



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**  
**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y**  
**REDES DE COMUNICACIÓN**

**TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE**  
**INGENIERA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN**

**TEMA**

**DISEÑO DE UNA RED DEFINIDA POR SOFTWARE EMPLEANDO UNA SO-**  
**LUCIÓN BASADA EN SOFTWARE, PARA LA INFRAESTRUCTURA DE**  
**CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS**  
**APLICADAS (FICA)**

**AUTOR: JOHANNA LISSETH LEYTON PORTILLA**

**DIRECTOR: Ing. CARLOS VÁSQUEZ**

**Ibarra- Ecuador**

**2016**



## UNIVERSIDAD TÉCNICA DEL NORTE

### BIBLIOTECA UNIVERSITARIA

#### AUTORIZACIÓN DE USO Y PUBLICACIÓN

#### A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

#### 1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información.

| <b>DATOS DEL CONTACTO</b> |   |
|---------------------------|---|
| Cédula de identidad       | 040161773-3   |
| Apellidos y Nombres       | Leyton Portilla Johanna Lisseth   |
| Dirección                 | Ibarra – Grijalva 6-34 y Bolivar  |
| E-mail                    | johannaleypo@hotmail.com  |
| Teléfono fijo             | 062-642-482   |
| Teléfono móvil            | 0995087966  |
| <b>DATOS DE LA OBRA</b>   |   |
| Título                    | Diseño de una red definida por software empleando una solución basada en software, para la infraestructura de Cloud de la facultad de Ingeniería en Ciencias Aplicadas (FICA) |
| Autora                    | Leyton Portilla Johanna Lisseth   |
| Fecha                     | Noviembre 2016  |
| Programa                  | Pregrado  |
| Título                    | Ingeniera en Electrónica y Redes de Comunicación  |
| Director                  | Ing. Carlos Vásquez   |

## **2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD**

Yo, Leyton Portilla Johanna Lisseth, con cédula de identidad Nro. 04016173-3, en calidad de autora y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad de material y como apoyo a la educación, investigación y extensión, en concordancia con la ley de Educación Superior Artículo 144.

## **3. CONSTANCIAS**

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Firma: 

Nombre: Leyton Portilla Johanna Lisseth

Cédula: 040161773-3

Ibarra, Noviembre de 2016



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**

**CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA**  
**UNIVERSIDAD TÉCNICA DEL NORTE**

Yo, Leyton Portilla Johanna Lisseth con cédula de identidad número 040161773-3 manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador artículos 4, 5 y 6, en calidad de la autora del trabajo de grado con el tema: “DISEÑO DE UNA RED DEFINIDA POR SOFTWARE EMPLEANDO UNA SOLUCIÓN BASADA EN SOFTWARE, PARA LA INFRAESTRUCTURA DE CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS (FICA)”. Que ha sido desarrollado con el propósito de obtener el título de Ingeniera en Electrónica y Redes de Comunicación de la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autora me reservo los derechos morales de la obra antes citada. En concordancia suscribo en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Leyton Portilla Johanna Lisseth

040161773-3

Ibarra, Noviembre de 2016



**UNIVERSIDAD TÉCNICA DEL NORTE**

**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**

**CERTIFICACIÓN**

Ing. CARLOS VÁSQUEZ, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN

**CERTIFICA**

Que, el presente Trabajo de Titulación “DISEÑO DE UNA RED DEFINIDA POR SOFTWARE EMPLEANDO UNA SOLUCIÓN BASADA EN SOFTWARE, PARA LA INFRAESTRUCTURA DE CLOUD DE LA FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS (FICA).” Ha sido desarrollado por la señorita Leyton Portilla Johanna Lisseth bajo mi supervisión.

Es todo en cuanto puedo certificar en honor a la verdad.

Ing. Carlos Vásquez

**DIRECTOR**



## UNIVERSIDAD TÉCNICA DEL NORTE

### FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

#### AGRADECIMIENTO

Agradezco a Dios por permitirme culminar una etapa en mi vida y por todas las bendiciones recibidas.

A mis padres por todo su esfuerzo, su comprensión, amor, preocupación, atención y palabras de aliento, sobre todo porque nunca deje de sentir su apoyo incondicional, para enfrentar día a día cada reto, por ser mi inspiración y ejemplo en los momentos difíciles, también a mis dos hermanas por estar a mi lado y brindarme su cariño y amor.

A la Universidad Técnica del Norte y mis maestros, por todas las enseñanzas compartidas, de manera especial al Ingeniero Carlos Vásquez, tutor del proyecto, por todo el tiempo dedicado, la y la orientación recibida.

Agradezco a mis amigos, compañeros y cómplices, que a lo largo de los años me han acompañado y compartido conmigo momentos únicos.



**UNIVERSIDAD TÉCNICA DEL NORTE**

**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**

## DEDICATORIA

Con mucho amor a mis padres Carmita y Jorge, quienes cada momento creyeron en mí, a mis hermanas Dayana y Alexa; su amor y calor familiar son el motor de mi vida.

## **RESUMEN**

El presente proyecto tiene como finalidad presentar el diseño de una Red Definida por Software (SDN), como una solución a las limitaciones de las redes actuales y las tendencias de tráfico, realizando un estudio referente a la arquitectura de las redes SDN tomando en cuenta la separación de los planos de datos y de control, además de la investigación del protocolo OpenFlow en el cual están basadas dichas redes.

En la infraestructura de las redes SDN se presenta un controlador, para lo cual se toma cuatro alternativas: NOX, POX, Beacon y Floodlight, con los cuales se realizarán pruebas para determinar el más óptimo y adecuado en la implementación de la red, otro elemento de la infraestructura es un switch virtual, la adaptación del mismo es por medio del software Open vSwitch, al mismo tiempo se definen las reglas de flujo.

En la implementación de la red se utilizara el software KVM y virtual manager, el cual permite crear máquinas virtuales, además de que toda la implementación se realizara sobre el sistema Operativo Ubuntu, finalmente se desarrolla una aplicación para el control de acceso al medio (NAC).



## **ABSTRACT**

This project present the design of a Network Software Defined (SDN) as a solution to the limitations of current networks and traffic trends, conducting a study concerning the architecture of SDN networks, taking into account the separation of the control and data planes, in addition to research OpenFlow protocol , the networks are based in that protocol.

In the SDN network infrastructure a controller is presented, for that four alternatives is taken: NOX, POX, Beacon and Floodlight with which one tests will be performed to determine the most optimal and appropriate in the implementation of the network, another infrastructure element is a virtual switch, the adaption of it is through the Open vSwitch software while flow rules are defined.

In the implementation of the network the KVM software and virtual manage are used, which lets create virtual machines, in addition to all the implementation will take place on the Ubuntu operating system, finally an application for media access control is developed (NAC )

# Contenido

|   |                               |
|---|-------------------------------|
| AUTORIZACIÓN DE USO Y PUBLICACIÓN .....                         | ii                            |
| CERTIFICACIÓN.....  | v                             |
| CONSTANCIAS .....   | ¡Error! Marcador no definido. |
| DEDICATORIA.....  | vii                           |
| RESUMEN.....  | viii                          |
| ABSTRACT.....   | ix                            |
| GLOSARIO.....   | xv                            |
| CAPÍTULO I.....   | 1                             |
| ANTECEDENTES.....   | 1                             |
| 1.1. TEMA .....   | 1                             |
| 1.2. PLANTEAMIENTO DEL PROBLEMA .....                           | 1                             |
| 1.3. OBJETIVOS .....  | 2                             |
| 1.3.1. Objetivo general .....                                   | 2                             |
| 1.3.2. Objetivos específicos.....                               | 2                             |
| 1.4. ALCANCE.....   | 3                             |
| 1.5. JUSTIFICACIÓN.....   | 4                             |
| CAPÍTULO II .....   | 5                             |
| FUNDAMENTO TEÓRICO.....   | 5                             |
| 2.1. INTRODUCCIÓN A LAS REDES DEFINIDAS POR SOFTWARE (SDN)..... | 5                             |
| 2.1.1. Redes: limitaciones con la tecnología actual .....       | 5                             |
| 2.1.2. Necesidad de una nueva arquitectura de red.....          | 6                             |
| 2.1.3. Solución con una Red Definida por Software .....         | 8                             |
| 2.2. ARQUITECTURA DE UNA RED DEFINIDA POR SOFTWARE.....         | 9                             |
| 2.2.1. Introducción.....  | 9                             |
| 2.2.2. Generalidades .....                                      | 10                            |
| 2.2.3. Componentes de la arquitectura .....                     | 10                            |
| 2.2.4. Stack SDN .....  | 12                            |
| 2.3. PROTOCOLO OPENFLOW .....                                   | 12                            |
| 2.3.1. Introducción.....  | 12                            |
| 2.3.2. Switch OpenFlow y componentes .....                      | 13                            |
| 2.3.3. Clasificación de Switches OpenFlow .....                 | 16                            |

|  |           |
|--|-----------|
| <b>2.4. SERVIDOR CONTROLADOR .....</b>   | <b>18</b> |
| <b>2.4.1.Descripción de controladores .....</b>  | <b>18</b> |
| <b>2.4.2.NOX.....</b>  | <b>19</b> |
| <b>2.4.3.Funcionamiento, componentes y eventos de NOX .....</b>  | <b>19</b> |
| <b>2.4.4.POX.....</b>  | <b>22</b> |
| <b>2.4.5. Componentes, objetos, eventos y acciones de POX .....</b>                                      | <b>22</b> |
| <b>2.4.6.BEACON y su funcionamiento .....</b>  | <b>24</b> |
| <b>2.4.7.Aplicaciones de BEACON .....</b>  | <b>26</b> |
| <b>2.4.8.FLOODLIGTH.....</b>   | <b>27</b> |
| <b>2.4.9.Módulos FLOODLIGHT.....</b>   | <b>27</b> |
| <b>2.4.10. La aplicación de Floodlight .....</b>   | <b>28</b> |
| <b>2.4.11. Topologías soportadas por FLOODLIGTH.....</b>   | <b>29</b> |
| <b>2.5. APLICACIONES PARA EL CONTROL EN SDN .....</b>  | <b>30</b> |
| <b>2.5.1.NAC (Control de acceso a la red).....</b>   | <b>30</b> |
| <b>2.5.2.Enrutamiento .....</b>  | <b>31</b> |
| <b>2.5.3.Seguridad de la red.....</b>  | <b>33</b> |
| <b>2.6. VIRTUALIZACIÓN DE LA RED.....</b>  | <b>34</b> |
| <b>2.6.1.Introducción.....</b>   | <b>34</b> |
| <b>2.6.2.Virtualización de las funciones de red (NFV) .....</b>  | <b>35</b> |
| <b>2.6.3.Componentes lógicos de la red .....</b>   | <b>35</b> |
| <b>2.7. SOFTWARE UTILIZADO EN LA VIRTUALIZACIÓN .....</b>  | <b>36</b> |
| <b>2.7.1.Open vSwitch .....</b>  | <b>36</b> |
| <b>2.7.2.Componentes de Open vSwitch.....</b>  | <b>37</b> |
| <b>CAPÍTULO III.....</b>   | <b>39</b> |
| <b>IMPLEMENTACIÓN DE LA RED SDN Y DESARROLLO DE LA APLICACIÓN PARA EL CONTROL DE ACCESO (NAC). .....</b> | <b>39</b> |
| <b>3.1. CONTROLADORES E IMPLEMENTACIÓN DE SWITCH BASADO EN SOFTWARE .....</b>                            | <b>43</b> |
| <b>3.1.1. Plataforma de hardware .....</b>   | <b>44</b> |
| <b>3.1.2. Configuración de la red SDN.....</b>   | <b>44</b> |
| <b>3.1.3. Evaluación de los controladores.....</b>   | <b>55</b> |
| <b>3.2. ELECCIÓN DE SOFTWARE BASÁNDOSE EN LA NORMA ISO/IEC/IEEE 29148:201 .....</b>                      | <b>56</b> |

|  |            |
|--|------------|
| 3.2.1. Switch virtual .....                            | 57         |
| 3.2.2. Servidor controlador .....                      | 57         |
| <b>3.3. DEFINICIÓN DE REGLAS DE CONTROL.....</b>       | <b>62</b>  |
| 3.3.1. Estadísticas de flujo .....                     | 63         |
| 3.3.2. Información de la topología .....               | 64         |
| 3.3.3. Características del switch.....                 | 64         |
| 3.3.4. Flujos estáticos e ingreso de los mismos.....   | 64         |
| 3.3.5. Eliminación de flujos estáticos.....            | 66         |
| <b>3.4. DESARROLLO DE LA APLICACIÓN PARA NAC .....</b> | <b>68</b>  |
| 3.4.1. Introducción.....                               | 68         |
| 3.4.2. Diseño del software .....                       | 69         |
| 3.4.3. Componentes del software y del hardware .....   | 70         |
| 3.4.4. Diagrama de flujo.....                          | 71         |
| 3.5.5. Módulos Floodlight empleados.....               | 73         |
| 3.5.6. Código desarrollado .....                       | 74         |
| <b>CAPÍTULO IV .....</b>                               | <b>84</b>  |
| <b>PRUEBAS Y RESULTADOS .....</b>                      | <b>84</b>  |
| <b>4.1. CONFIGURACIÓN FINAL DE SWITCH .....</b>        | <b>84</b>  |
| <b>4.2. TOPOLOGÍA .....</b>                            | <b>86</b>  |
| <b>4.3. SWITCHES Y HOST.....</b>                       | <b>86</b>  |
| <b>4.4. PING ENTRE DISPOSITIVOS DE RED.....</b>        | <b>88</b>  |
| <b>4.5. TRAFICO EN WIRESHARK .....</b>                 | <b>88</b>  |
| <b>4.6. MENSAJES ICMP.....</b>                         | <b>90</b>  |
| <b>4.7. INSERCIÓN DE REGLAS .....</b>                  | <b>91</b>  |
| <b>4.8. CONECTIVIDAD .....</b>                         | <b>92</b>  |
| <b>CAPÍTULO V.....</b>                                 | <b>93</b>  |
| <b>CONCLUSIONES Y RECOMENDACIONES.....</b>             | <b>93</b>  |
| <b>5.1. CONCLUSIONES.....</b>                          | <b>93</b>  |
| <b>5.2. RECOMENDACIONES.....</b>                       | <b>96</b>  |
| <b>BIBLIOGRAFÍA.....</b>                               | <b>98</b>  |
| <b>ANEXOS .....</b>                                    | <b>101</b> |

