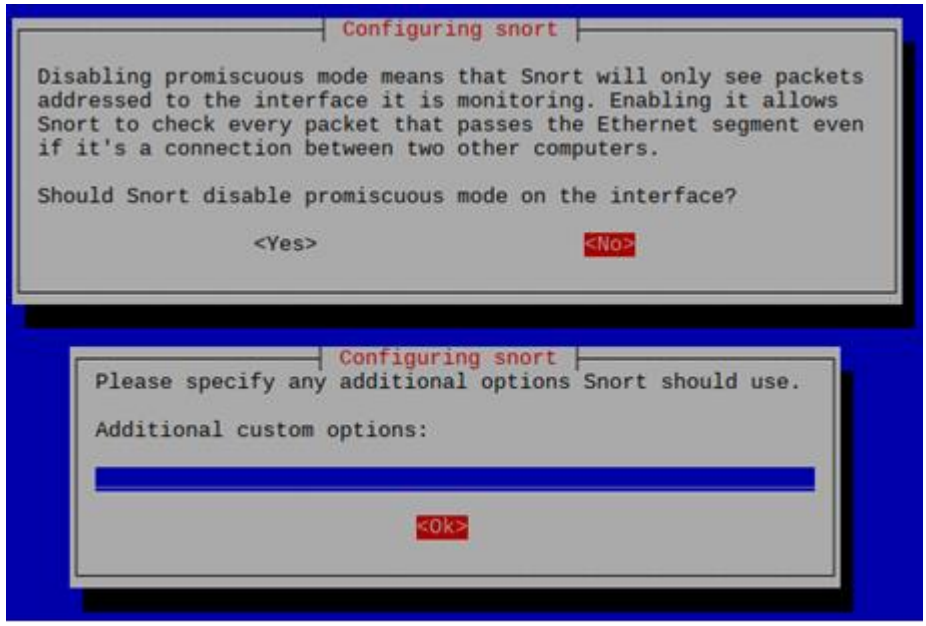

Fig. 9 **Configuración de los diferentes apartados de Snort [Fuente propia]**

Fig. 10 **Finalización de la configuración [Fuente propia]**

Se debe realizar la integración de la aplicación Pigrelay, ésta escucha los mensajes de alerta de Snort del socket del dominio UNIX (socket virtual utilizado para la

comunicación entre procesos en los sistemas operativos) y lo envía al controlador Ryu usando el socket de red. La integración se realiza con el comando: **sudo git clone https://github.com/John-Lin/pigrelay.git**.

Crear el archivo de reglas que utilizara Snort para la monitorización del sistema. Para ingresar a la dirección de *Snort* y crear las reglas se usa el comando: **sudo nano /etc/snort/rules/reglasids**.

Las cabeceras de una regla de Snort son:

<acción> <protocolo> <dirección de origen> <puerto origen> <dirección> <dirección destino> <puerto destino>

Acción: La acción es "alerta", esto confirma la generación de una alerta cuando se cumplan las condiciones establecidas. Los paquetes se registran de manera predeterminada cuando se genera una alerta.

Protocolo: En el caso de ser el protocolo ICMP, significa que la regla se aplica solo en paquetes de tipo ICMP. En el motor de detección de Snort, si el protocolo de un paquete no es ICMP, el resto de la regla no se considera para ahorrar tiempo en los procesos. La parte del protocolo juega un papel importante cuando desea aplicar las reglas de Snort sólo a paquetes de un tipo particular.

Dirección de origen y puerto de origen: Cuando se configura con la opción "any", significa que la regla se adapta a todos los paquetes que provengan de cualquier fuente.

Por supuesto, los números de puerto no tienen relevancia para los paquetes ICMP. Los números de puerto son relevantes solo cuando el protocolo es TCP o UDP.

Dirección: La dirección se establece de izquierda a derecha utilizando el símbolo ->. Esto muestra que la dirección y el número de puerto en el lado izquierdo del símbolo son el origen y los del lado derecho son el destino. También significa que la regla se aplica a los paquetes que viajan desde el origen hasta el destino. También puede usar un símbolo <- para invertir el significado de la dirección de origen y de destino del paquete. El símbolo <> también se puede utilizar para aplicar la regla en paquetes que van en cualquier dirección.

Dirección de destino y dirección del puerto: Al estar ambos configurados con "any", significa que la regla se aplica a todos los paquetes independientemente de su dirección de destino. La dirección de una regla no juega ningún papel porque la regla se aplica a todos los paquetes ICMP que se mueven en cualquier dirección, debido al uso de la palabra clave "any" en las partes de dirección de origen y destino.

Agregar las siguientes reglas al archivo “reglasids” recién creado; éstas reglas se las visualizan en la **Fig. 11**. Las reglas son:

- **alert icmp any any -> any any (msg:"Mensajes ICMP detectados"; sid:1000001;)**

- **alert tcp any any -> any any (msg:"Acceso al puerto 80"; sid:1000002;)**

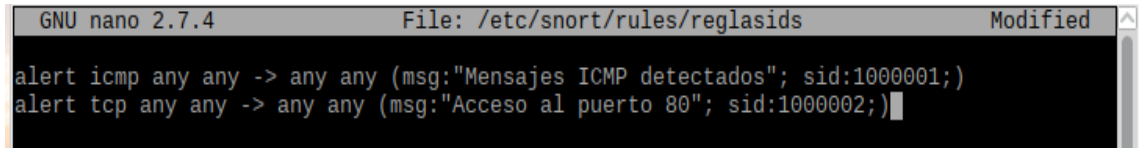

Fig. 11 **Archivo de reglas [Fuente propia]**

Editar el archivo de configuración de Snort y establecer la ruta del archivo de las reglas con el comando: **sudo nano /etc/snort/snort.conf**.

Seguido, modificar la ruta del archivo de las reglas creadas sustituyendo **local.rules** por **reglasids** el archivo que fue creado anteriormente, como se señala en la **Fig. 12**:

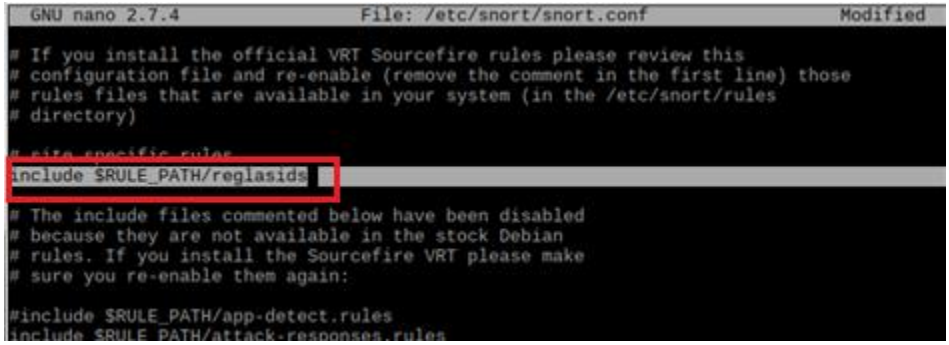

Fig. 12 **Archivo de configuración de Snort [Fuente propia]**

Ingresar al directorio del controlador Ryu: **cd ryu/ryu/app** y modificar el archivo de la aplicación que permite la comunicación entre el controlador y Snort: **sudo nano simple_switch_snort.py** con los siguientes datos:

- # Copyright (C) 2013 Nippon Telegraph and Telephone Corporation.
- # Licensed under the Apache License, Version 2.0 (the "License");

- # you may not use this file except in compliance with the License.
- # You may obtain a copy of the License at
- # http://www.apache.org/licenses/LICENSE-2.0
- # Unless required by applicable law or agreed to in writing, software
- # distributed under the License is distributed on an "AS IS" BASIS,
- # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
- # See the License for the specific language governing permissions and limitations under the License.
- **#Integración del controlador Ryu y Snort.**
- **#Importación de librerías.**
- from __future__ import print_function
- import array
- from ryu.base import app_manager
- from ryu.controller import ofp_event
- from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
- from ryu.controller.handler import set_ev_cls
- from ryu.ofproto import ofproto_v1_3
- from ryu.lib.packet import packet
- from ryu.lib.packet import ethernet
- from ryu.lib.packet import ipv4
- from ryu.lib.packet import icmp
- from ryu.lib import snortlib
- **#Inicio de la aplicación.**

- **#Definición del controlador.**
- class SimpleSwitchSnort(app_manager.RyuApp):
- OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
- _CONTEXTS = {'snortlib': snortlib.SnortLib}
- **#Definición de variables.**
- def __init__(self, *args, **kwargs):
- super(SimpleSwitchSnort, self).__init__(*args, **kwargs)
- self.snort = kwargs['snortlib']
- self.snort_port = 3
- self.mac_to_port = {}
- **#Asegurarse que el campo del socket se encuentre siempre en True.**
- socket_config = {'unixsock': True}
- self.snort.set_config(socket_config)
- self.snort.start_socket_server()
- def packet_print(self, pkt):
- pkt = packet.Packet(array.array('B', pkt))
- eth = pkt.get_protocol(ethernet.ethernet)
- _ipv4 = pkt.get_protocol(ipv4.ipv4)
- _icmp = pkt.get_protocol(icmp.icmp)
- if _icmp:
    - self.logger.info("%r", _icmp)
- if _ipv4:
    - self.logger.info("%r", _ipv4)
- if eth:
    - self.logger.info("%r", eth)

- # for p in pkt.protocols:
- # if hasattr(p, 'protocol_name') is False:
- # break
- # print('p: %s' % p.protocol_name)
- **#Alerta de paquetes entrantes hacía el controlador.**
- @set_ev_cls(snortlib.EventAlert, MAIN_DISPATCHER)
- def _dump_alert(self, ev):
- msg = ev.msg
- print('alertmsg: %s' % ''.join(msg.alertmsg))
- self.packet_print(msg.pkt)
- **#Configuración del Switch.**
- @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
- def switch_features_handler(self, ev):
- datapath = ev.msg.datapath
- ofproto = datapath.ofproto
- parser = datapath.ofproto_parser
- # install table-miss flow entry
- # We specify NO BUFFER to max_len of the output action due to
- # OVS bug. At this moment, if we specify a lesser number, e.g.,
- # 128, OVS will send Packet-In with invalid buffer_id and
- # truncated packet data. In that case, we cannot output packets
- # correctly.
- match = parser.OFPMatch()
- actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
    - ofproto.OFPCML_NO_BUFFER)]

- self.add_flow(datapath, 0, match, actions)
- def add_flow(self, datapath, priority, match, actions):
- ofproto = datapath.ofproto
- parser = datapath.ofproto_parser
- inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
    - actions)]
- mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
    - match=match, instructions=inst)
- datapath.send_msg(mod)
- @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
- def _packet_in_handler(self, ev):
- msg = ev.msg
- datapath = msg.datapath
- ofproto = datapath.ofproto
- parser = datapath.ofproto_parser
- in_port = msg.match['in_port']
- pkt = packet.Packet(msg.data)
- eth = pkt.get_protocols(ethernet.ethernet)[0]
- dst = eth.dst
- src = eth.src
- dpid = datapath.id
- self.mac_to_port.setdefault(dpid, {})
- # self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)
- **# Captura de la dirección MAC para evitar inundación.**
- self.mac_to_port[dpid][src] = in_port

- if dst in self.mac_to_port[dpid]:
    - out_port = self.mac_to_port[dpid][dst]
- else:
    - out_port = ofproto.OFPP_FLOOD
- actions = [parser.OFPActionOutput(out_port),
    - parser.OFPActionOutput(self.snort_port)]
- # **Instalación de flujos para evitar un paquete entrante.**
- if out_port != ofproto.OFPP_FLOOD:
    - match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
    - self.add_flow(datapath, 1, match, actions)
- data = None
- if msg.buffer_id == ofproto.OFP_NO_BUFFER:
    - data = msg.data
- out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
    - in_port=in_port, actions=actions, data=data)
- datapath.send_msg(out)

Iniciar el controlador *Ryu* junto con la aplicación que permite la unión entre controlador e *IDS* con el comando: **sudo ryu-manager --verbose ~/flowmanager/flowmanager.py ryu.app.simple_switch_snort**. En la **Fig. 13** se muestra la ejecución de lo mencionado anteriormente:

```
pi@raspberrypi:~ $ sudo ryu-manager --verbose ~/flowmanager/flowmanager.p
y ryu.app.simple_switch_snort
loading app /home/pi/flowmanager/flowmanager.py
require_app: ryu.topology.switches is required by flowmanager
You are using Python v3.5.3.final.0
loading app ryu.app.simple_switch_snort
loading app ryu.controller.ofp_handler
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app None of SnortLib
creating context snortlib
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app /home/pi/flowmanager/flowmanager.py of FlowManager
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.app.simple_switch_snort of SimpleSwitchSnort
[snort][INFO] {'unixsock': True}
instantiating app ryu.topology.switches of Switches
BRICK snortlib
  PROVIDES EventAlert TO {'SimpleSwitchSnort': {'main'}}
BRICK dpset
  CONSUMES EventOFPStateChange
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPSwitchFeatures
BRICK ofp_event
```

Fig. 13 **Inicio del controlador con la aplicación de Snort [Fuente propia]**

En otra terminal se debe iniciar Snort detallando que se debe monitorear la interfaz recién creada **eth1** utilizando el comando **unsock** que envía las alertas por medio de un socket para ser analizadas por otra aplicación y con la dirección de un archivo temporal **/tmp**. El comando a usar es: **sudo snort -i eth1 -A unsock -l /tmp -c /etc/snort/snort.conf**.

En la **Fig. 14** se visualiza en la terminal los mensajes de inicio exitoso de *Snort*:

```
Reload thread starting...
Reload thread started, thread 0x5da17470 (2969)
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.8
```

Fig. 14 **Inicio de Snort [Fuente propia]**

Situarse en la carpeta temporal para localizar el archivo temporal que se generó al iniciar Snort con los comandos:

- **cd /tmp**
- **ls –l**

Copiar el nombre del archivo log de Snort que se encuentra acompañado de un número, tal cual como se muestra en la **Fig. 15**:

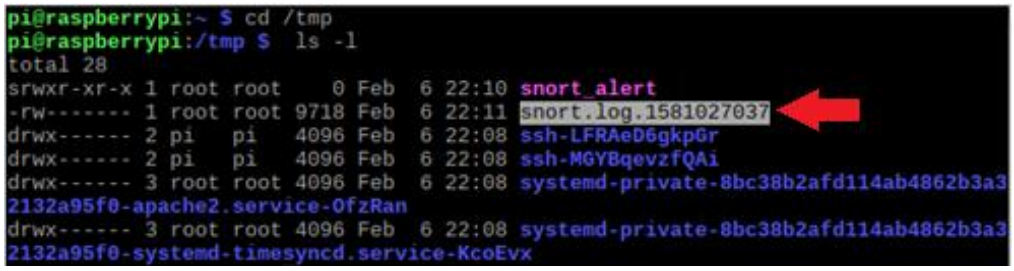

Fig. 15 **Archivo log de Snort [Fuente propia]**

Situarse en la carpeta de pigrelay y editar el archivo de configuración de la aplicación para realizar la integración entre los tres programas: controlador *Ryu*, *Snort* y pigrelay.

- **cd**
- **cd pigrelay**
- **sudo nano pigrelay.py**

En la **Fig. 16** se señala cada parte de la configuración de pingrelay:

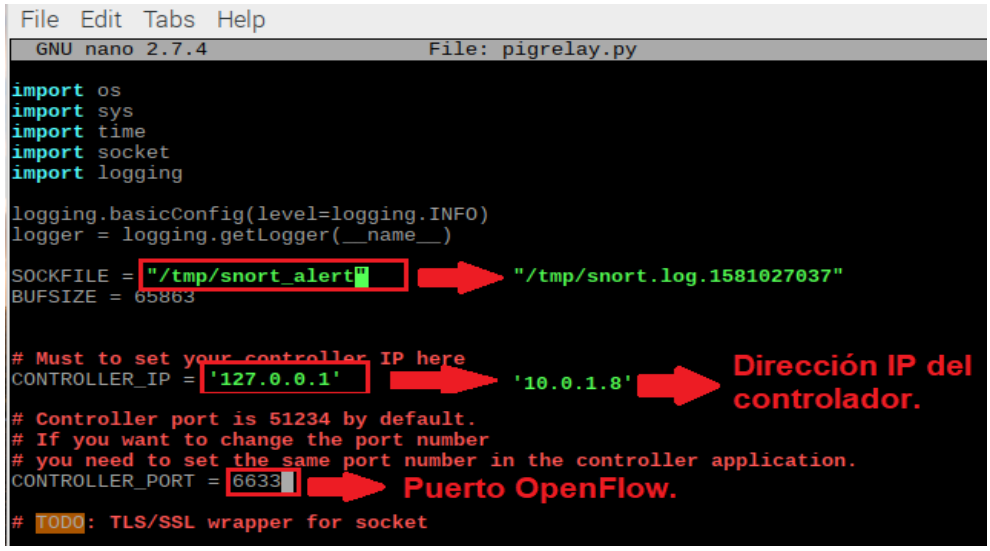

Fig. 16 **Configuración de pigrelay [Fuente propia]**

Ejecutar el archivo de pigrelay con el comando: **sudo python pigrelay.py**. En la **Fig. 17** se visualiza la conexión de pingrelay con el controlador *Ryu*:

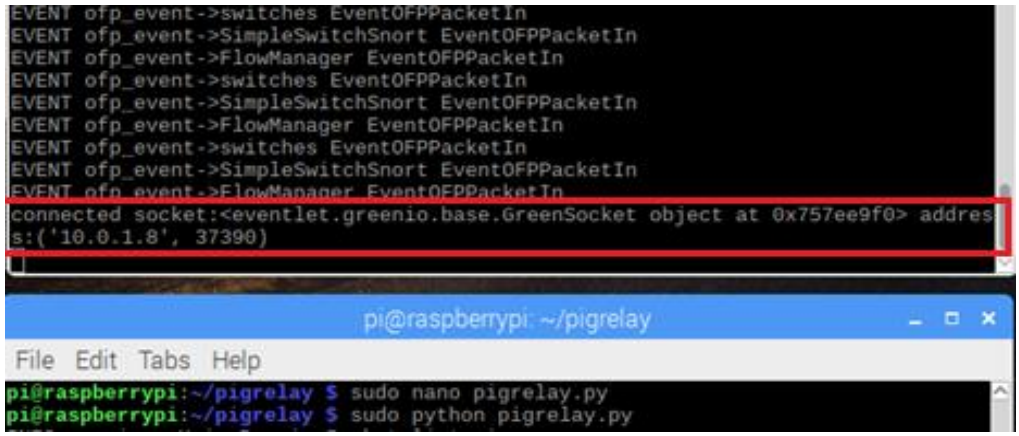

Fig. 17 **Conexión de pigrelay [Fuente propia]**

Los hosts conectados a la tarjeta *Zodiac FX* tiene interfaces de red configuradas, a ambas *Raspberry P*i se les asignaron direcciones *IP* dentro del rango de la red **10.0.1.0/24**:

- Raspberry Pi host 1 = 10.0.1.9/24
- Raspberry Pi host 2 = 10.0.1.10/24

Para comprobar el correcto funcionamiento e integración del controlador *Ryu* con *Snort*, se realiza una prueba de conectividad realizando el envío de un mensaje *ICMP* desde la terminal en el host 2 hacia el host 1, lo cual se muestra en la **Fig. 18**:

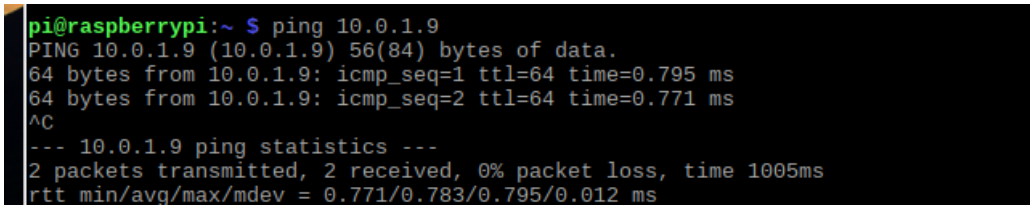

Fig. 18 **Envió de mensajes desde el host 2 al host 1 [Fuente propia]**

Como resultado se obtiene la visualización del mensaje que fue configurado para la regla de mensajes *ICMP* en el archivo **reglasids** en el controlador *Ryu*:

- **src**: Dirección *IP* de la host fuente.
- **dst**: Dirección *IP* del host destino.