## Índice de figuras

|  |    |
|--|----|
| <b>Figura 1.</b> Arquitectura SDN.....   | 10 |
| <b>Figura 2.</b> Componentes de switch OpenFlow .....                          | 14 |
| <b>Figura 3.</b> Proceso de pipeline.....                                      | 15 |
| <b>Figura 4.</b> Red con Switches Habilitados para OpenFlow .....              | 17 |
| <b>Figura 5.</b> Relación controlador Floodlight, módulos y aplicaciones ..... | 29 |
| <b>Figura 6.</b> Red con Floodlight en zona OpenFlow .....                     | 29 |
| <b>Figura 7.</b> Red con Floodlight en varias zonas OpenFlow .....             | 30 |
| <b>Figura 8.</b> Infraestructura física de la red .....                        | 40 |
| <b>Figura 9.</b> Conexión desde el Switch al Servidor SDN.....                 | 42 |
| <b>Figura 10.</b> Infraestructura interna de la red.....                       | 43 |
| <b>Figura 11.</b> Diagrama para pruebas de Open vSwitch.....                   | 43 |
| <b>Figura 12.</b> Resultado ifconfig .....                                     | 46 |
| <b>Figura 13.</b> Resultado ovs-vsctl show .....                               | 47 |
| <b>Figura 14.</b> Resultado de ps -ea   grep ovs .....                         | 47 |
| <b>Figura 15.</b> Resultado de /etc/init.d/openvswitch-switch restart .....    | 47 |
| <b>Figura 16.</b> Asignación de Datapath NOX .....                             | 48 |
| <b>Figura 17.</b> Conexión entre NOX y switch .....                            | 48 |
| <b>Figura 18.</b> Comunicación entre el switch y NOX.....                      | 49 |
| <b>Figura 19.</b> Mensajes generados por NOX.....                              | 49 |
| <b>Figura 20.</b> Asignación de Datapath POX.....                              | 50 |
| <b>Figura 21.</b> Conexión entre POX y Switch .....                            | 50 |
| <b>Figura 22.</b> Comunicación entre el switch y POX.....                      | 50 |
| <b>Figura 23.</b> Mensajes generados por POX .....                             | 51 |
| <b>Figura 24.</b> Asignación del Datapath Beacon.....                          | 51 |
| <b>Figura 25.</b> Conexión Beacon y Switch .....                               | 51 |
| <b>Figura 26.</b> Comunicación entre el switch y Beacon.....                   | 52 |
| <b>Figura 27.</b> Mensajes generados por Beacon .....                          | 52 |
| <b>Figura 28.</b> Interfaz gráfica de Beacon.....                              | 53 |
| <b>Figura 29.</b> Resultado de java -jar target/floodlight.jar .....           | 53 |
| <b>Figura 30.</b> Asignación del Datapath Floodlight .....                     | 54 |
| <b>Figura 31.</b> Comunicación entre el Switch y Floodlight .....              | 54 |
| <b>Figura 32.</b> Mensajes LLDP.....   | 55 |
| <b>Figura 33.</b> Interfaz Web Floodlight .....                                | 55 |
| <b>Figura 34.</b> Ingreso al Archivo floodlightdefault.properties .....        | 63 |
| <b>Figura 35.</b> Modificación de archivo floodlightdefault.properties .....   | 63 |
| <b>Figura 36.</b> Comando para ver las características de los flujos .....     | 64 |
| <b>Figura 37.</b> Comando para ver los switches conctados a la red .....       | 64 |
| <b>Figura 38.</b> Comando para verlas características del switch .....         | 64 |
| <b>Figura 39.</b> Inserción de flujo para trafico ARP.....                     | 67 |
| <b>Figura 40.</b> Ingreso de flujo cliente1 .....                              | 67 |
| <b>Figura 41.</b> Ingreso flujo cliente2.....                                  | 67 |
| <b>Figura 42.</b> Aplicación Northbound y Southbound .....                     | 68 |
| <b>Figura 43.</b> Infraestructura de red.....                                  | 70 |

|   |    |
|---|----|
| <b>Figura 44.</b> Diagrama de flujo aplicación cliente .....      | 71 |
| <b>Figura 45.</b> Diagrama de flujo aplicación ServidorNAC .....  | 72 |
| <b>Figura 46.</b> Código de la clase Hilo.....                    | 76 |
| <b>Figura 47.</b> Código clase Escuchar.....                      | 76 |
| <b>Figura 48.</b> Código clase principal ServidorNAC.....         | 77 |
| <b>Figura 49.</b> Código método loadSwitchData .....              | 77 |
| <b>Figura 50.</b> Código método Filtro .....                      | 78 |
| <b>Figura 51.</b> Código método loadDeviceData.....               | 79 |
| <b>Figura 52.</b> Continuación código método loadDeviceDat.....   | 80 |
| <b>Figura 53.</b> Código método conexionCliente .....             | 81 |
| <b>Figura 54.</b> Continuación código método conexionCliente..... | 82 |
| <b>Figura 55.</b> Continuación código método conexionCliente..... | 82 |
| <b>Figura 56.</b> Código Aplicación Cliente .....                 | 83 |
| <b>Figura 57.</b> Resultado de ovs-vsctl show .....               | 84 |
| <b>Figura 58.</b> Resultado de ps -ea   grep ovs .....            | 85 |
| <b>Figura 59.</b> Resultado de ovs-ofctl show ovsbr0.....         | 85 |
| <b>Figura 60.</b> Resultado de route -n.....                      | 86 |
| <b>Figura 61.</b> Topología de la red.....                        | 86 |
| <b>Figura 62.</b> Switch virtual de la red.....                   | 87 |
| <b>Figura 63.</b> Host de la red.....                             | 87 |
| <b>Figura 64.</b> Resultado ping Cliente1-Cliente2.....           | 88 |
| <b>Figura 65.</b> Resultado ping Cliente2-Cliente1.....           | 88 |
| <b>Figura 66.</b> Paquetes TCP, ARP, ICMP Y OSPF.....             | 89 |
| <b>Figura 67.</b> Segmento TCP.....                               | 89 |
| <b>Figura 68.</b> Conexión OpenFlow.....                          | 89 |
| <b>Figura 69.</b> Conexión OpenFlow.....                          | 89 |
| <b>Figura 70.</b> Tráfico ARP .....                               | 90 |
| <b>Figura 71.</b> Paquetes ICMP .....                             | 90 |
| <b>Figura 72.</b> Paquetes ICMP .....                             | 90 |
| <b>Figura 73.</b> Mensajes ICMP .....                             | 91 |
| <b>Figura 74.</b> Reglas insertadas en el Switch .....            | 91 |
| <b>Figura 75.</b> Verificación de reglas en el explorador.....    | 92 |
| <b>Figura 76.</b> Reglas iniciales.....                           | 92 |
| <b>Figura 77.</b> Reglas después de desconexión.....              | 92 |

## Índice de tablas

|   |    |
|---|----|
| <b>Tabla 1.</b> Características servidor SDN .....                    | 44 |
| <b>Tabla 2.</b> Direcciones IP de servidor y máquinas virtuales ..... | 45 |
| <b>Tabla 3.</b> Requerimientos Funcionales .....                      | 60 |
| <b>Tabla 4.</b> Requerimientos no Funcionales .....                   | 61 |
| <b>Tabla 5.</b> Propiedades tabla de flujo.....                       | 65 |
| <b>Tabla 6.</b> Opciones campo acción .....                           | 65 |
| <b>Tabla 7.</b> Argumentos de flujos estáticos .....                  | 66 |

## GLOSARIO

- **SDN (Red Definida por Software):** son una manera de abordar la creación de redes en la cual el control se desprende del hardware y se le da a una aplicación de software llamada controlador.
- **NAC (Control de Acceso a la Red):** es un enfoque de la seguridad en redes de computadoras que intenta unificar la tecnología de seguridad en los equipos finales, usuario o sistema de autenticación y reforzar la seguridad de la red de acceso.
- **UDP (Protocolo de Datagrama de Usuario):** un protocolo sin conexión que, funciona en redes IP, proporciona pocos servicios de recuperación de errores, ofreciendo en su lugar una manera directa de enviar y recibir datagramas a través una red IP.
- **IT (Tecnología Informática):** Es un término que se refiere a hardware, software, telecomunicaciones, redes y personas involucradas para crear, almacenar, intercambiar y utilizar la información.
- **API (Application Programming Interface):** es un conjunto de reglas y especificaciones que las aplicaciones siguen para comunicarse; sirviendo de interfaz entre programas diferentes al igual que la interfaz de usuario facilita la interacción humano-software
- **TCP (Protocolo de Control de Transmisión):** es uno de los principales protocolos de la capa de transporte del modelo TCP/IP. En el nivel de aplicación, posibilita la administración de datos que vienen del nivel más bajo del modelo, o van hacia él.
- **Python:** es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

- **JSON (Notación de Objetos de JavaScript):** es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.
- **HTTP (Hypertext Transfer Protocol):** protocolo de transferencia de hipertextos', que se utiliza en algunas direcciones de internet.
- **ARP (Address Resolution Protocol):** Es un protocolo de nivel de red responsable de encontrar la dirección hardware (MAC Address) que corresponde a una determinada dirección IP.
- **LLDP (Protocolo de Descubrimiento de la Capa de Enlace):** ha sido diseñado para dispositivos de redes Ethernet (como switches y routers)
- **ICMP (Protocolo de Mensajes de Control de Internet):** es un protocolo que permite administrar información relacionada con errores de los equipos en red.
- **DHCP (Dynamic Host Configuration Protocol):** es un servidor que usa protocolo de red de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas
- **MIB (Management Information Base):** s una base de datos estándar formada por diferentes variables SNMP, las cuales se definen en un idioma independiente del sistema destino
- **ACL (Lista de Control de Acceso):** es un concepto de seguridad informática usado para fomentar la separación de privilegios.



# **CAPÍTULO I**

## **ANTECEDENTES**

### **1.1.TEMA**

Diseño de una red definida por software empleando una solución basada en software, para la infraestructura de Cloud de la facultad de Ingeniería en Ciencias Aplicadas (FICA)

### **1.2.PLANTEAMIENTO DEL PROBLEMA**

En la actualidad la mayor fuente de información se la encuentra en la red, pero para que la red funcione de la mejor forma existe protocolos, que se encargan de definir las normas para la interconexión de la misma red, estos protocolos son limitados y muy poco flexibles al momento de realizar cambios, con el desarrollo de la tecnología se ha creado dispositivos de red que tienen la capacidad de reemplazar a los actualmente utilizados en la red, pero el cambio de dichos dispositivos significaría una inversión demasiado costosa. (CISCO, 2015)

La Universidad Técnica del Norte cuenta con distintas Facultades, entre ellas la Facultad de Ciencias Aplicadas (FICA ), en la cual el acceso a Internet se lo realiza a través de medios cableados y también existen diferentes puntos de acceso inalámbricos utilizados por los estudiantes, profesores y demás personas en particular, pero los dispositivos que se conectan a través de medios inalámbricos no poseen la suficiente inteligencia para mantenerse conectados continuamente a más de una red inalámbrica, limitando de alguna forma el acceso al internet en toda la Facultad.

El acceso a Internet posee diferentes restricciones y limitaciones que se deben a que existe una gran complejidad en los centros de datos donde se almacena la información, además de que las políticas que rigen la red son inconsistentes dentro de los dispositivos de la misma, dando

paso a que el crecimiento dinámico y la adaptación a las demandas de los usuarios presente dificultades. (Gandarillas, 2013)

“ En el presente proyecto se plantea la utilización de una Red Definida por Software (SDN), esta red es una arquitectura que separa el plano de control del plano de datos, permite la flexibilidad de configurar, implementar y administrar protocolos personalizados para cumplir con los requerimientos de la red, monitorear y enrutar tráfico UDP (streaming).” (Jimenez, 2013)

## **1.3.OBJETIVOS**

### **1.3.1. Objetivo general**

Diseñar una Red Definida por Software (SDN), por medio de la utilización del Protocolo OpenFlow para mejorar las conexiones a la red en la Facultad de Ingeniería en Ciencias Aplicadas (FICA)

### **1.3.2. Objetivos específicos**

- Elaborar una fundamentación teórica acerca de las Redes Definidas por Software (SDN), su arquitectura y protocolos de implementación.
- Implementar la red SDN mediante la utilización de switches basados en software.
- Desarrollar la aplicación que permita el control de acceso a la red.
- Realizar las pruebas del correcto funcionamiento de la red implementada.

## **1.4.ALCANCE**

El presente proyecto tiene como finalidad la implementación de una red SDN en la infraestructura de Cloud que se encontrara ubicada en la Facultad de Ingeniería en Ciencias Aplicadas (FICA), mejorando las conexiones a la red de la misma.

Para dar inicio al proyecto se considera necesario realizar una fundamentación teórica de todo lo que se refiere a SDN, en el fundamento teórico se presenta una solución basada en SDN y las limitaciones que las tecnologías de red poseen actualmente, tomando en cuenta las tendencias de trafico de los datos de una red, se resume un estudio de la arquitectura de la red SDN, del protocolo OpenFlow y del software que se requiere para el servidor controlador, luego se describe las aplicaciones que se encuentran en el plano de control además de estudiar acerca de la virtualización de los componentes de una red, la cual se enfoca en equipos de conmutación.

En la implementación de la red SDN se considera las características para el software del servidor controlador, tomando en cuenta los escenarios de switch virtual Open, luego se definen las reglas del flujo de información, determinando las aplicaciones que se van a soportar en el plano de control y que permitirá el desarrollo de la aplicación para el control del acceso a la red (NAC), se realizará el diseño de la misma y el código que se utiliza para su ejecución.

Una vez instalada y configurada la red SDN se realizaran las pruebas del correcto funcionamiento de la red, de la programación y ejecución de la misma, además de efectuar las pruebas para el control de acceso hacia la red, cave recalcar que para estas pruebas se utilizará los servidores disponibles en la infraestructura de cloud de la Facultad de Ingeniería en Ciencias Aplicadas.

Finalmente se recopilaran conclusiones y recomendaciones del trabajo realizado.

## 1.5.JUSTIFICACIÓN

En la actualidad la información y los datos que se encuentran en la red de redes que es el Internet, crece de forma acelerada, debido a que la necesidad de comunicar a todo el mundo cada vez es mayor y exige la prestación de más servicios.

La comunicación entre el cliente y el servidor a través de una red se realiza de forma vertical, de manera que se impide realizar cambios en las redes, los datos confidenciales de los usuarios y la información están susceptibles a que se expongan de forma insegura. El crecimiento de los servicios en la nube da paso a la necesidad de servicios y a la flexibilidad de los mismos por parte de los proveedores de dichos servicios.

Con la adaptación de la nube para recopilar y almacenar información, se ve la necesidad de cambiar el punto de vista de la red, posibilitando brindar servicios mejorados en la nube, escalabilidad del ancho de banda y la adaptación de forma mejorada de los distintos dispositivos de red. (Gandarillas, 2013)

Con la creación de una red SDN, manejada por el protocolo OpenFlow es posible brindar la centralización de otras redes con distintas plataformas y el control de las mismas, permitiendo automatizar los procesos de la red, mejorar la innovación en la programación de nuevos servicios y sobre todo aumentando la seguridad y confiabilidad de la red, brindando también la facilidad de controlar la red y su infraestructura (MorilloFuentala, 2014)

# **CAPÍTULO II**

## **FUNDAMENTO TEÓRICO**

El capítulo describe el fundamento teórico para la realización del proyecto, como primera instancia se realiza un descripción breve de las limitaciones que poseen las redes en la actualidad, se da paso a una solución por medio de las Redes Definidas por Software, la arquitectura de dichas redes y el protocolo OpenFlow que es el que se utiliza, además del plano de control y el software para el servidor controlador, la virtualización de los componentes.

### **2.1.INTRODUCCIÓN A LAS REDES DEFINIDAS POR SOFTWARE (SDN)**

Las empresas y usuarios de una red en la actualidad presentan muchos problemas en sus redes tradicionales, puesto que no cumplen con los requerimientos solicitados, por lo que las Redes Definidas por Software (SDN), se presentan como una solución estandarizada para cubrir las necesidad de las empresas y los usuarios de red, dando paso a la transformación de la arquitectura de las redes tradicionales.

#### **2.1.1. Redes: limitaciones con la tecnología actual**

Utilizar las redes tradicionales para satisfacer los requerimientos de telecomunicaciones, generados por los usuarios resulta improbable; las empresas a nivel mundial utilizan los departamentos de IT para aprovechar al máximo sus redes, como una solución pasajera, debido a que las redes existentes no están diseñadas para cubrir los requerimientos actuales de los usuarios, compañías y proveedores. (Velez, 2012)

Las limitaciones de las redes en la actualidad son:

- **Complejidad:** las necesidades de los usuarios son cada vez más exigentes y para cubrir las los protocolos de red se han mejorado, haciéndolos más seguros y con mayor eficacia, los protocolos de red se designan para resolver un problema específico sin favorecerse de una acción conjunta.
- **Políticas inconsistentes:** para brindar mayor seguridad a una red se implementan políticas, las cuales se configuran a través de mecanismos y distintos dispositivos, que dejan vulnerable a la red ante ataques y violaciones de seguridad, además de configuraciones equívocas y otras consecuencias perjudiciales.
- **Imposibilidad de escalabilidad:** la red debe crecer al mismo tiempo que las demandas de los usuarios, no obstante la red se vuelve más compleja debido a la suma de miles de dispositivos con sus propias características de configuración y gestión.
- **Dependencia del vendedor:** los servicios y capacidades desarrollados para satisfacer las demandas de los usuarios, se ven limitadas por el periodo de producción de los equipos por parte de los fabricantes y la poca existencia de interfaces generalizadas que obstaculizan el ajuste de las redes a entornos específicos. (opennetworking, 2013)

Las redes convencionales no poseen la infraestructura para cumplir con los requerimientos de los usuarios que día a día crecen con rapidez, y se debe a las limitaciones de seguridad, complejidad de la red y el hecho de depender equipos que no son fabricados al mismo tiempo que el desarrollo de servicios.

### 2.1.2. Necesidad de una nueva arquitectura de red

Las redes tradicionales existentes no se adaptan a las necesidades de las redes presentes, debido al gran aumento de dispositivos, servicios de Cloud, contenido audiovisual y a los servicios

requeridos por los usuarios de la red, llevando a la industria de comunicaciones a pensar el desarrollo de nuevas arquitecturas de red, que cumplan con los requerimientos de comunicaciones actuales.

Algunos de los factores que se toman para pensar en nuevas arquitecturas de red son:

- **Cambio en patrón de tráfico:** los servidores crean una comunicación entre máquinas antes de conectarse al cliente final, por lo que se crea un patrón (norte-sur), del mismo modo los usuarios ingresan a las aplicaciones de las empresas por medio de una gran variedad de dispositivos, en cualquier momento y desde cualquier lugar; dando paso a tráfico adicional en una red.
- **Uso de IT:** la utilización de distintos dispositivos para acceder a los servicios de la red de una empresa. es cada vez más exhaustiva, los departamentos de IT se ven en la necesidad de controlar el acceso de dispositivos a la red, brindando seguridad, disponibilidad y confiabilidad a la empresa. (Spera, 2013)
- **Servicios de Cloud:** la disponibilidad de los servicios en la nube es adoptada por muchas empresas, por medio de la creación de nubes privadas y públicas, para que exista acceso a las aplicaciones, infraestructura y recursos de IT de dichas empresas de forma remota, por tal razón los departamentos de IT necesitan brindar seguridad y cumplir con los requerimientos de las organizaciones. (informaticahoy, 2013)
- **Big Data:** la manipulación de grandes conjuntos de datos requiere que varios servidores conectados entre sí, procesen dicha información al mismo tiempo, dando paso a una considerable demanda de los recursos de red y de la escalabilidad de las redes para soportar mayor tráfico y conectividad. (Dans, 2011)

### 2.1.3. Solución con una Red Definida por Software

Con la arquitectura de red SDN se pueden crear redes que cubran los requerimientos de los usuarios y empresas en la actualidad, debido a que permite la separación del plano de control y el plano de datos, para que las personas encargadas de la administración de la red, lo realicen de forma centralizada. La manipulación del plano de control y del plano de datos se realiza por separado, para el plano de control se utiliza el protocolo OpenFlow, por medio de equipos controladores y el plano de datos es operado por los equipos de conexión.

OpenFlow es un protocolo relativamente nuevo, el cual permite a un servidor establecer el camino por el cual se deberían enviar los paquetes en una red de switches, la utilización de OpenFlow da paso a que una red se trate como un solo individuo y no por separado cada uno de sus dispositivos, el servidor controlador contiene la información de las reglas e instrucciones para establecer el enrutamiento de los paquetes, dichas reglas son aplicadas a un grupo de paquetes especificando lo que se produce en el tráfico de red. (opennetworking, 2015)

Debido a que SDN es una arquitectura con características que la hacen dócil, eficaz, dinámica, beneficiosa y flexible, se la considera óptima para cumplir con los requerimientos de las redes y aplicaciones actuales, el manejo individual del plano de control permite que los equipos puedan ser tratados como objetos lógicos o virtuales, las redes SDN están siendo adoptadas por muchas empresas y por una gran variedad de dispositivos. (citrix, 2013)

Los beneficios que SDN ofrece son:

- **Programable directamente:** el control está separado de las funciones de reenvío de datos.
- **Ágil:** los administradores de red pueden manipular el tráfico de la red para cumplir con los requerimientos y necesidades actuales.



- **Centralizada:** el manejo de dispositivos se hace como un solo ente para mantener una visión global de la red.
- **Configuración mediante programación:** los programas de configuración no poseen propietarios por lo que se las redes SDN pueden ser administradas y configuradas brindando seguridad y recursos mejorados.
- **Estándares abiertos y neutrales:** las instrucciones de enrutamiento son dadas por controladores SDN sin la utilización de protocolos y equipos determinados de los proveedores. (Jimenez, 2013)

SDN se presenta como una solución que cubre con las necesidades y los requerimientos de las redes actuales, que las redes tradicionales no cumplen, brindando una plataforma de servicios que pueda responder a los cambios y requisitos del mercado, a partir de seguridad y dinámica de control de los dispositivos que conforman una red.

## **2.2.ARQUITECTURA DE UNA RED DEFINIDA POR SOFTWARE**

### **2.2.1. Introducción**

Los dispositivos de conectividad en una red se encuentran integrados por dos componentes importantes: el plano de control y el plano de datos, el plano de datos es el encargado de la transmisión de los datos y el plano de control determina el camino por el cual la información debe circular para llegar a su destino.

Debido al crecimiento de las necesidades y requerimientos de las redes, se pensó en separar los planos de control y de datos para aportar servicios y brindar mayor seguridad, dando paso a una interfaz abierta entre los planos y un control lógico de la red. (cmqueue, 2013)

Una Red Definida por Software (SDN) aparece como una solución para las demandas y requerimientos tanto del plano de control como del plano de datos. (MorilloFuentala, 2014)

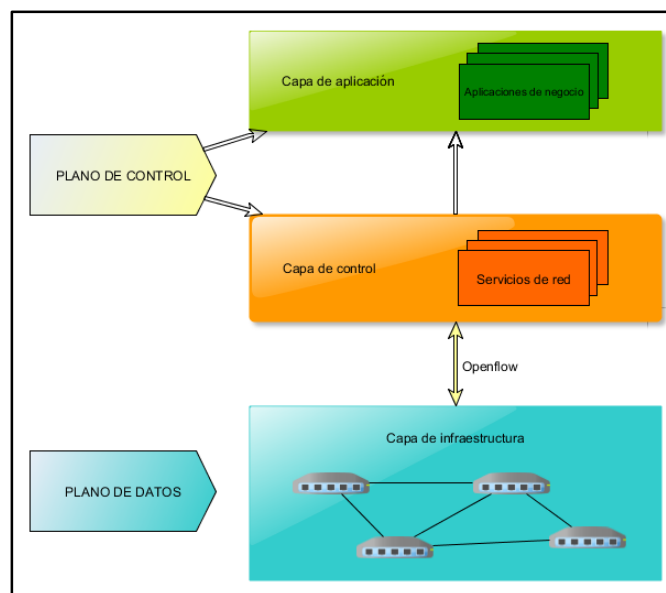
### 2.2.2. Generalidades

En una red tradicional, si un paquete de datos llega a un switch este se encarga de direccionar el paquete hacia su destino final siguiendo una ruta determinada y repite el proceso con todos los paquetes que lleguen al mismo, en una red SDN es posible configurar las reglas de tráfico sin la necesidad de configurar cada uno de los switches que conformen la red, facilitando la gestión del tráfico de la red de forma flexible y eficiente.

### 2.2.3. Componentes de la arquitectura

“Red Definida por Software (SDN) es una arquitectura de red cuyo dinamismo, manejabilidad, rentabilidad y adaptabilidad permiten que sea adecuada para la naturaleza dinámica y de alto consumo de ancho de banda de las aplicaciones modernas”. (opennetworking, 2015)

En la Figura 1 se representa la arquitectura de una Red Definida por Software, en la cual se pueden definir 3 componentes principales.