En la **Fig. 19** se tienen las alertas de eventos y los mensajes configurados:

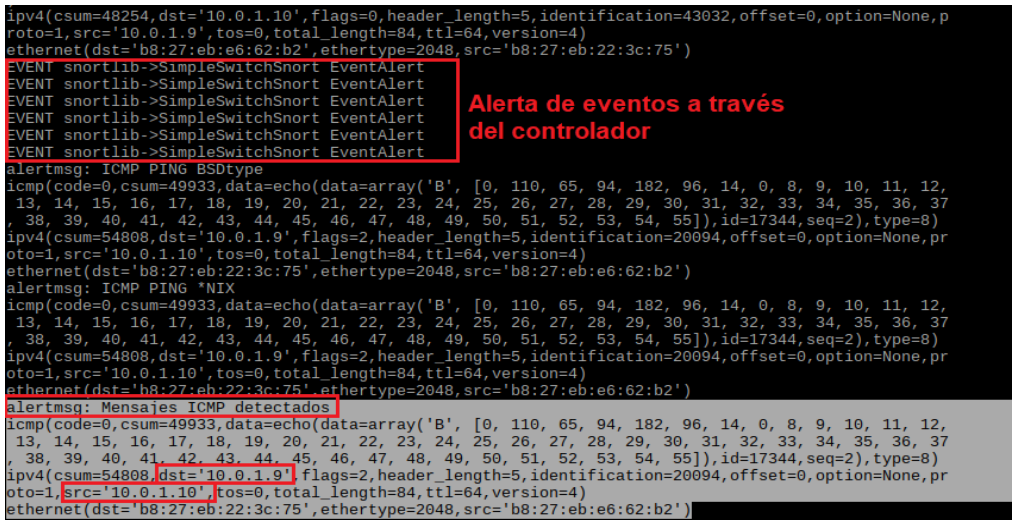

Fig. 19 **Detección de una alerta en el controlador Ryu [Fuente propia]**

Anexo J

UNIVERSIDAD POLITÉCNICA SALESIANA

Sede Guayaquil

Práctica # 10

RED DEFINIDA POR SOFTWARE CON EL CONTROLADOR RYU MEDIANTE SU APLICACIÓN FAUCET.

Objetivos:

- Analizar las tablas de flujo creadas con *Open vSwitch* y con la conexión física con la *Zodiac FX*.
- Ejecutar la aplicación *Faucet* de *Ryu*
- Configurar el archivo faucet.yaml para la conexión con la *Zodiac FX*.
- Observar y verificar en el archivo .log la conexión exitosa con la *Zodiac FX*.

Instrucciones:

- Agregar un SO (Sistema Operativo) en la *Raspberry Pi 3 Model B*.
- Instalar Grafana, Prometheus y observar cada uno de sus apartados.

Desarrollo:

Esta práctica se puede realizar con raspbian stretch en sus versiones 2017 o 2019, ambas *ISO* se encuentran en el repositorio anteriormente acordado, se debe instalar y configurar los componentes que se menciona en la **TABLA I** a continuación:

TABLA I

Componentes para la Instalación de Faucet [Fuente propia].

Componente.	**Propósito.**
Faucet.	Controlador de red
Gauge.	Controlador de monitoreo.
Prometheus.	Sistema de monitoreo y base de datos.
Grafana.	Panel de control.

Como primer paso es actualizar el contenido del sistema de la *Raspberry Pi* con el siguiente comando: **sudo apt-get update**.

Como siguiente paso es agregar el repositorio oficial de faucet a nuestro sistema para después realizar la instalación con los siguientes comandos:

- **sudo apt-get install curl gnupg apt-transport-https lsb-release**

- **echo "deb https://packagecloud.io/faucetsdn/faucet/$(lsb_release -si | awk '{print tolower($0)}')/ $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/faucet.list**

- **curl -L https://packagecloud.io/faucetsdn/faucet/gpgkey | sudo apt-key add –**

- **sudo apt-get update**

Después, se instalan los paquetes que se requiere, en este caso se usó faucet-all-in-one, el cual es un paquete que permite la instalación de todas las dependencias correctas para su debido funcionamiento, como se visualiza en la **Fig. 1**, el comando a usar es el siguiente: **sudo apt-get install faucet-all-in-one**.

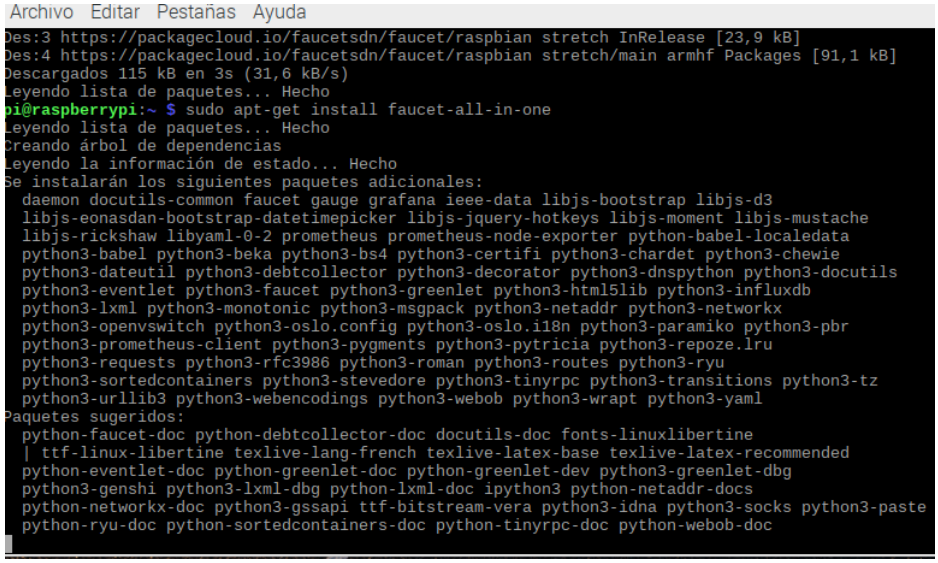

Fig. 1 **Instalación de Faucet [Fuente propia]**

Una vez haya terminado la instalación de *Faucet*, se procede a configurar *Prometheus*, específicamente el archivo "**prometheus.yml**", pero, *Faucet* nos otorga un archivo de muestra que está configurado para que *Faucet* y Gauge que se ejecuten en la misma máquina que *Prometheus*. Para observar el archivo usamos el siguiente comando: **sudo vi /etc/faucet/prometheus/prometheus.yml**. En la **Fig. 2** se muestra el contenido del archivo "prometheus.yml":

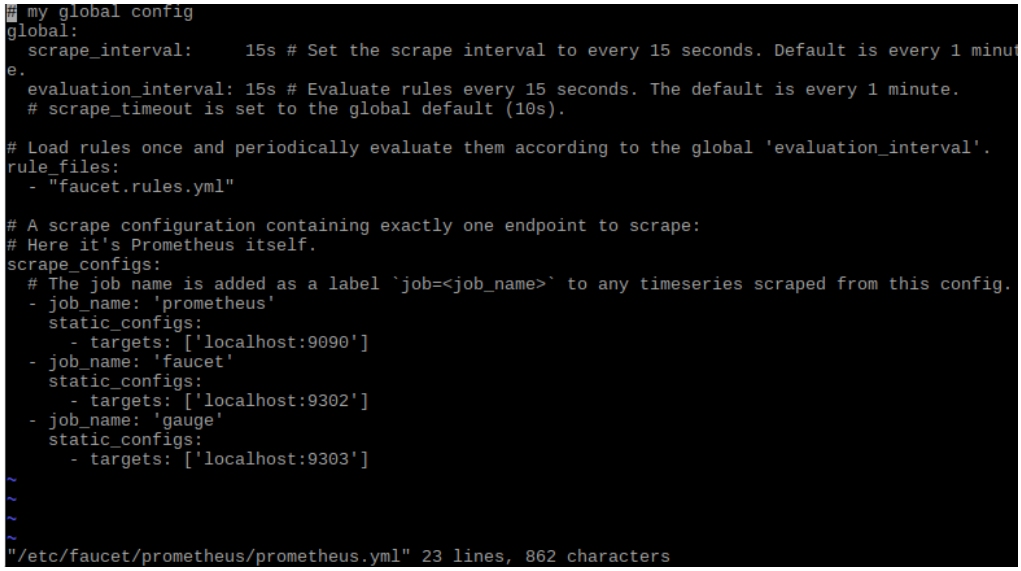

Fig. 2 **Prometheus.yml [Fuente propia]**

Para tener más conocimiento sobre este archivo, se puede observar el archivo faucet.rules.yml, que está ubicado en la misma dirección que el archivo anterior. En la **Fig. 3** se presenta el contenido e información del archivo faucet.rule.yml. Para acceder a éste se usa el siguiente comando: **sudo vi /etc/faucet/prometheus/faucet.rules.yml**.

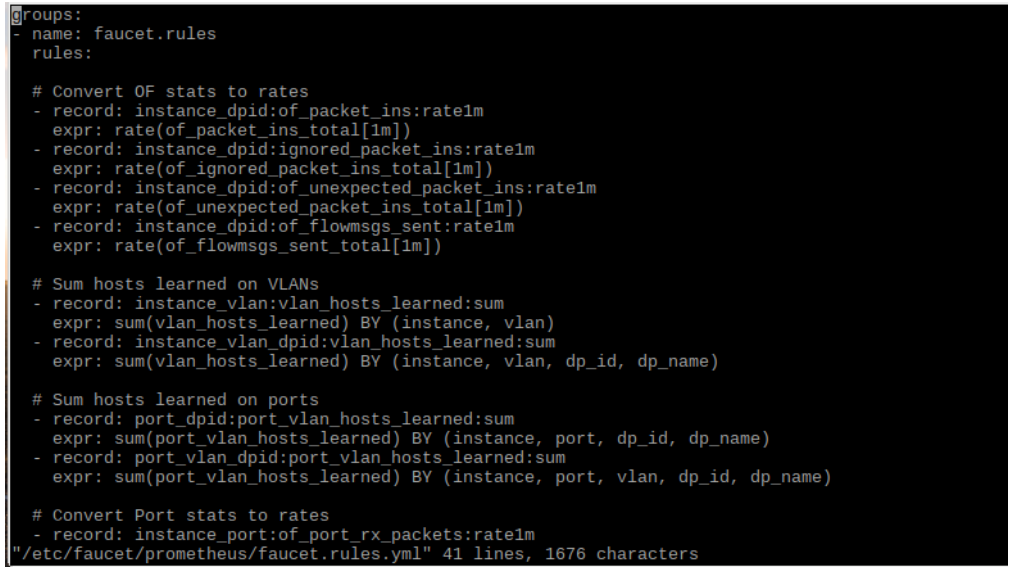

Fig. 3 **Faucet.rules.yml [Fuente propia]**

Después hacer que prometheus use el archivo de configuración con faucet agregando “—config.file=/etc/faucet/prometheus/prometheus.yml”, como se señala en la **Fig. 4.** El comando a usar es el siguiente: **sudo nano /etc/default/prometheus**.

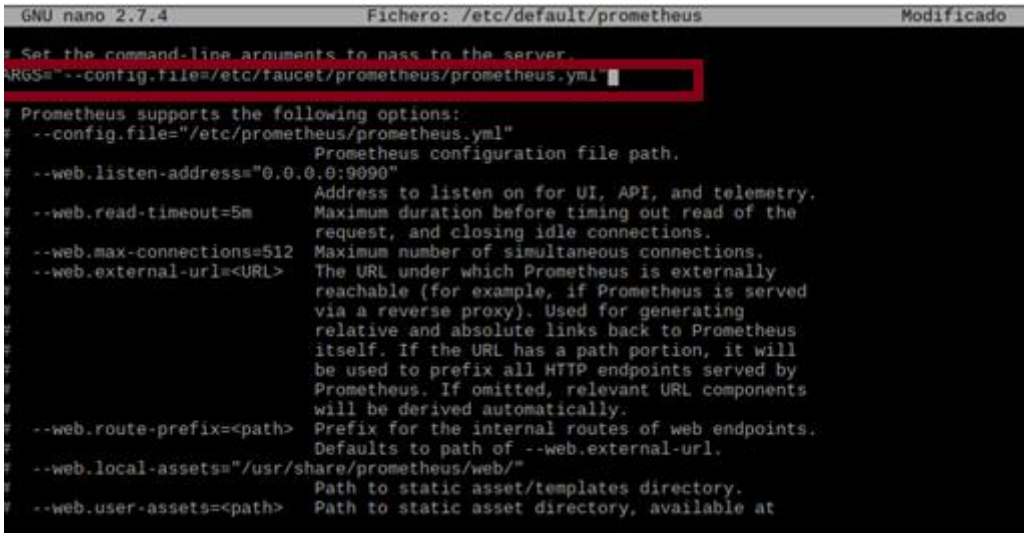

Fig. 4 **Edición de prometheus [Fuente propia]**

Para la finalización de la configuración de Prometheus, lo último a realizar es reiniciar Prometheus para aplicar los cambios: **sudo systemctl restart prometheus**.

Ahora hay que configurar Grafana, para ello debe de iniciar en el arranque y luego iniciarlo manualmente, esto se puede realizar con los siguientes comandos:

- **sudo systemctl daemon-reload**
- **sudo systemctl enable grafana-server**
- **sudo systemctl start grafana-server**

En la **Fig. 5** se muestra el uso de los comandos mencionados anteriormente:

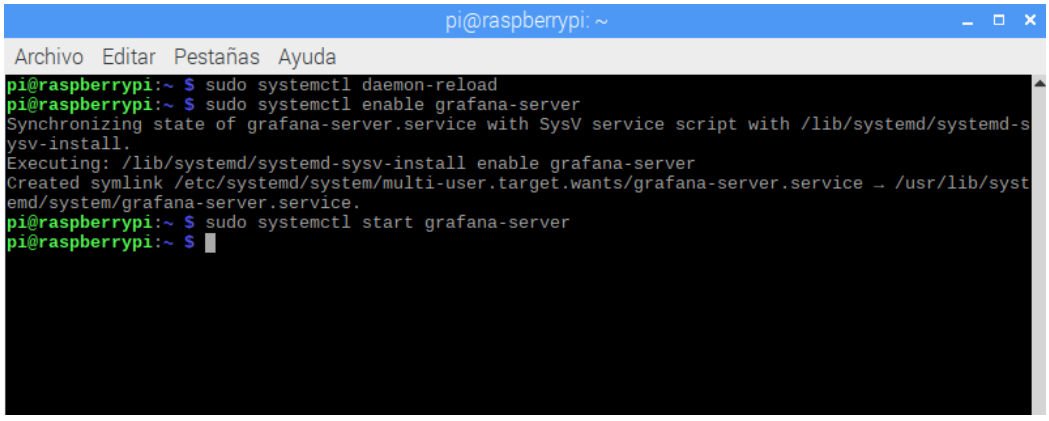

Fig. 5 **Configuración de Grafana en el arranque [Fuente propia]**

Una vez hecho esto, se procede a realizar la configuración de Grafana por medio de su interfaz web. Se accede con el enlace "***http://localhost:3000***", como se observa en la **Fig. 6**, las credenciales son las siguientes:

- Usuario: admin
- Contraseña: admin

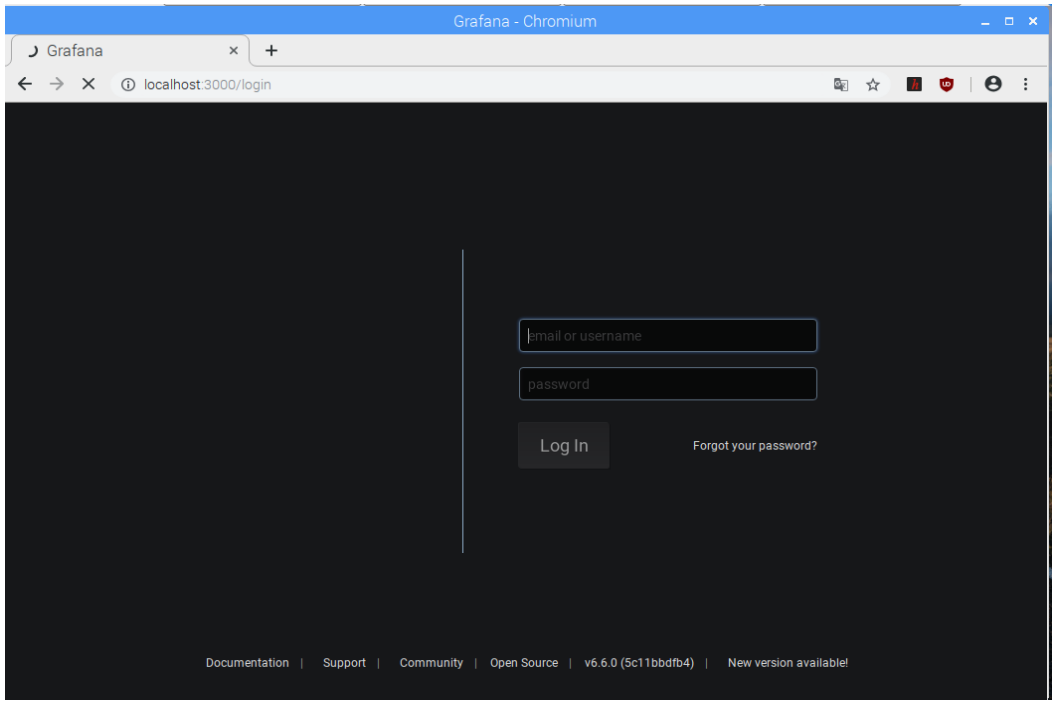

Fig. 6 **Login de Grafana [Fuente propia]**

Se debe de agregar una fuente de datos, en el apartado Name se escribe ***Prometheus***, en URL se escribe ***http://localhost:9090*** como se señala en la **Fig. 7**:

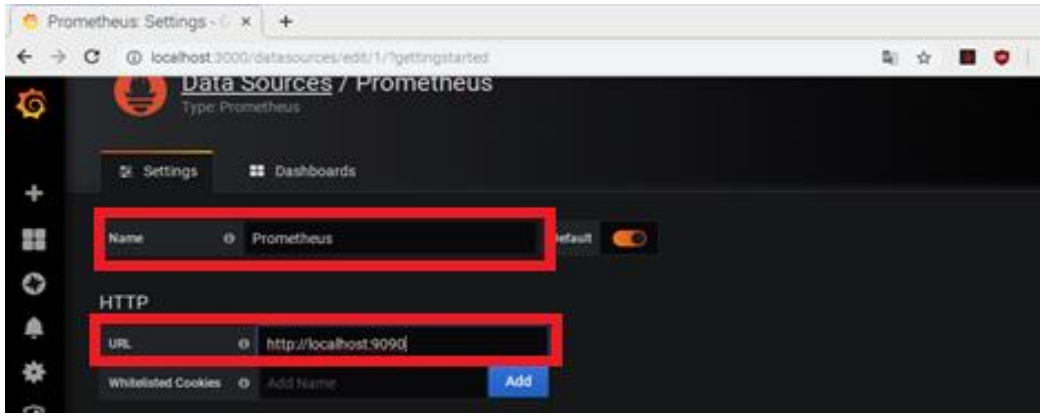

Fig. 7 **Interfaz web de Grafana [Fuente Propia]**

Posteriormente, se debe de agregar ciertos paneles para luego poder ver las métricas de *Faucet*. En la interfaz web hay un ícono "+" el cual se debe de dar click en el ícono y luego click en importar. Estos pasos se los muestran en las **Fig. 8**, **Fig. 9** y **Fig. 10**:

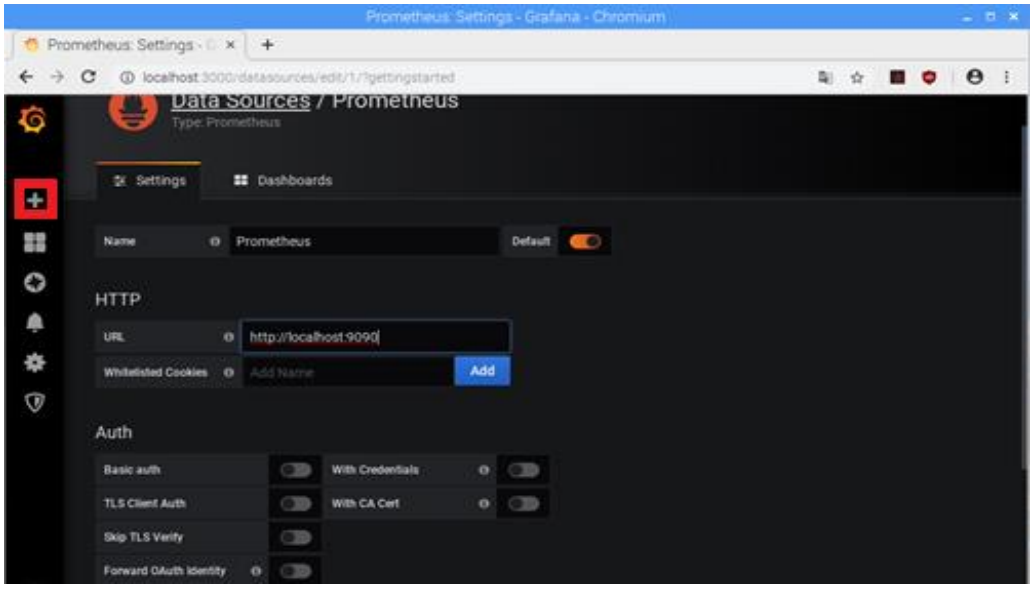

Fig. 8 **Adición de paneles [Fuente propia]**

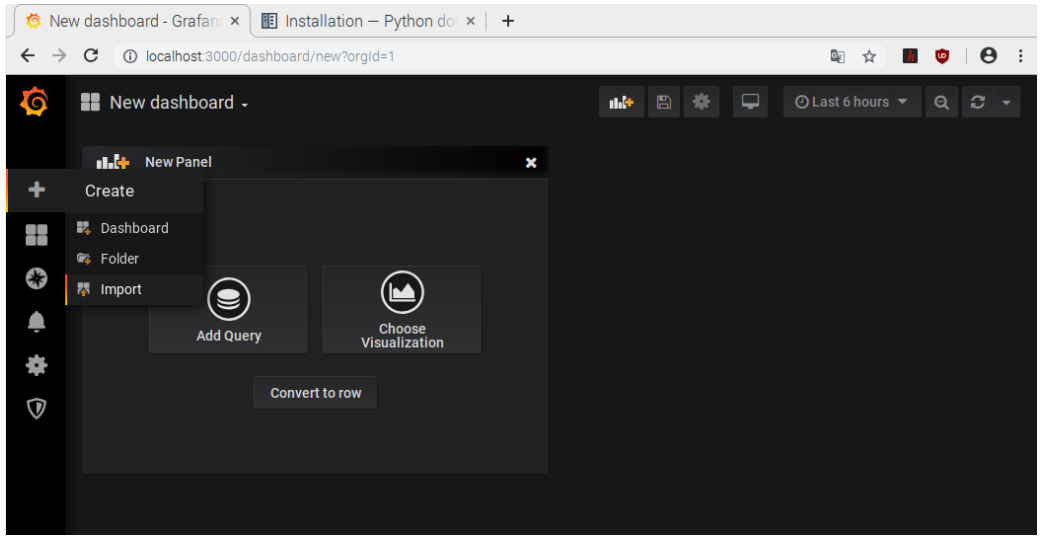

Fig. 9 **Importación de paneles [Fuente propia]**

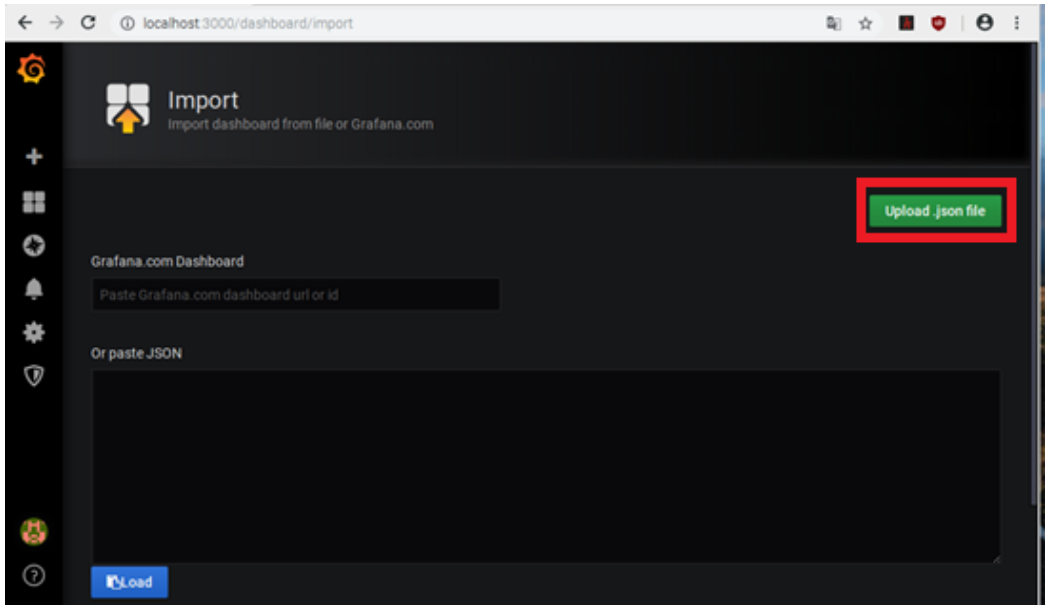

Fig. 10 **Subida de archivos .json [Fuente propia]**

Para la importación de los paneles, solamente se debe de descargar los siguientes enlaces y cargarlos a través de la pantalla de importación del panel Grafana, los archivos los podremos descargar desde el repositorio anteriormente acordado. Los

archivos son: faucet_instrumentation.json, faucet_inventory.json y faucet_port_statistics.json. Después de subir los archivos uno por uno, se debe de seguir los pasos tal cual están en las **Fig. 11**, **Fig. 12**, **Fig. 13** y **Fig. 14**:

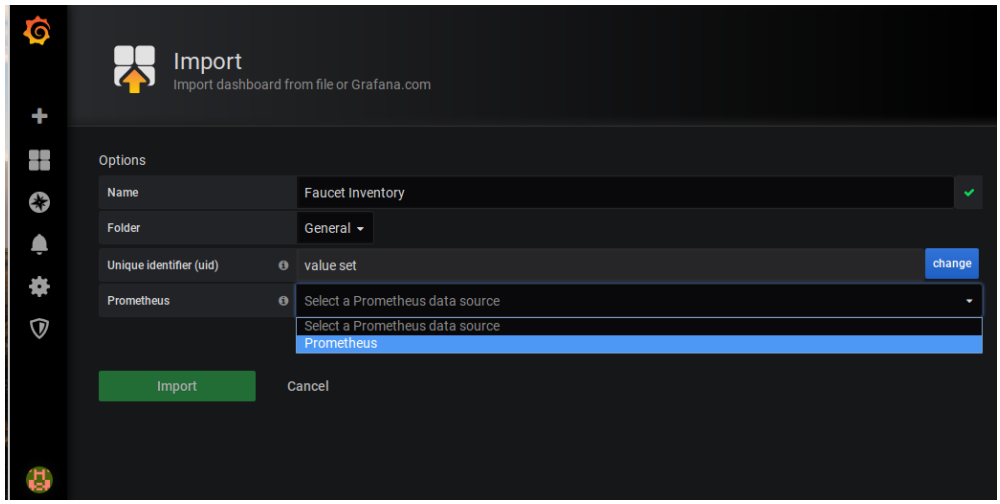

Fig. 11 **Opciones en la importación [Fuente propia]**

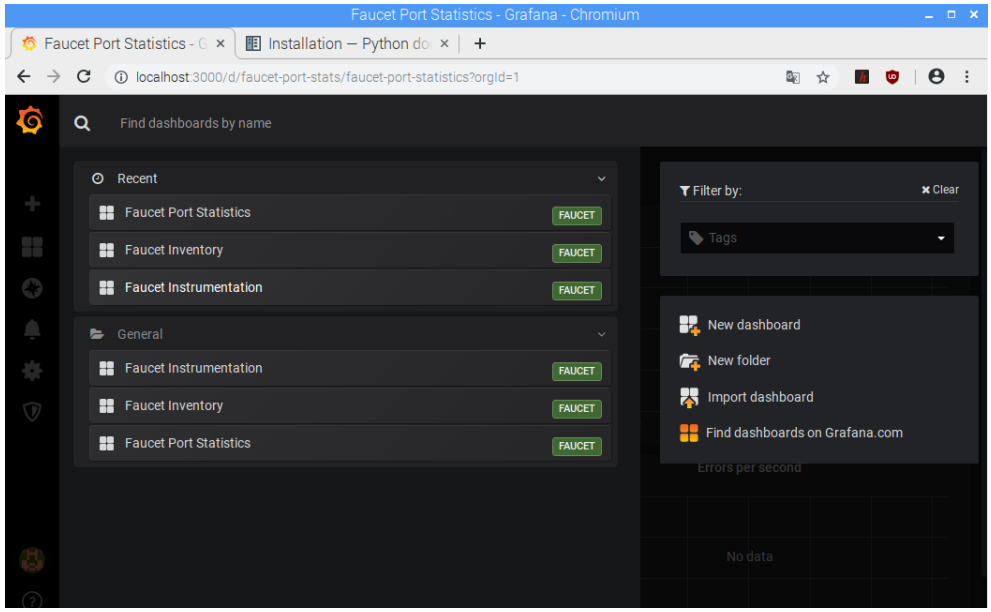

Fig. 12 **Lista de paneles agregados [Fuente propia]**

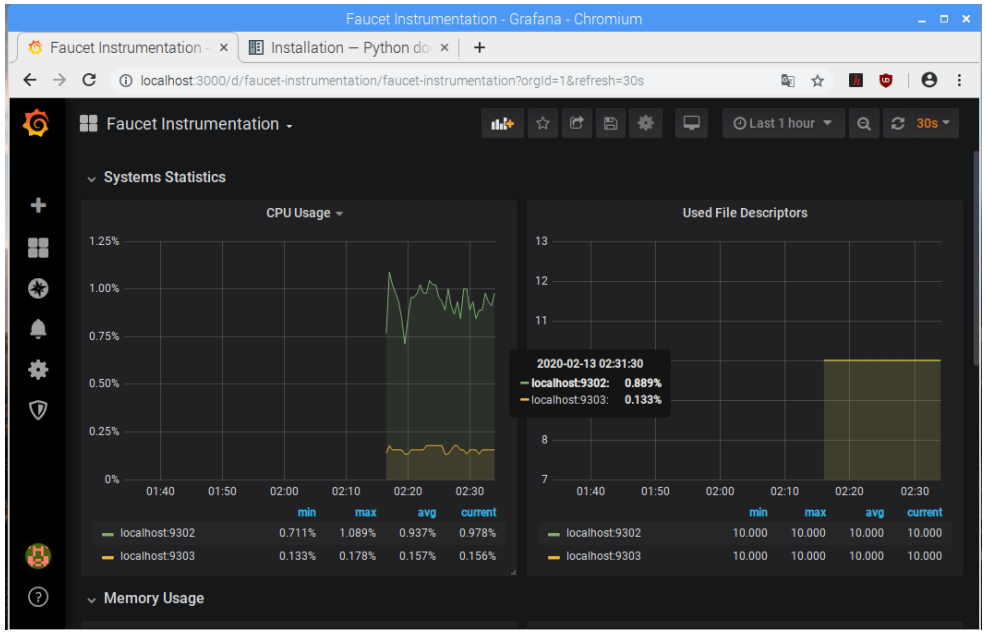

Fig. 13 **Faucet Instrumentation [Fuente propia]**

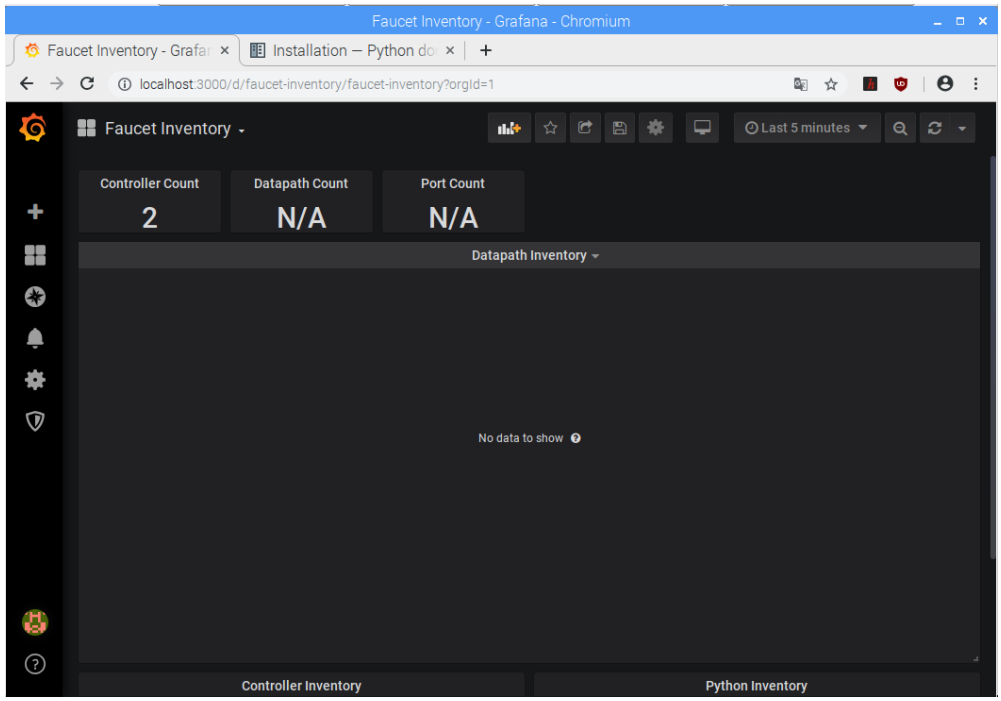

Fig. 14 **Faucet Inventory [Fuente propia]**

Ahora hay que configurar Faucet, en este caso primero usaremos Open vSwitch para después observar los datos en el archivo .log. Para la configuración se debe de editar el archivo faucet.yaml (en el cual se debe de cerciorar la ubicación que por defecto es /etc/faucet/faucet.yaml), el cual contiene cierta información sobre la topología y las VLANs, para editar este archivo se debe de usar el siguiente comando: **sudo nano /etc/faucet/faucet.yaml**. En la **Fig. 15** se muestra el contenido del archivo faucet.yaml:

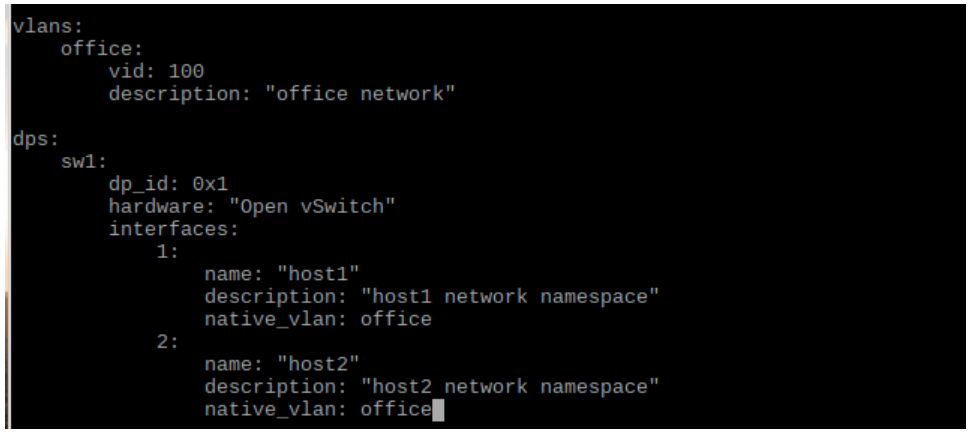

Fig. 15 **Faucet.yaml [Fuente propia]**

Mediante la configuración dada, se crea solamente una ruta *VLAN* y una ruta de datos con dos puertos, en este caso solamente basta usar dos puertos tanto en la conexión con *Open vSwitch* o con la conexión con la tarjeta *Zodiac FX*. Para verificar si la configuración es correcta se debe de usar el siguiente comando **check_faucet_config /etc/faucet/faucet.yaml** y si todo está correctamente configurado, un mensaje que contiene toda la configuración de Faucet. En la **Fig. 16** se detalla la correcta configuración del archivo:

***Fig. 16* Verificación de la configuración de faucet.yaml [Fuente propia]**

Para comprobar y demostrar un error en el archivo faucet.yaml, cambiar la *VLAN ID* de 100 a 1000 y después correr el comando para confirma la configuración:

- **sudo nano /etc/faucet/faucet.yaml**

- **check_faucet_config /etc/faucet/faucet.yaml**

Después de observar el mensaje de error, lo editamos el archivo faucet.yaml con los valores iniciales (VLAN ID = 100) y recargar *Faucet* con el siguiente comando: **sudo systemctl reload faucet**.