**Figura 1.** Arquitectura SDN

**Fuente:** SDxCentral. (2012). What's Software Defined Networking (SDN)? Definition. Obtenido de: <https://goo.gl/i94XIG>

#### 2.2.3.1. Capa de infraestructura

Se encuentra conformada por los elementos de la red que se encargan de la conmutación y el enrutamiento de los paquetes de datos, ofreciendo acceso a través de una API, como es OpenFlow.

#### 2.2.3.2. Capa de control

La función de control es centralizada, permitiendo la utilización de las capacidades de la red, el controlador de una SDN es un ente, el cual tiene el control de un conjunto de recursos del plano de control, permitiendo el correcto encaminamiento del tráfico de red, el controlador se encarga de la conmutación de datos y de elegir el camino óptimo para la transmisión de los mismos, el controlador tiene la capacidad de gestionar un rango de recursos del plano de datos, facilitando la configuración del mismo. (Millan, 2015)

#### 2.2.3.3. Capa de aplicación

Consta de las aplicaciones solicitadas por los usuarios finales, las solicitudes se realizan por medio de las API de SDN las cuales permiten la utilización de servicios de comunicación, la capa de aplicación logra que las configuraciones resulten simples y da paso a la gestión de nuevos servicios, formas de acceso y mejora de la red. (cmqueue, 2013)

#### 2.2.3.4. OpenFlow

OpenFlow es una interfaz utilizada en las redes SDN la cual permite la comunicación entre los planos de datos y de control de los equipos que forman la red, ofrece la facilidad de acceder al plano de control y manipularlo de forma directa, ya sea en dispositivos físicos o virtuales. (Princeton, 2013)

#### 2.2.4. Stack SDN

El sus principios las redes SDN se desarrollaron basadas en el controlador NOX juntamente con el protocolo OpenFlow, los mensajes enviados por OpenFlow son considerados como eventos de NOX, el stack de SDN está formado por:

- **Abstracciones:** para la implementación y desarrollo del controlador se debe basar en abstracciones como:
  - Vista global de la red: es necesario un programa de control el cual genere una vista global de la red, donde la configuración de dicho programa es una función de la vista.
  - Infraestructura de reenvío: para la reducción de tráfico generado por las aplicaciones es necesario una estructura de control, la cual dé lugar a la comunicación extremo a extremo.
  - Especificación: es necesario generar un resumen de la vista global de la red, para dar a las aplicaciones la información mínima requerida para sus necesidades. (Stanford, 2012)
- **Distribución:** para que una red SDN tenga mayor escalabilidad, debe existir una distribución, para que le diseño de la red no sea complejo.
- **Depuración:** un depurador se encarga de la detección de configuraciones que no están acorde a las proporcionadas por el programa de control, permitiendo a los usuarios conocer como la red responde a diferentes eventos. (Stanford, 2012)

### 2.3.PROTOCOLO OPENFLOW

#### 2.3.1. Introducción

Las Redes Definidas por Software se encuentran basadas en el protocolo OpenFlow, el cual fue desarrollado a inicio del 2007 por la Universidad de Stanford y de California, actualmente la

ONF (Open Networking Foundation), es la encargada del desarrollo y mantenimiento del protocolo. (HP, 2014)

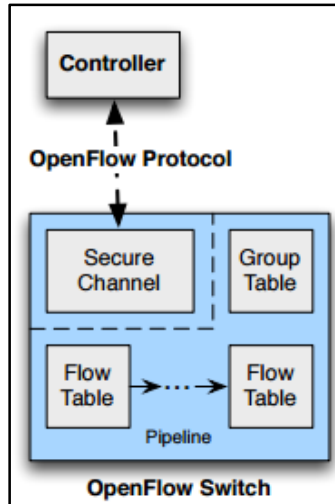
OpenFlow es un protocolo reciente en lo que se refiere a comunicaciones, el cual permite determinar el correcto enrutamiento de los paquetes en una red, utilizando OpenFlow una red puede ser manipulada como un solo ente, centralizando las decisiones de movilidad de paquetes, en OpenFlow un controlador se encarga del determinar el encaminamiento de nivel alto.

(opennetworking, 2015)

En una red SDN el protocolo OpenFlow debe estar operativo en los equipos que forman la red y en el controlador de la misma, para distinguir el tráfico de la red el protocolo utiliza flujos, que definen como el tráfico viaja a través de los dispositivos de red tomando en cuenta parámetros de uso y aplicaciones, basándose en reglas predeterminadas en el controlador.

### **2.3.2. Switch OpenFlow y componentes**

Un Switch OpenFlow es un dispositivo que envía paquetes de una red SDN el cual opera con la utilización de tablas de flujo y acciones de cada entrada del flujo, siendo posible dividir el tráfico de la red, el Switch OpenFlow permite la adopción de protocolos nuevos, modelos de seguridad y diseños de direccionamiento, el tráfico de la red no se ve perjudicado debido a que trata de forma aislada y se procesa de igual manera que en una red tradicional, se encuentra compuesto por 3 parámetros principales que se muestran en la Figura 2:



**Figura 2.** Componentes de switch OpenFlow

**Fuente:** OpenFlow. (2011). *OpenFlow Switch Specification*. Obtenido de: <http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf>

- **Tabla de flujos:**

Un Switch OpenFlow consta de varias tablas de flujo, las cuales tienen una acción asociada a cada entrada de flujo, mostrándole al Switch como debe manipular el flujo, esto lo hace mediante la utilización de un proceso llamado matching, en el cual se comparan las entradas de cada una de las tablas de flujo.

Los componentes de una entrada de flujo son:

- La cabecera, la cual define el flujo del paquete
- Acción, informa como deben ser procesados los paquetes
- Estadísticas, registran el número de paquetes, los bytes en cada flujo y tiempo desde que se aplicó una regla aun paquete

Los campos de una entrada de flujo:

- Campos coincidentes: consisten en el puerto de entrada y la cabecera, sirven para identificarlos paquetes.
- Prioridad: coincidencias con el flujo de la entrada.

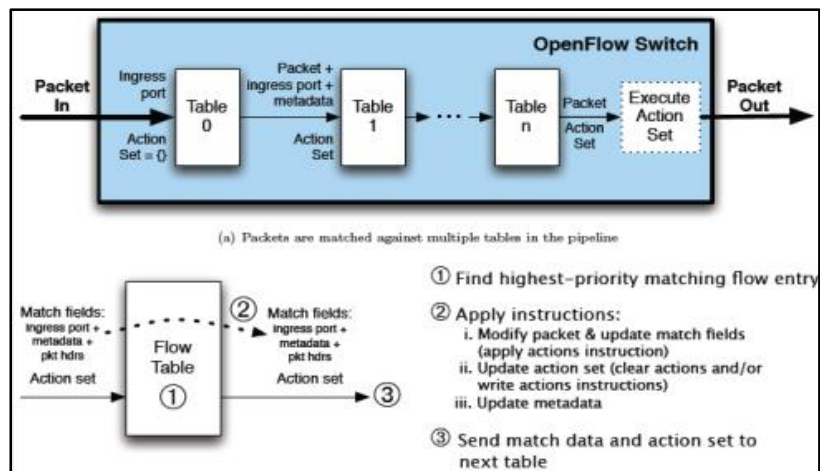
- Contadores: al encontrar una coincidencia se actualizan.
- Instrucciones: sirven para modificar las acciones
- Timeout: tiempo máximo que se considera para que el flujo sea descartado.
- Cookie: valor dado por el controlador para filtrar características de los flujos.

(Archive OpenFlow, 2011)

### Matching:

El proceso de matching es el proceso que se realiza para la búsqueda de coincidencias, el cual inicia cuando un Switch OpenFlow recibe un paquete y realiza una búsqueda en la primera tabla de flujo y si no encuentra coincidencias pasa a la siguiente tabla, al encontrar una coincidencia con alguna entrada se extraen los campos coincidentes del paquete, luego se realizan una serie de acciones sobre el paquete y se envía a su destino final, si no existe coincidencia alguna el paquete regresa al controlador o se descarta.

Un pipeline es una serie de elementos de un proceso ordenado, de forma que la salida de cada elemento es la entrada del siguiente, en la Figura 3 se muestra el proceso de pipeline:



**Figura 3.** Proceso de pipeline

**Fuente:** Vanessa Moreno. (2012). *OpenFlow*. Obtenido de: [http://www.dit.upm.es/~posgrado/doc/TFM/TFMs2011\\_012/TFM](http://www.dit.upm.es/~posgrado/doc/TFM/TFMs2011_012/TFM)

1. En un paquete de entrada su encabezado se compara con la información de la tabla de flujo y la coincidencia de mayor prioridad se elige.
2. Al encontrar coincidencia se aplican acciones:
  - Modificación del paquete
  - Actualización de campos que coinciden
  - Actualización de las acciones
  - Actualización de metadatos.
3. Se envía el dato y las acciones a la siguiente tabla (MorilloFuentala, 2014)

- **Canal seguro:**

Se llama canal seguro debido a que maneja el protocolo SSL, el cual da mayor seguridad cuando se envían los paquetes, el canal se encarga de conectar al Switch con el controlador, permitiéndole al controlador la configuración y la gestión de Switch de forma remota, también da paso a paquetes y comandos entre el Switch y el controlador por medio de la utilización del protocolo OpenFlow.

- **Controlador:**

Se encarga de insertar o quitar entradas en las tablas de flujo

### 2.3.3. Clasificación de Switches OpenFlow

Los switches OpenFlow se pueden clasificar en 3 grupos:

- **Switches dedicados:**

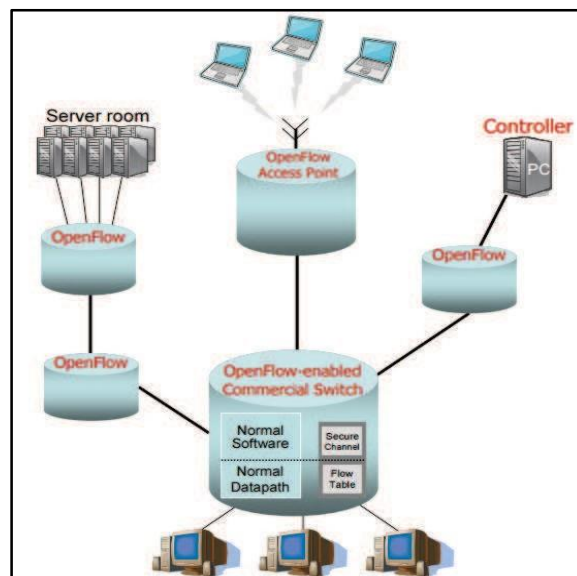
Son dispositivos simples, que soportan únicamente OpenFlow y no son aptos para el procesamiento en las capas 2 y 3, estos switches se encargan del envío de paquetes de acuerdo a lo establecido en el controlador, el flujo del tráfico está definido claramente y solo se lo limita a la información de las tablas de flujo.



Cada flujo tiene asociadas acciones, las principales son:

- Reenvió: se envía un flujo de datos aun puerto determinado
  - Encapsulamiento: el paquete es entregado al canal seguro, en el cual se lo encapsula y se lo envía hacia el controlador, este decide si se aumenta la entrada de flujo en la tabla de flujo.
  - Descarte: utilizado para dar seguridad, reducir el tráfico de broadcast, quitando los paquetes de denegación de servicios.
- **Switches habilitados para OpenFlow:**

Los actuales switches, routers y puntos de acceso del mercado se encuentran mejorando para que sea posible la adopción del protocolo OpenFlow, añadiendo a su configuración las tablas de flujo que pueden utilizar el hardware del Switch y el canal seguro, el protocolo se adaptará para que se ejecute por el switch, permitiendo que las tablas de flujo sean manipuladas por un mismo controlador y de esta forma aumentar el rendimiento de la red. En la Figura 4 los switches soportan el procesamiento de las capas 2 y 3 (Santamaría, 2012)



**Figura 4.** Red con Switches Habilitados para OpenFlow

**Fuente:** Sergio Rodríguez. (2012). *Mecanismos de control de las comunicaciones en la Internet del futuro a través de OpenFlow*. Obtenido de: <http://goo.gl/RACz39>

- **Switches tipo cero “0”:**

Estos switches son capaces de soportar el formato de encabezado de OpenFlow y las acciones básicas de reenvío, encapsulado y descarte.

## **2.4.SERVIDOR CONTROLADOR**

Un servidor controlador es el encargado de centralizar el control de los elementos que conforman una red, es capaz de obtener una visión general de la misma, debido a que posee información acerca del estado de los dispositivos, el comportamiento de flujo de tráfico es determinado por las aplicaciones ejecutadas en el controlador, el cual interactúa con los elementos de red por medio de la utilización del protocolo OpenFlow.