Para verificar los registros del archivo faucet.log, hay que estar seguros en donde se encuentra el archivo, este archivo de log se encuentra en la siguiente dirección: "**/var/log/faucet/faucet.log"**. Para observar los logs de *Faucet* en el archivo hay que usar el siguiente comando: **les /var/log/faucet/faucet.log**. En la **Fig. 17** se muestran los mensajes del archivo faucet.log:

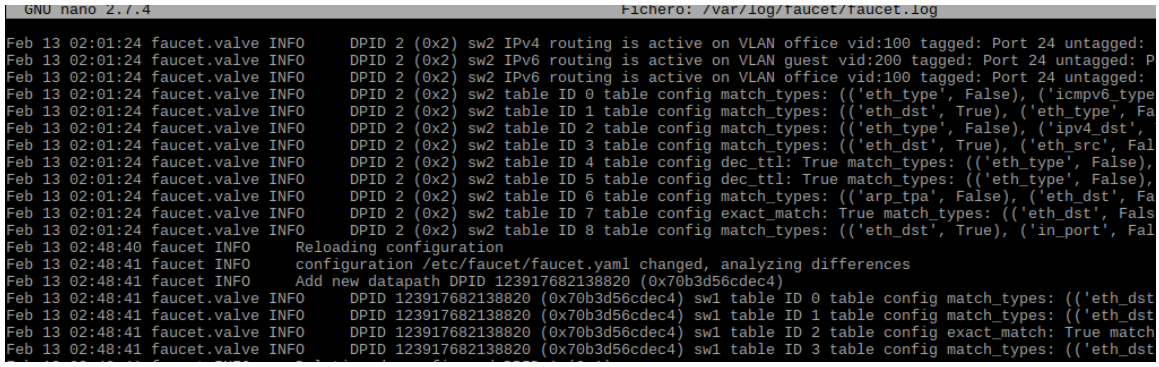

Fig. 17 **Faucet.log [Fuente propia]**

Seguido a eso se procede a configurar *Gauge*, pero no necesitamos editar la configuración predeterminada que se envía con *Faucet* porque esta configuración es suficiente para cumplir nuestro propósito. Si se requiere de alguna modificación del archivo la ruta es **"/etc/faucet/gauge.yaml"**, para poder revisar la configuración predeterminada lo haremos con el siguiente comando: **sudo nano /etc/faucet/gauge.yaml**. En la **Fig. 18** se visualiza el archivo gauge.yaml:

```
# Recommended configuration is Prometheus for all monitoring, with all_dps: True
faucet_configs:
    - '/etc/faucet/faucet.yaml'
watchers:
    port_status_poller:
        type: 'port_state'
        all_dps: True
        db: 'prometheus'
    port_stats_poller:
        type: 'port_stats'
        all_dps: True
        interval: 10
        db: 'prometheus'
        #db: 'influx'
    flow_table_poller:
        type: 'flow_table'
        all_dps: True
        interval: 60
        db: 'prometheus'
dbs:
    prometheus:
        type: 'prometheus'
        prometheus_addr: '0.0.0.0'
        prometheus_port: 9303
    ft_file:
        type: 'text'
        compress: True
        path: 'flow_tables'
    influx:
        type: 'influx'
        influx_db: 'faucet'
        influx_host: 'influxdb'
        influx_port: 8086
        influx_user: 'faucet'
        influx_pwd: 'faucet'
        influx_timeout: 10
```

Fig. 18 **Gauge.yaml [Fuente propia]**

Necesitaremos reiniciar *Gauge* para poder retomar los otros pasos: **sudo systemctl restart gauge**.

Ahora se procede a realizar la conexión de la ruta de datos, primero se evalúa con *Open vSwitch*, por lo que hay que instalar dicho componente de la siguiente forma:

- **sudo apt-get install apt-transport-https**
- **echo "deb https://packages.wand.net.nz $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/wand.list**
- **sudo curl https://packages.wand.net.nz/keyring.gpg -o /etc/apt/trusted.gpg.d/wand.gpg**

- **sudo apt-get update**

- **sudo apt-get install openvswitch-switch**

Se utilizan dos espacios de nombres de red de Linux para poder suponer hosts que permite crear tráfico en nuestra red, como se muestra en la **Fig. 19**. Los comandos son:

- **as_ns () {**

- **NAME=$1**

- **NETNS=faucet-${NAME}**

- **shift**

- **sudo ip netns exec ${NETNS} $@**

- **}**

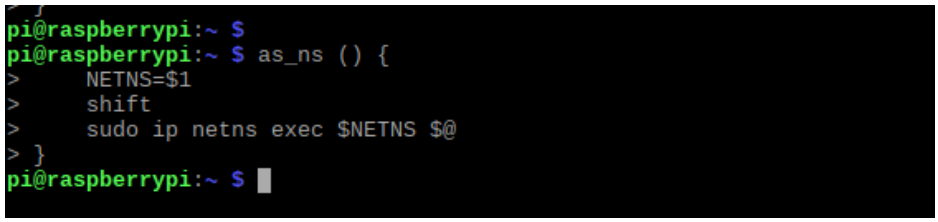

Fig. 19 **Espacios de red [Fuente propia]**

Se crea otro espacio de red, como se detalla en la **Fig. 20**, con los comandos:

- **create_ns () {**
- **NAME=$1**
- **IP=$2**
- **NETNS=faucet-${NAME}**
- **sudo ip netns add ${NETNS}**
- **sudo ip link add dev veth-${NAME} type veth peer name veth0 netns ${NETNS}**
- **sudo ip link set dev veth-${NAME} up**
- **as_ns ${NAME} ip link set dev lo up**
- **[-n "${IP}"] && as_ns ${NAME} ip addr add dev veth0 ${IP}**
- **as_ns ${NAME} ip link set dev veth0 up**
- **}**

```
pi@raspberrypi:~ $ create_ns () {
>     NETNS=$1
>     IP=$2
>     sudo ip netns add ${NETNS}
>     sudo ip link add dev veth-${NETNS} type veth peer name veth0 netns $NETNS
>     sudo ip link set dev veth-${NETNS} up
>     sudo ip netns exec $NETNS ip link set dev veth0 up
>     sudo ip netns exec $NETNS ip addr add dev veth0 $IP
>     sudo ip netns exec $NETNS ip link set dev lo up
> }
pi@raspberrypi:~ $
```

Fig. 20 **Creación de espacio de red [Fuente propia]**

Ahora se deben de crear hosts y asignarles sus direcciones *IP* tal cual se ve en la **Fig. 21***:*

- **create_ns host1 192.168.0.1/24**
- **create_ns host2 192.168.0.2/24**
- **ip netns list**

```
> }
pi@raspberrypi:~ $ create_ns host1 192.168.0.1/24
pi@raspberrypi:~ $ create_ns host2 192.168.0.2/24
pi@raspberrypi:~ $ ip netns list
host2 (id: 1)
host1 (id: 0)
pi@raspberrypi:~ $
```

Fig. 21 **Hosts y direcciones IP [Fuente propia]**

Para configurar un puente *Open vSwitch*, que sirve como la ruta de datos, se debe de agregar dos puertos:

- **sudo ovs-vsctl add-br br0 **
- **set bridge br0 other-config:datapath-id=0000000000000001 **
- **set bridge br0 other-config:disable-in-band=true **
- **set bridge br0 fail_mode=secure **
- **add-port br0 veth-host1 -- set interface veth-host1 ofport_request=1 **
- **add-port br0 veth-host2 -- set interface veth-host2 ofport_request=2 **
- **set-controller br0 tcp:127.0.0.1:6653 tcp:127.0.0.1:6654**

Para mostrar la configuración se usa el siguiente comando, como se detalla en la **Fig. 22**: **sudo ovs-vsctl show**

Fig. 22 **Configuración de Open vSwitch [Fuente propia]**

En la interfaz web de *Grafana* debe aparecer una nueva ruta de datos, la cual es la que sea creó con los comandos anteriores, esto se señala en la **Fig. 23**:

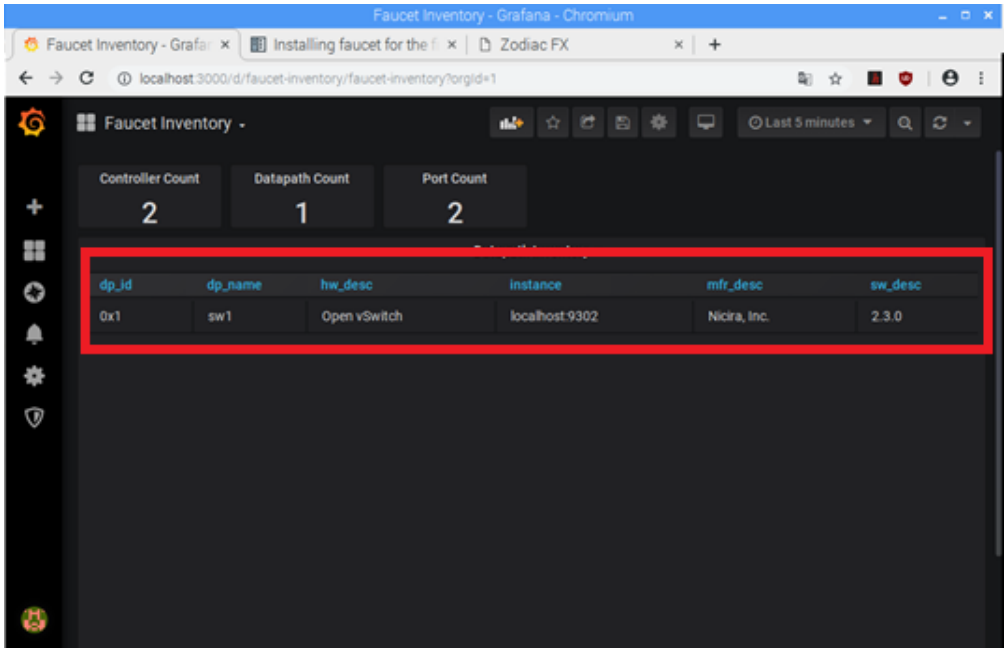

Fig. 23 **Ruta de datos [Fuente propia]**

En el archivo de faucet.log se deben de observar los mensajes de la ruta de datos registrada en la *Zodiac FX*, tal cual en la **Fig. 24**:

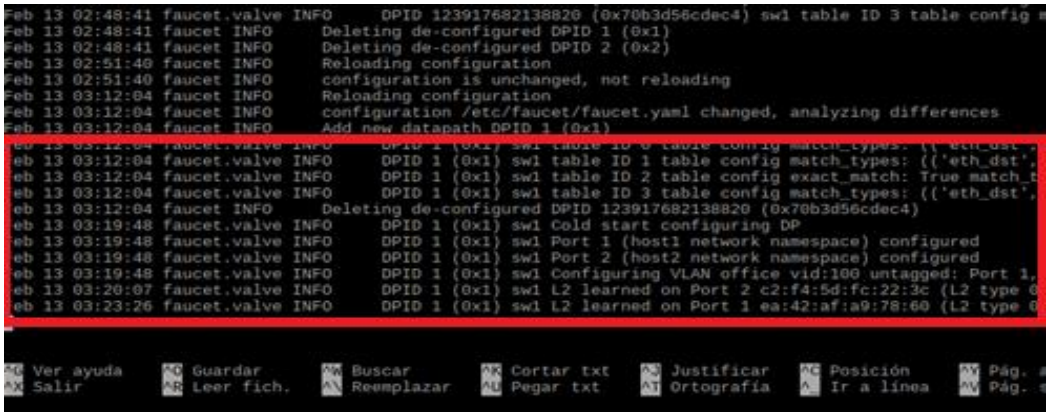

Fig. 24 **Registro de ruta de datos [Fuente propia]**

Hacemos ping a cualquier host creado, en este caso al host 2 y después revisar el archivo faucet.log nuevamente para verificar la siguiente información que se señala en la **Fig. 25**. Se realiza el ping con el siguiente comando: **ping 192.168.0.2**.