### **2.4.1. Descripción de controladores**

Existen varios software para el controlador de una red SDN, los que se han tomado en cuenta son:

- NOX/POX: utiliza lenguaje C++ y Python, ofrece una plataforma para la creación de aplicaciones OpenFlow.
- BEACON: utiliza el lenguaje Java, ofrece una interfaz extensible y personal.
- FLOODLIGHT: utiliza el lenguaje Java, ofrece un sistema de módulos, por lo cual es fácil reconfigurar, tiene mayor rendimiento y es extensible, puede ser utilizado en cualquier equipo de red. (MorilloFuentala, 2014)

## **2.4.2. NOX**

En OpenFlow, NOX es el controlador original, fue desarrollado por Nicira, NOX es básicamente una plataforma que permite crear aplicaciones para el control de la red, la versión más actual de NOX ofrece un framework más liviano y de mayor rendimiento, su desarrollo se hace en el lenguaje C++ para Linux.

Las características principales de NOX son:

- Ofrece una aplicación de alto nivel para OpenFlow
- Las reglas de flujo de tráfico son desarrolladas en C++
- Ofrece soporte OpenFlow
- Se desarrolló para las versiones de Linux: Ubuntu 12.04 y 11.10
- Permite realizar el descubrimiento de la topología de red
- Ubica todos los equipos detectados
- Aprender acerca de la conmutación
- Análisis de redes extendidas de conmutación

## **2.4.3. Funcionamiento, componentes y eventos de NOX**

### **2.4.3.1. Funcionamiento**

NOX funciona a base de tablas de flujo, el manejo de dichas tablas lo realiza por medio de una interfaz de programación en la cual se pueden configurar aplicaciones que proporcionen mayor control de acceso a la red ,dentro de los eventos de la misma, mediante la manipulación de los flujos de red se puede controlar las decisiones de reenvío y el tráfico de la red de una forma escalable; cuando se detecta una nueva entrada de flujo, dicho flujo es enviado al controlador y allí se lo transfiere a las aplicaciones adecuadas, además el correcto manejo de los flujos permite la modificación y análisis de paquetes y la obtener estadísticas. (noxrepo.org, 2014)

#### 2.4.3.2. Componentes

Las funciones y eventos en NOX son proporcionados por las aplicaciones control, entre las cuales se observan:

- **Aplicaciones de Core**

Las aplicaciones del núcleo de NOX se encuentran en el directorio `src/nox/coreapps/`, las cuales ofrecen funciones para aplicaciones de red y servicios web, dichas aplicaciones son:

- **Messenger:** abastece de sockets desde el servidor TCP/IP para la comunicación con otros equipos.
- **Snmp:** utiliza `snmptrap`, por medio de un script desarrollado en Python, para el manejo empleando `NetSNMP`. (onlab.us, 2015)

- **Aplicaciones de red**

Las aplicaciones de red se encuentran en el directorio `src/nox/netapps/`, forman un conjunto de aplicaciones las cuales son:

- **Discovery:** seguimiento de los enlaces de los switches que se conectan al controlador.
- **Topology:** registro de los enlaces, hasta la actual configuración de la red.
- **Authenticator:** ubicación de las PC y switches de la red.
- **Routing:** calcula la ruta
- **Monitoring:** muestra cada cierto tiempo las estadísticas de los switches.

- **Aplicaciones web**

Se encuentran en el directorio `src/nox/webapps/`, son usadas para gestionar el controlador por medio de servicios web y son:

- **Webservice:** interfaz de los servicios web para aplicaciones.
- **Webserver:** tiene la interfaz de control

- Webserviceclient: tiene la interfaz del cliente. (onlab.us, 2015)

#### 2.4.3.3. Eventos

Los eventos son los elementos encargados de la ejecución de NOX, los eventos existentes son:

- **Eventos de Core:**

Se asignan de forma directa a los mensajes que se reciben de un switch, estos son:

- Datapath\_join\_event: muestra cuando se agrega un switch a la red.
- Datapath\_leave\_event: muestra cuando un switch sale de la red.
- Packet\_in\_event: es un llamado para un paquete recibido, en él se incluye el ID del switch, puerto de entrada y bufer de paquete
- Flow\_mod\_event: muestra cuando se agrega un nuevo flujo o asido modificado.
- Flow\_removed\_event: muestra cuando un flujo a caducado o ha sido eliminado
- Port\_status\_event: muestra un cambio en el estado del puerto.
- Port\_status\_in: producido por un mensaje Port\_status que es recibido por el switch como respuesta a un mensaje Port\_status\_request.

- **Eventos de aplicación:**

Los elementos de una red pueden definir ciertos eventos de nivel superior, los cuales son manejados por otros eventos, los eventos que se usan en componentes NOX son:

- Host\_event: este evento es creado por el Authenticator, cuando se agrega o se quita una PC de la red,
- Flow\_in\_event: este evento es creado por el Authenticator, cuando se recibe desde la red un Packet\_in\_event.

- **Link\_event:** este evento es creado por Discovery, cuando hay cambios en la red, usado para crear la topología de la red. (noxrepo.org, 2014)

#### 2.4.4. POX

NOX fue tomado como base para el desarrollo de POX, es un controlador de OpenFlow que se encuentra en constante desarrollo, creado para cubrir las necesidades de las SDN con el lenguaje Python para otros sistemas operativos.

Las características principales de POX son:

- Su interfaz está desarrollada en Python.
- Se desarrolló para Windows, Mac OS y Windows.
- La interfaz gráfica y las herramientas para visualizar son parecidas a las de NOX
- Los elementos muestran la ruta de acceso, la topología, entre otras.

#### 2.4.5. Componentes, objetos, eventos y acciones de POX

##### 2.4.5.1. Componentes

POX está compuesto por elementos que ofrecen funciones de núcleo, específicas y aplicaciones simples, estos elementos son:

- **PY:** permite iniciar un intérprete Python para la depuración y experimentación interactiva.
- **Forwarding.I2\_learning:** permite a los switches OpenFlow comportarse como switches de capa 2
- **Forwarding.I2\_pairs:** permite a los switches OpenFlow ser tipo L2, instalando reglas basadas en la dirección MAC
- **Forwarding.I3\_learning:** no precisa el comportamiento ni de un router ni de un switch capa 2, utiliza una librería para examinar y crear solicitudes y respuestas ARP.

- **Forwarding.I2\_multi:** es un switch de aprendizaje, el aprendizaje entre switches se realiza de switch a switch, y de esta forma aprender toda la topología de red.
- **OpenFlow.spanning\_tree:** usa el descubrimiento y crea una vista de la topología de la red, deshabilitando los floodings de los puertos para evitar bucles, además construye un árbol de expansión de la red.
- **Web.webcore:** se arranca un servidor Web en POX. Para que otros componentes interactúen con el mismo proporcionando contenido web estático y dinámico.
- **Messenger:** ofrece una interfaz para comunicarse con procesos externos por medio de mensajes JSON, se usan sockets TCP y HTTP.
- **Misc.arp\_responder:** responde a las peticiones ARP.
- **Misc.dns\_spy:** revisa las respuestas DNS e indica resultados de las mismas.
- **Misc.mac\_blocker:** es usado con aplicaciones de reenvío como I2\_learning y I2\_pairs, permite bloquear direcciones Mac por medio de una interfaz de usuario.
- **OpenFlow.of\_01:** se comunica con switches OpenFlow 1.0.
- **OpenFlow.discovery:** envía mensajes LLDP, desde los switches para revelar la topología de red.
- **OpenFlow.debug:** permite crear trazas con los mensajes de OpenFlow, dichas trazas pueden ser analizadas por Wireshark.
- **OpenFlow.keepalive:** envía en intervalos de tiempo solicitudes “echo” a los switches.  
(openflow.stanford, 2015)

#### 2.4.5.2. Objeto Core de POX

El objeto núcleo o Core es el sitio central de las aplicaciones de POX, es capaz de ofrecer compatibilidad entre los componentes

#### 2.4.5.3. Sistema de eventos

Se denomina `pox.lib.revent`, se encarga de la publicación e inscripción, si un objeto público un evento, se puede inscribir a eventos determinados de otros objetos.

#### 2.4.5.4. Sistema de paquetes

POX posee una librería para la construcción y el análisis de diferentes paquetes, los paquetes poseen cabecera y carga útil, entre los cuales son: Ethernet, ARP, IPv4, ICMP, TCP, UDP, DHCP, DNS, VLAN.

#### 2.4.5.5. Acciones

Son aplicadas a paquetes coincidentes con políticas en un flujo de datos, las más habituales son:

- **OUTPUT:** envía paquetes a puertos físicos y virtuales, se trata a los puertos físicos con números y a los virtuales con símbolos
- **Set Ethernet source or destination address:** identifica el origen y destino MAC
- **Set IP source or destination address:** identifica origen y destino IP
- **Set VLAN ID:** el ID es el máximo y la prioridad cero “0”, si no existe cabecera
- **Set IP type or service:** usado para poner el campo ToS en paquetes IP.
- **Set TCP/UDP source or destination port:** usado para poner el Puerto TCP/UDP de origen y destino. (openflow.stanford, 2015)

### 2.4.6. BEACON y su funcionamiento

#### 2.4.6.1. Características

Beacon es un controlador muy eficaz y rápido, multiplataforma y modular, está desarrollado en el lenguaje Java, además provee paquetes con características avanzadas que permiten determi-



nar el mejor camino y el más corto, la detección de la topología y una interfaz web, este controlador está basado en eventos y threads que son subprocesos planificados para un sistema.

(MorilloFuentala, 2014)

Las características de Beacon son:

- **Estable:** soporta 100 switches virtuales y 20 físicos, se han realizado muchas pruebas.
- **Multiplataforma:** desarrollado en Java y se ejecuta en muchas plataformas.
- **Código abierto:** combinación de licencia GPL v2 y de la Universidad de Stanford.
- **Dinámico:** se puede iniciar, detener, instalar y actualizar paquetes durante la ejecución sin afectar otros paquetes.
- **Desarrollo rápido:** fácil de descargar y ejecutar, desarrollo simple y filtrado de aplicaciones.
- **Rápido:** debido a que realiza multiprocesamiento.
- **Interfaz web de usuario:** servidor web Jetty Enterprise e interfaz de usuario personalizada.
- **Guías:** ofrece guías y manuales para un correcto funcionamiento y eficaz.

#### 2.1.1.1. Funcionamiento

Beacon funciona por medio de la utilización de módulos llamados “budles”, los cuales poseen metadatos, clases de Java y otros archivos, los budles de Beacon funcionan en conjunto, las características de los budles son:

- Pueden importar y exportar paquetes Java.
- Ofrecen una interfaz web de gestión.
- Soportan OpenFlow 1.0,
- Compatibilidad con paquetes TCP/UDP e IP
- Operaciones basadas en eventos e hilos.

- Pueden correr varias versiones al mismo tiempo.
- Ofrecen módulos para la creación de aplicaciones.
- Se pueden modificar paquetes en la ejecución sin dañar otros paquetes.
- Manejan Proxies ARP y DHCP. (Erickson, 2013)

#### **2.4.7. Aplicaciones de BEACON**

La aplicación de Beacon funciona con una librería llamada “OpenflowJ”; el desarrollo de la API de Beacon resulta simple, debido a que su diseño no es complicado, permitiendo la utilización de cualquier Java.

Algunas de las aplicaciones desarrolladas en el core de Beacon son:

- **Device manager**

Se encarga de buscar dispositivos y de registrar los eventos de los mismos, al agregar o quitar un dispositivo, esto lo realiza por medio de la interfaz IDeviceManager, puede identificar características de los dispositivos de la red como la dirección MAC, la IP, el switch al que está conectado, el puerto y la última detección que se registró.

- **Topology**

Se obtiene un listado en el que se encuentra la información de los enlaces existentes entre los switches y los eventos, con ellos se identifican cuando un enlace es agregado o quitado de la red, esto lo realiza utilizando la interfaz ITopology, permitiendo descubrir todos los switches de una red.

- **Routing**

Depende de las interfaces utilizadas en las aplicaciones DeviceManager y Topology, permite determinar el camino más corto para que interactúen dos dispositivos, lo realiza por medio de la interfaz IRoutingEngine.

- **Web**

Ofrece una interfaz web para el usuario, lo realiza por medio de la interfaz IWebManegeable.

(opennetworking, 2013)

#### **2.4.8. FLOODLIGTH**

Este controlador fue desarrollado en Java con una licencia Apache, puede resistir tanto switches virtuales como físicos, está compuesto por un módulo principal, el cual se encarga de identificar los paquetes y asignar eventos, también lo forman módulos secundarios registrados con el modulo principal, encargados de operar y anexar entradas a las tablas de flujos.

#### **2.4.9. Módulos FLOODLIGHT**

El funcionamiento de Floodlight se basa en módulos entre los cuales existen dos tipos:

##### **2.4.9.1. Módulos de controlador**

En los módulos del controlador se encuentran las características que se usan en las aplicaciones continuamente, como la revelación de topología y los eventos, el enlace con el controlador, una interfaz web de usuario, y los recursos compartidos de la red.

Los módulos que posee Floodlight en cuanto al controlador son:

- **FloodlightProvider:** manipula los enlaces entre switches y envía mensajes a otros switches.
- **DeviceManagerImpl:** determina la ubicación de los equipos de red.
- **LinkDiscoveryManager:** revela y conserva el estado de los enlaces de la red.
- **TopologyService:** guarda la información acerca de la topología.
- **RestApiServer:** muestra los módulos en la API REST por medio de HTTP.
- **MemoryStorageSource:** almacena la información. (floodlight.atlassian, 2015)

#### 2.4.9.2. Módulos de aplicación

Estos módulos se encargan de introducir funciones determinadas, como las de un switch capa 2, firewall o enlaces bajados, los módulos de aplicación son los siguientes:

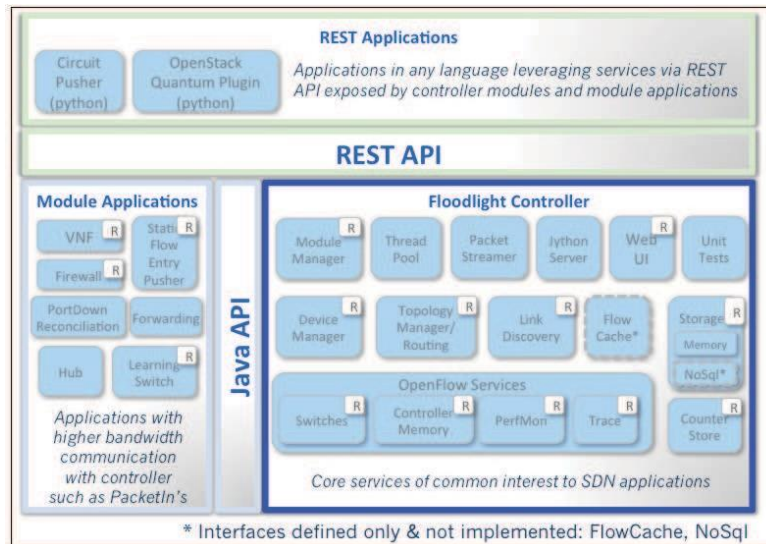
- **Virtual NetworkFilter:** ofrece la creación de muchas redes virtuales en el nivel de capa 2.
- **Forwarding:** admite el envío de paquetes entre dos equipos.
- **Firewall:** se encarga de gestionar el cumplimiento de las reglas que aceptan o rechazan el flujo del tráfico.
- **Port Down Reconciliation:** el tráfico que va por un puerto se descarta si dicho puerto deja de funcionar. (projectfloodlight, 2014)

#### 2.4.10. La aplicación de Floodlight

Floodlight se maneja a partir de un conjunto de aplicaciones llamados API REST como se muestra en la Figura 5, la cual ofrece diferentes funcionalidades a las que se puede acceder por medio del puerto 8080 desde el controlador, utilizando 3 módulos que son:

- **Firewall:** permite la creación de políticas, para que se acepte o se rechace un flujo de tráfico de red.
- **Filtro de red virtual:** permite virtualizar una red al nivel de capa 2, creando muchas redes lógicas que funcionan en un dominio de la misma capa.
- **StaticFlowEntryPusher:** permite que los usuarios agreguen tráfico a un red de forma manual, lo realiza a través de dos métodos que son:
  - **Inserción de flujo reactivo:** es cuando un paquete no pertenece a un flujo, este paquete se envía al controlador y se lo agrega al flujo determinado y el switch conoce que acción debe tomar para el reenvío.

- **Inserción de flujo proactivo:** un flujo es insertado de forma proactiva en los switches, antes de que los paquetes lleguen al mismo. (floodlight.atlassian, 2015)



**Figura 5.** Relación controlador Floodlight, módulos y aplicaciones

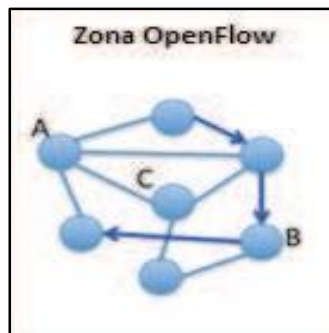
**Fuente:** James. (2012). *Floodlight OpenFlow Controller*. Obtenido de: <https://launchpad.net/floodlight-openflow>

#### 2.4.11. Topologías soportadas por FLOODLIGHT

Floodlight es capaz de soportar dos tipos de topologías que son:

- **Dentro de una zona OpenFlow:**

Una de las aplicaciones de Floodlight es Forwarding, la cual se encarga de calcular el mejor camino para la interacción de paquetes entre dos equipos, que están en una misma zona, como en la Figura 6.

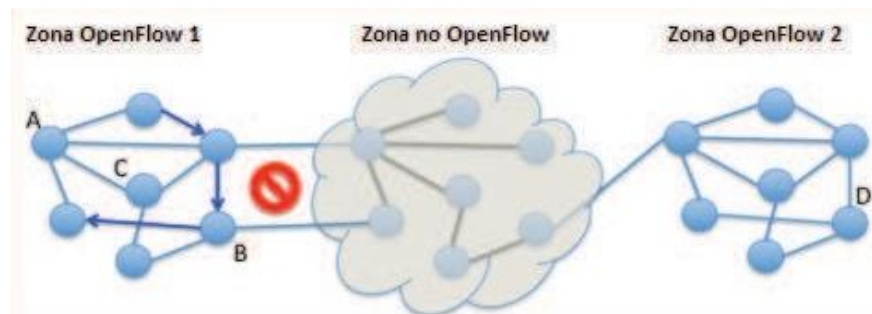


**Figura 6.** Red con Floodlight en zona OpenFlow

**Fuente:** Floodlight Project. (2012). *Supported Topologies*. Obtenido de: <http://www.openflowhub.org/display/floodlightcontroller/Supported+Topologies>

- **Dentro de zonas OpenFlow con zonas no OpenFlow entre ellos:**

La aplicación Forwarding se encarga de evaluar el mejor camino en cada zona de OpenFlow y reenvía los paquetes hacia zonas no OpenFlow, en la Figura 7, para evitar los bucles en la red las zonas OpenFlow se conectan por medio de un enlace a una zona no OpenFlow.



**Figura 7.** Red con Floodlight en varias zonas OpenFlow

**Fuente:** Floodlight Project. (2012). *Supported Topologies*. Obtenido de: <http://www.openflowhub.org/display/floodlightcontroller/Supported+Topologies>

## 2.5.APLICACIONES PARA EL CONTROL EN SDN

Existen varias aplicaciones que se pueden utilizar en una Red SDN para gestionar el plano de control, las cuales se corren en el servidor controlador y en la parte de aplicación del plan de control, entre dichas aplicaciones se encuentran el control de acceso a la red y el enrutamiento.

### 2.5.1. NAC (Control de acceso a la red)

Se denomina NAC al conjunto de tecnologías y protocolos, que permiten dar seguridad a los nodos de una red, utilizando la infraestructura de la misma para ejecutar las políticas de seguridad en cada dispositivo que solicita acceder a la red y a sus recursos. (softsecuritycorp, 2015)

El principal objetivo que NAC persigue es limitar el acceso de dispositivos a una red, lo realiza por medio de la verificación de las reglas de seguridad implementadas en la red, permitiendo el acceso únicamente a equipos confiables que cumplan con dichas reglas y que no pongan en riesgo los datos e información contenidos en la red.

Para permitir que un dispositivo accede a la red se cumplen las siguientes fases:

- **Detección:** intento de conexión física que realiza un dispositivo.
- **Cumplimiento:** comprobación del cumplimiento de las reglas de seguridad.
- **Remediación:** variación lógica de los requisitos del dispositivo.
- **Aceptación:** permite el acceso de dispositivo a la red.
- **Persistencia:** vigilancia durante la conexión del dispositivo que accedió a la red. (secutatis, 2015)

### 2.5.2. Enrutamiento

El encaminamiento de los datos en una red se puede realizar por medio de varias aplicaciones en las que se encuentran las siguientes:

- **FlowScale:** se encarga de que el tráfico de datos que pasa a través de un switch físico, sea asignado y fragmentado en múltiples puertos, utilizando balanceo de carga, por medio de software permite la manipulación del plano de control y el manejo del switch para reenviar paquetes, las características de FlowScale son:
  - Interfaz web con el usuario, la cual muestra el estado del tráfico y como se divide y asigna el mismo.
  - Configuración de políticas, para cuando el controlador envía reglas basadas en la dirección IP MAC y el puerto.
  - Interruptor TdR, el cual puede reemplazar un equipo costoso.
  - Multiplataforma, está basado en Java pero funciona en plataformas como Linux y Mac OS X.
  - HotSwapping, se encarga de descargar dinámicamente el tráfico en los puertos muy cargados y lo dirige a los que tengan menor carga.

- Mirroring traffic, permite al TdR reflejar el tráfico duplicado, directamente a los puertos de salida.
- API REST, aplicación para la administración de FlowScale. (floodlight.openflow, 2015)
- **RouteFlow:** ofrece soluciones virtuales de encaminamiento IP por medio de hardware para redes OpenFlow, combinando el rendimiento del hardware comercial con la flexibilidad del enrutamiento en código abierto.

Componentes de un entorno RouteFlow:

- Aplicación para el controlador de OpenFlow.
- Servidor RouteFlow.
- Red virtual. (sites, 2015)

RouteFlow se encuentra basado en las siguientes tecnologías:

- Switches OpenFlow habilitados.
- Controladores OpenFlow.
- Quagga, utilizado para el servicio de enrutamiento.
- Open vSwitch, para conectar maquina virtuales.
- MongoDB, proporciona la base de datos para la red.
- LXC contenedores, máquinas virtuales para ejecutar RFClient y Quagga.  
(cpqd.github, 2014)
- **Quagga:** está diseñado para software libre y proporciona servicios de enrutamiento por medio de rutas estáticas y de los protocolos OSPFv2, OSPFv3, RIP y BGP-4 para switches habilitados con OpenFlow, lo realiza por medio de un demonio llamado Zebra, el cual se encarga de la manipulación de las tablas de enrutamiento.



Características de Quagga:

- Soporte de SNMP y MIB
- Usa una arquitectura de software para alto rendimiento y soporte multi-servidor.
- Interfaz de usuario para cada protocolo de enrutamiento.
- Se distribuye con la Licencia Pública General GNU. (MorilloFuentala, 2014)

### 2.5.3. Seguridad de la red

En OpenFlow se han desarrollado algunos proyectos para brindar seguridad a la red, entre los cuales existen:

- **Fresco:** es un conjunto de aplicaciones en desarrollo para brindar servicios de seguridad en una red, Fresco permite ingresar a la sucesión de comandos de flujos de red y a los estados de toda la conexión, con la finalidad de atenuar amenazas causadas por la implementación de reglas de seguridad en el controlador, fresco proporciona funciones como Firewall, detección de intrusos y la exploración. (MorilloFuentala, 2014)

La unidad básica de fresco es el módulo, el cual se considera la parte más importante de Fresco, las funciones de seguridad ejecutada en Fresco se realizan a través de un conjunto de módulos, que se encuentran definidos como objetos en los q se incluyen las interfaces de entrada, salida, parámetro, acción y evento, utilizadas para la transmisión y recepción de datos de los módulos, la configuración del mismo y el momento en el que debe realizar una acción. (Shin, 2013)

- **Fortnox:** se encarga de extender el controlador NOX utilizado en OpenFlow, para proporcionar reglas de flujo, basadas en las políticas de las solicitudes de inserción de aplicaciones, tiene como principal objetivo mejorar la capacidad de NOX para cumplir las restricciones de flujo de una red producido por las aplicaciones de seguridad, Fortnox permite analizar las

reglas en tiempo real, comprobando de forma automática si las reglas se cumplen.

(MorilloFuentala, 2014)

Fortnox resuelve los conflictos existentes en las reglas mediante la derivación de funciones de autorización utilizando reglas de flujo firmadas digitalmente, donde cada aplicación puede firmar sus solicitudes de inserción de una regla, resultando en una asignación de privilegios para cada regla. (Porrás, 2012)

## **2.6. VIRTUALIZACIÓN DE LA RED**

### **2.6.1. Introducción**

Los diseños de red actuales utilizan la virtualización para los recursos de cálculo y almacenamiento, los cuales ofrecen agilidad, escalabilidad y reducción de costos pero se encuentran limitados por la virtualización de la red, la distribución de los recursos de calculado y de almacenamiento se realiza en poco tiempo de forma automática, en cambio la distribución de la red se realiza de forma manual y demora más tiempo, la virtualización de la red busca completar el proceso de virtualización, permitiendo una rápida manipulación de la red y mejorando la operatividad de la misma. (hp.com, 2014)

La virtualización de la red hace posible que la red sea más portátil y escalable, haciendo más fácil la gestión del ciclo de vida de la misma, una red virtualizada combina los recursos del hardware con los recursos de red de software en una sola administración, facilitando el uso compartido de recursos para brindar rapidez, seguridad y control a los usuarios y sistemas. (docs.oracle, 2014)

Los principios de virtualización de red se emplean en la infraestructura de red física, separando los servicios de la misma para proporcionar un depósito flexible de la capacidad de transporte, el cual posibilite la asignación, utilización y reasignación de acuerdo a la demanda, tomando en cuenta los posibles errores generados al manipular el hardware de la red. (vmware, 2014)

### **2.6.2. Virtualización de las funciones de red (NFV)**

La virtualización de las funciones de red, permite que los servicios de la misma sean virtuales, disminuyendo el hardware propietario utilizado para manipular y operar los servicios, lo realiza por medio de la separación de las funciones de red de equipos de hardware dedicados, los servicios que son realizador por equipos como routers, firewalls, balanceadores de carga y otros son instalados en máquinas virtuales, facilitando a los administradores de red la construcción de una cadena de servicios, permitiendo aumentar la capacidad de un servidor por medio de software. (Rouse, serachdatacenter, 2013)

La virtualización de las funciones de red presenta los siguientes beneficios:

- Reducción de CapEx: evita la compra de hardware diseñado.
- Reducción de OpEx: disminuye la utilización de espacio, energía y equipos de enfriamiento.
- Menor time to market: disminuye el tiempo de implementación de nuevos servicios y los riesgos q incluyen.
- Agilidad y flexibilidad: escalabilidad de servicios para cumplir con las demandas.