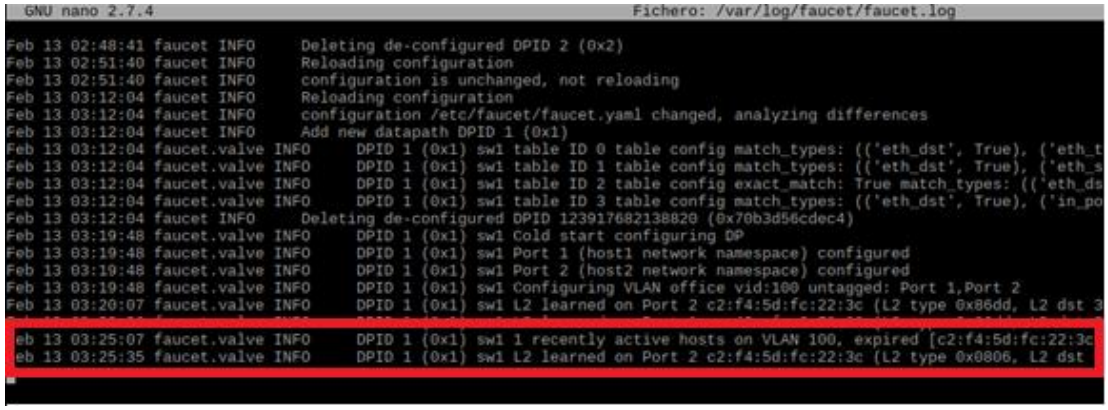

Fig. 25 **Reenvío de tráfico [Fuente propia]**

Una vez realizados los pasos anteriores, se procede a realizar la conexión con la tarjeta *Zodiac FX,* lo importante es editar el archivo "**faucet.yaml**" para que pueda permitir la conexión con dicho dispositivo, para la configuración basta con cambiar

"**Open vSwitch**" por "**ZodiacFX**"; no se debe de escribir "*Zodiac FX"* de forma separada debido a la compatibilidad, debe de escribirse de la forma correcta para que se realice de forma correcta la conexión. Para esto se usa **check_faucet_config** para demostrar que el archivo .yaml se encuentra configurado correctamente. En la **Fig. 26** se señala la parte a editar:

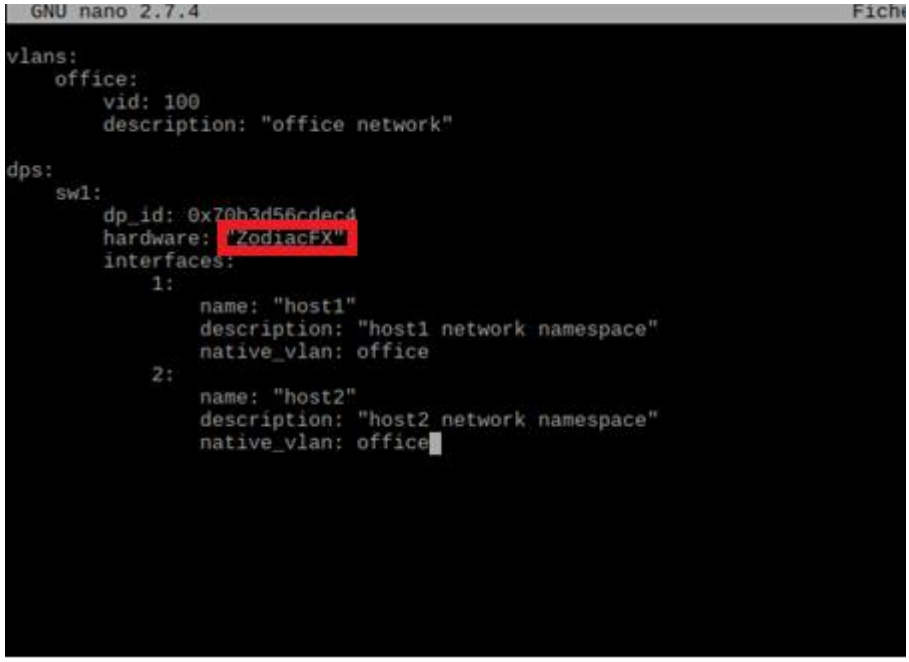

Fig. 26 **Edición del archivo faucet.yaml [Fuente propia]**

Antes de correr la aplicación *Faucet*, hay que paralizar algún proceso relacionado con *Ryu* ya que *Faucet* es una aplicación de *Ryu*: **sudo killall ryu-manager**.

Antes de correr la aplicación de *Faucet,* se debe de copiar la aplicación *Faucet* a la carpeta de *Ryu* con el comando**: sudo cp /usr/lib/python3/dist-packages/faucet/faucet.py /usr/lib/python3/dist-packages/ryu/app**. Para correr la

aplicación *Faucet* se usa el siguiente comando: **sudo ryu-manager --verbose ryu.app.faucet**. En la **Fig. 27** se muestran los mensajes de ejecución:

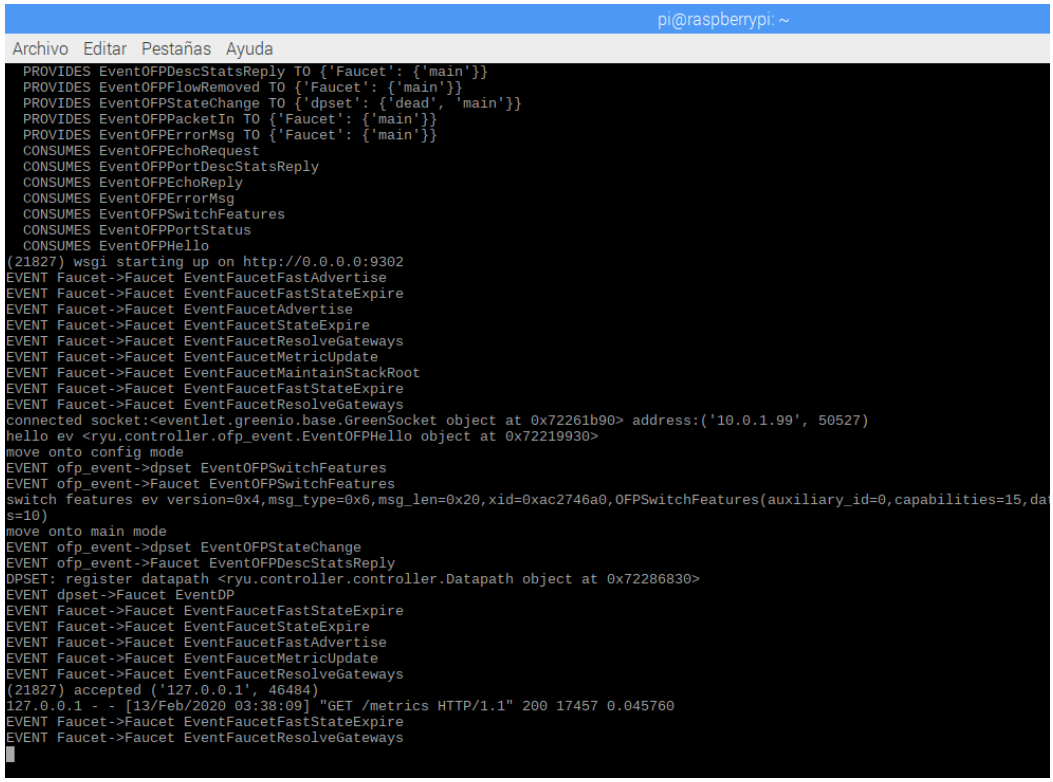

Fig. 27 **Ejecución de Faucet [Fuente propia]**

Se debe de relizar ping entre cualquier host. Después, verificar las tablas de flujo de datos por medio de la interfaz web de la tarjeta *Zodiac FX* o con *Putty*. En la **Fig. 28** se visualizan los flujos instalados en la *Zodiac FX* por medio de la interfaz web.

Para ingresar a la interfaz se debe de colocar la dirección *IP* de la tarjeta *Zodiac FX,* la cual es "**10.0.1.99**" en el navegador y en el apartado **"Flows"** se alojan los datos de las tablas de flujos:

Fig. 28 **Tablas de flujos de datos [Fuente propia]**

En el archivo log de *Faucet* se deben visualizar los mensajes y más información detallado sobre las tablas de flujos de datos, como lo muestra la **Fig. 29**:

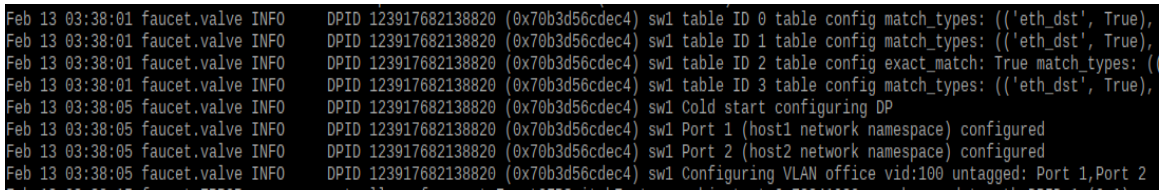

Fig. 29 **Información de tablas de flujo de datos [Fuente propia]**

Por lo siguiente, se comprueba la comunicación entre los hosts y el funcionamiento de la aplicación Faucet.