(sdxcntral, 2014)

### **2.6.3. Componentes lógicos de la red**

Se pueden tomar en cuenta los siguientes componentes, considerándolos como capas lógicas diferentes:

- **Plano de control:** se encarga de la automatización de las funciones de una red, para añadir, eliminar y restaurar circuitos, solventar un fallo, protocolos y descubrimiento de topología, anuncia y reserva de recursos, enrutamiento e intercambio de información, monitorizando el estado de los enlaces de la red. (filotecnologa, 2013)
- **Plano de reenvío lógico:** en este plano se incluyen dos aspectos fundamentales:
  - Búsqueda de tablas: las acciones del reenvío conciernen a las disponibles en el plano de reenvío físico, se muestran una o más tablas, de la capa 2, capa 3 y ACL.
  - Puertos: se pueden conectar a puertos físicos o a abstracciones de puertos que posean la interfaz de maquina virtuales. (Nicira, 2013)
- **Network Hypervisor:** permite obtener una visión global de los recursos físicos y los planos de reenvío lógicos de la red, facilita la creación de redes virtuales independientes de la red física subyacente, ofrece segmentos de una red virtual, que se manipulan de forma autónoma y se abastecen dinámicamente, trata los servicios de la red y los servidores de la misma como un conjunto de recursos, proporcionando una capa de abstracción de hardware.
- **Plano de reenvío físico:** está constituido por los elementos de reenvío que conforman la red física, el Hypervisor de red se encarga de la configuración de los elementos de reenvío tanto en el hardware como en el software, de acuerdo a lo que la red necesite con las especificaciones del plano de reenvío lógico. (Rouse, searchsdn, 2014)

## 2.7.SOFTWARE UTILIZADO EN LA VIRTUALIZACIÓN

### 2.7.1. Open vSwitch

Es un software de múltiples capas que trabaja con código abierto, permite ser utilizado como un switch virtual, el cual se encarga de reenviar tráfico entre máquinas virtuales que se encuentran en el mismo host o entre máquinas virtuales y las interfaces físicas de la red, Open vSwitch

se encarga de exportar las interfaces para manipular el estado del envío y la gestión de configuración de la red, es controlado por medio de la utilización de los protocolos OpenFlow y OVSDb. (git.openvswitch, 2012)

Open vSwitch es utilizado en entornos virtuales, permite el control y visibilidad para la capa de red virtual, realiza la distribución por medio de múltiples servidores físicos, es capaz de soportar varias tecnologías de virtualización como Xen/XenServer, KVM y Virtualbox, el código de Open vSwitch es portátil para funcionar en diversos sistemas operativos.

Open vSwitch presenta las siguientes características:

- Modelo estándar de VLAN 802.1Q con puertos de acceso y trocales.
  - NetFlow y sFlow y el reflejo de mayor visibilidad, detectar anomalías en la red.
  - Openflow 1.0 y extensiones
  - QoS de configuración y vigilancia
  - Gestión de fallos de conectividad 802.1ag
  - Base de datos de configuración con C y Python
  - Reenvío de alto rendimiento por medio del módulo del kernel de Linux
  - STP (Spanning Tree Protocol), parte del modelo OSI, gestiona bucles en la topología de red.
- (github, 2014)

### 2.7.2. Componentes de Open vSwitch

- **Ovs-vswitchd:** demonio que efectúa las funciones de switch, junto a un módulo de kernel de Linux, para el intercambio que está basado en el flujo de datos, este proceso es el principal de Open vSwitch, se comunica directamente con Ovsdb-server para manipular la base de datos que posee el Ovsdb-server.

- **Ovsdb-server:** es un servidor que posee una base de datos, en la base de datos están almacenados todos los datos de configuración del switch, el proceso Ovs-vswitchd realiza consultas para la configuración en dicha base.
- **Ovs-brcompatd:** demonio que le permite al proceso Ovs-vswitchd reemplazar por un tiempo al bridge de Linux.
- **Ovs-dpctl:** permite configurar el modulo del kernel.
- **Ovs-vsctl:** permite verificar y actualizar la configuración del proceso Ovs-vswitchd.
- **Ovs-appctl:** envía comandos para la ejecución de los demonios de Open vSwitch.
- **Ovs-ofctl:** verifica y controla los switches y los controladores.
- **Ovsdbmonitor:** permite observar de forma remota la base de datos y las tablas de flujo OpenFlow. (opennetworking, 2013)

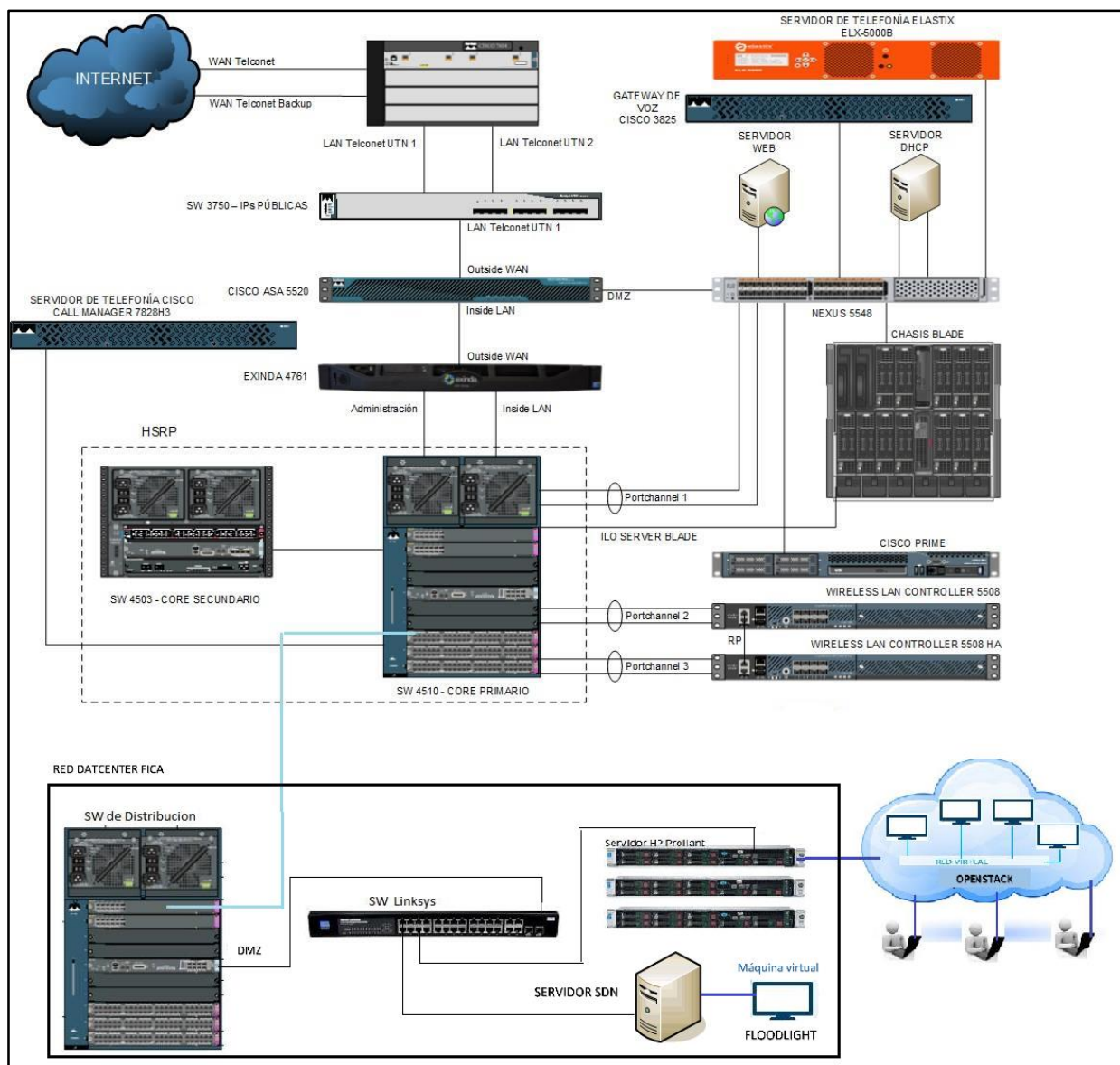
## **CAPÍTULO III**

### **IMPLEMENTACIÓN DE LA RED SDN Y DESARROLLO DE LA APLICACIÓN PARA EL CONTROL DE ACCESO (NAC).**

En el presente capítulo se muestra el diseño de la red SDN, la implementación de los diferentes controladores y de un Switch virtual, en base a las pruebas realizadas con cada controlador se realiza la elección del controlador de acuerdo a parámetros del estándar ISO/IEC/IEEE 29148, se ingresaran las reglas de flujo de datos y finalmente se presenta el desarrollo de la aplicación de acceso al medio.

#### **3.1. INFRAESTRUCTURA DE RED**

Para la realización de proyecto se plantea la infraestructura como se muestra en la figura 8, en la que se evidencia la de red física de toda la Universidad de la que forma parte el servidor SDN.



**Figura 8.** Infraestructura física de la red  
**Fuente:** Departamento de Desarrollo Tecnológico e Informático



La Red SDN que se presenta en el proyecto estará formando parte de la distribución de equipos del Data center ubicado en la Facultad FICA, conectado directamente a un switch, el cual le da acceso a internet y pasa a formar parte de la infraestructura de red de toda la Universidad, en conjunto con el servidor de OpenStack, en un futuro los servidores q se declaren dentro de la red SDN serán la base de los recursos de la nube en OpenStack.

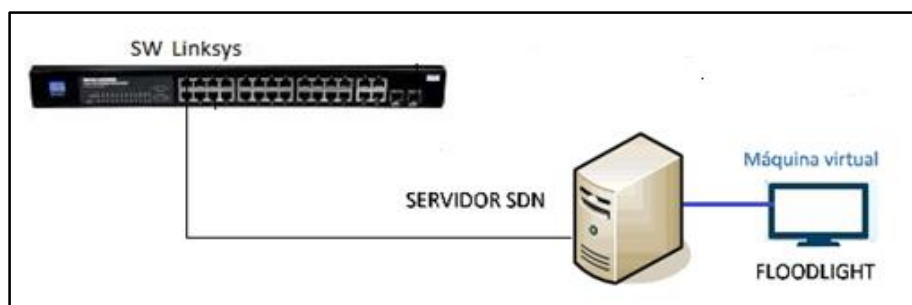
La red de la Universidad funciona a base de Vlan, que se encargan de separar el tráfico de red, de acuerdo a los requerimientos de los usuarios, en una res SDN las Vlans sieguen siendo utilizadas, pero son manejadas de forma automática, debido a que la infraestructura SDN se encarga de buscar la mejor solución y utiliza la tecnología adecuada para la conexión entre diferentes vlans.

VLAN es un esquema de capa 2 para aislar grupos de hosts en una red física común. Los usuarios de una VLAN no pueden ver a otros host en otra VLAN de capa 2. Ellos tienen que comunicarse a través de la puerta de enlace Layer3. SDN es un esquema que se encarga de encontrar una ruta de comunicación óptima entre redes (o hosts) que no requiere de cualquier protocolo distribuido que se ejecuta en los distintos nodos de la red. Todas estas rutas se calculan en una máquina central, de gran alcance llamada controlador.

Un firewall es un sistema que asegura paquetes de red entrantes, que provienen de diversas fuentes, así como paquetes de red salientes. Se puede supervisar y controlar el flujo de datos que entra en la red a partir de diferentes fuentes, y funciona sobre la base de reglas predefinidas, en una red SDN, controladores SDN pueden programar el Firewall usando la API basada en REST, con direcciones dinámicas y reorientación dinámica del tráfico.

El controlador que se utiliza en la infraestructura de red SDN, permite ingresar reglas de tráfico, dentro de sus características de Firewall, de acuerdo al controlador que se elija dichas reglas pueden ser ingresadas de forma manual o por medio de comandos.

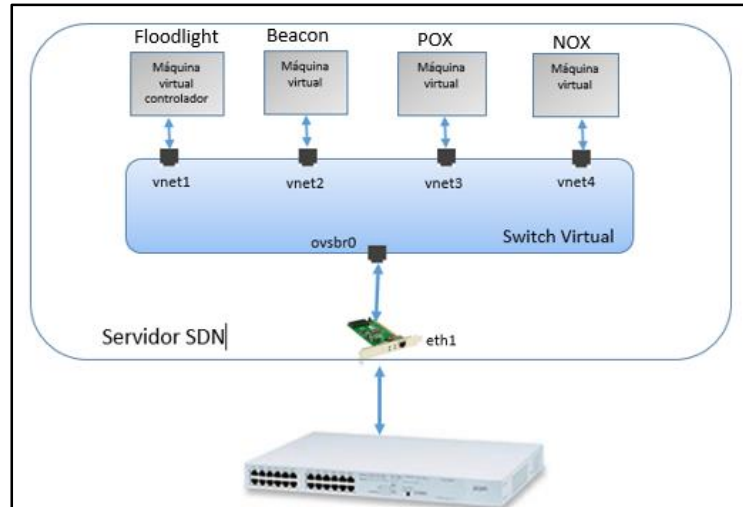
En la figura 9 se puede evidenciar la conexión desde el switch hacia el servidor SDN, por medio de este Switch físico se obtiene la conexión a internet y a la Red de toda la Universidad, se muestra la conexión física de la red SND.



**Figura 9.** Conexión desde el Switch al Servidor SDN

**Fuente:** Criterio de diseño

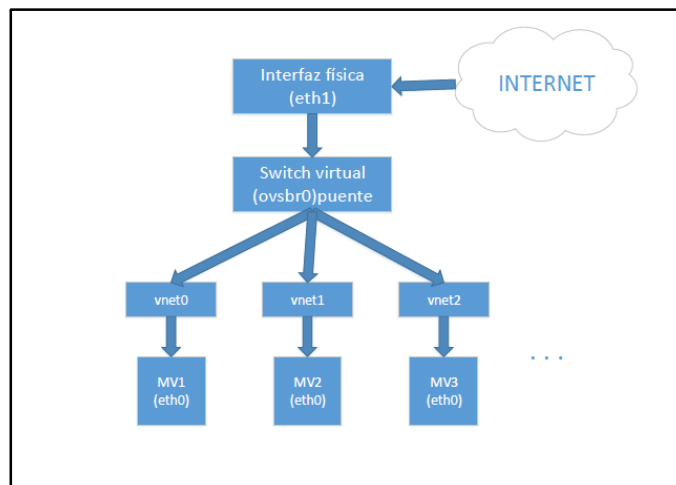
Una vez establecida la conexión física de la Red es necesario la infraestructura interna del servidor, las conexiones se realizan como se muestra en la figura 10, en la cual desde el switch físico que es la salida al internet, se conecta a la interfaz física del servidor SDN, dicha interfaz se conecta por medio de un puente al switch virtual y desde ahí se realiza la conexión a las máquinas virtuales y controladores instalados en cada máquina.



**Figura 10.** Infraestructura interna de la red  
**Fuente:** Criterio de diseño

### 3.2.CONTROLADORES E IMPLEMENTACIÓN DE SWITCH BASADO EN SOFTWARE

Para la realización del proyecto se plantea la utilización de Open vSwitch, que permite crear un puente con la interfaz física del equipo donde se instalará la red SDN, dicho puente permite la comunicación de la interfaz física con la interfaz virtual de cada una de las máquinas virtuales, como se evidencia en la Figura 11 .



**Figura 11.** Diagrama para pruebas de Open vSwitch  
**Fuente:** Paulo Colomes. (2015). *Introducción a Open vSwitch*. Obtenido de: <http://www.redescisco>

### 3.1.1. Plataforma de hardware

La red SDN se instalara en un servidor localizado en el Data Center de la Facultad de Ciencias Aplicada (FICA), las características de dicho equipo están especificadas en la Tabla 1.

**Tabla 1.** Características servidor SDN

| <b>Características técnicas del servidor utilizado en la red SDN</b> |  |
|--|--|
| <b>Fabricante</b>  | HP   |
| <b>Modelo</b>  | ProLiant ML150 G5  |
| <b>Versión</b>   | 1.0  |
| <b>Dimensiones</b>   | 61,7x42,4x20cm   |
| <b>Tipo de chasis</b>  | Torre de 5U  |
| <b>Procesador</b>  | Intel Xeon(R) CPU E5405 2.0GHz   |
| <b>Número de procesadores</b>  | 4  |
| <b>Memoria RAM</b>   | 4.8G   |
| <b>Disco duro</b>  | 232G   |
| <b>Memoria gráfica</b>   | 8MB DDR2   |
| <b>Serial</b>  | MXS8460A5J   |
| <b>Potencia eléctrica</b>  | 526 vatios   |
| <b>Fuente de alimentación</b>  | Carga 11.6A: 100 a 127 VCA; carga: 5,5A de 200 a 240 VCA, 47 a 66 Hz   |
| <b>Temperatura operativa</b>   | 10 a 35 °C   |
| <b>Sistema operativo</b>   | Ubuntu Server 14.043 LTS 64bits  |
| <b>Puertos de entrada y salida</b>                                   | <ul style="list-style-type: none"><li>- Serie: 1</li><li>- Puntero (ratón, PS2): 1</li><li>- Gráficos: 1</li><li>- Teclado (PS2): 1</li><li>- USB 2.0: 8 (4 posteriores, 2 panel frontal, 2 internos )</li><li>- RJ-45 (Ethernet): 1 (10/100/1000 Gbits/s)</li></ul> |

**Fuente:** Amazon. (2008). *HP ProLiant ML150 G5 Intel® Xeon® E5405 Quad Core Processor 2 GHz 12MB 2GB 1P SATA/SAS Base Model Tower Server*. Obtenido de: <https://goo.gl/V0afpN>

### 3.1.2. Configuración de la red SDN

Para la realización de la red SDN, en el servidor físico se dispondrá del sistema operativo Ubuntu Server 14.043 LTS, en dicho servidor se instalara el Software Open vSwitch, el software de maquina virtuales KVM y el gestor Virt Manager, se crearan máquinas virtuales con cada

uno de los controladores y dos clientes para la realización de pruebas para la elección del controlador más apto para la red, las máquinas virtuales cuentan con el sistema operativo Ubuntu 14.043 Desktop, en el anexo A se detalla la instalación de KVM y Virt Manager.

El gestor Virt Manager, configura por defecto una interfaz virbr0 que cumple de puente con la interfaz física del servidor, la cual de ver ser agregada al switch virtual, para que las máquinas virtuales puedan conectarse, por cada máquina virtual que se cree se crea de forma automática una interfaz virtual denominada vnetn donde “n” es el identificador de la máquina virtual. La instalación y configuración de Open vSwitch se encuentran en el anexo B. En la Tabla 2 se detalla la tabla de direcciones que se utilizan para el servidor SDN, las máquinas virtuales de los controladores, y los clientes

**Tabla 2.** Direcciones IP de servidor y máquinas virtuales

| Tabla de direcciones   |                 |                  |
|------------------------|-----------------|------------------|
| Equipo                 | Dirección IP    | Puerta de enlace |
| Servidor SDN           | 10.24.8.77 /24  | 10.24.8.1        |
| Controlador NOX        | 10.24.8.205 /24 | 10.24.8.1        |
| Controlador POX        | 10.24.8.206 /24 | 10.24.8.1        |
| Controlador Beacon     | 10.24.8.207 /24 | 10.24.8.1        |
| Controlador Floodlight | 10.24.8.208 /24 | 10.24.8.1        |
| Cliente1               | 10.24.8.209 /24 | 10.24.8.1        |
| Cliente2               | 10.24.8.210 /24 | 10.24.8.1        |

**Fuente:** Criterios de diseño

Para verificar el mejor controlador para la red SDN, se realizan una serie de pruebas que se presentan a continuación, en las que se incluyen pruebas de configuración y funcionamiento de Switch Virtual (Open vSwitch) y de cada controlador.

Mediante el comando ifconfig en la Figura 12, se puede observar la configuración de las interfaces, tanto de la física eth1 y de la interfaz puente virbr0, además de las interfaces vnet de cada máquina virtual.

```
root@SDN: /home/sdnfica
root@SDN:/home/sdnfica# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:23:7d:93:69:cb
          inet6 addr: fe80::223:7dff:fe93:69cb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:608448 errors:0 dropped:0 overruns:0 frame:0
          TX packets:127886 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:556105301 (556.1 MB)  TX bytes:10840627 (10.8 MB)
          Interrupt:17

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:480471 errors:0 dropped:0 overruns:0 frame:0
          TX packets:480471 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:257169217 (257.1 MB)  TX bytes:257169217 (257.1 MB)

ovsbr0    Link encap:Ethernet  HWaddr 00:23:7d:93:69:cb
          inet addr:10.24.8.77  Bcast:10.24.8.255  Mask:255.255.255.0
          inet6 addr: 2800:68:19:2408:48a6:48f0:7b9e:7df/64 Scope:Global
          inet6 addr: fe80::e86b:f9ff:fe5b:e4e0/64 Scope:Link
          inet6 addr: 2800:68:19:2408:1dc7:af96:2e09:337a/64 Scope:Global
          inet6 addr: 2800:68:19:2408:223:7dff:fe93:69cb/64 Scope:Global
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:196074 errors:0 dropped:10 overruns:0 frame:0
          TX packets:16346 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:55569935 (55.5 MB)  TX bytes:879943 (879.9 KB)

vnet0     Link encap:Ethernet  HWaddr fe:54:00:fd:02:cb
          inet6 addr: fe80::fc54:ff:fed:2cb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:111473 errors:0 dropped:0 overruns:0 frame:0
          TX packets:544312 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:9008865 (9.0 MB)  TX bytes:542395332 (542.3 MB)

vnet1     Link encap:Ethernet  HWaddr fe:54:00:ce:5e:42
          inet6 addr: fe80::fc54:ff:fece:5e42/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

**Figura 12.** Resultado ifconfig  
**Fuente:** Servidor SDN

En la Figura 13, la interfaz física del servidor se asoció con una interfaz virtual, por medio de un puente, con el comando ovs-vsctl show se puede verificar la configuración de dicha interfaz, además se visualizan las interfaces vnet activas.

```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ovs-vsctl show
f3cbf526-b407-4654-8a39-54816f235757
    Bridge "ovsbr0"
        Port "vnet3"
            Interface "vnet3"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
        Port "vnet1"
            Interface "vnet1"
        Port "eth1"
            Interface "eth1"
    ovs_version: "2.0.2"
root@sdnfica:/home/sdnfica#
```

Figura 13. Resultado ovs-vsctl show

Fuente: Servidor SDN

En la Figura 14, se puede verificar los procesos que crea el switch virtual al iniciar y los módulos q emplea, por medio del comando ps -ea | grep ovs y en la Figura 15, se reinicia el servicio.

```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ps -ea | grep ovs
20640 ?        00:01:01 ovsdb-server
20651 ?        00:10:16 ovs-vswitchd
root@sdnfica:/home/sdnfica#
```

Figura 14. Resultado de ps -ea | grep ovs

Fuente: Servidor SND

```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# /etc/init.d/openvswitch-switch restart
openvswitch-switch stop/waiting
openvswitch-switch start/running
root@sdnfica:/home/sdnfica#
```

Figura 15. Resultado de /etc/init.d/openvswitch-switch restart

Fuente: Servidor SND

Las pruebas con los diferentes controladores instalados en la red SDN, tomado en cuenta que el Open vSwitch posee dos puertos por default por los cuales escucha y espera paquetes, esto puertos son el 6633 y 6653:

## NOX

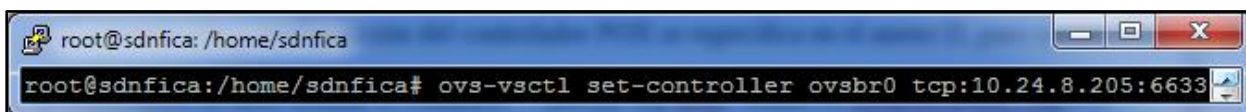
La instalación y configuración del controlador NOX se especifica en el anexo C, para iniciar el controlador se debe utilizar el comando:`/nox_core -i tcp switch`, en este caso el switch se comporta como uno de capa 2.

Para iniciar el controlador de forma general se utiliza:

```
./nox_core --i [protocolo]: [ip_controlador]: [puerto]-[otros]
```

En donde, el campo protocolo se inserta el tipo de protocolo se TCP o SSL, luego la dirección IP del controlador, seguido del puerto de escucha y en el campo otros es para las opciones de funcionamiento del switch.

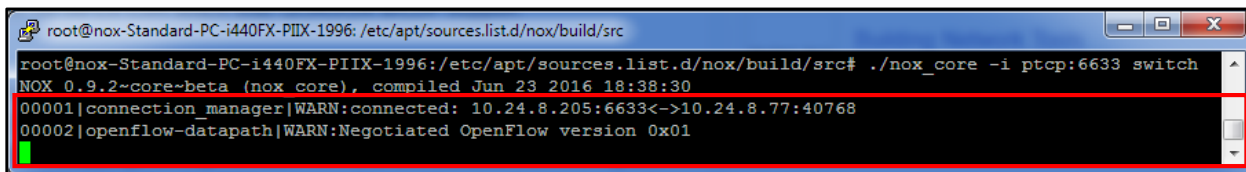
En la Figura 16 se establece el datapath en el switch con la dirección IP del controlador y el puerto de escucha del switch., que en este caso es el 6633.



```
root@sdfnca: /home/sdfnca
root@sdfnca: /home/sdfnca# ovs-vsctl set-controller ovsbr0 tcp:10.24.8.205:6633
```

**Figura 16.** Asignación de Datapath NOX  
**Fuente:** Servidor SDN

En la Figura 17 se inicializa el controlador y se establece la conexión con el switch.

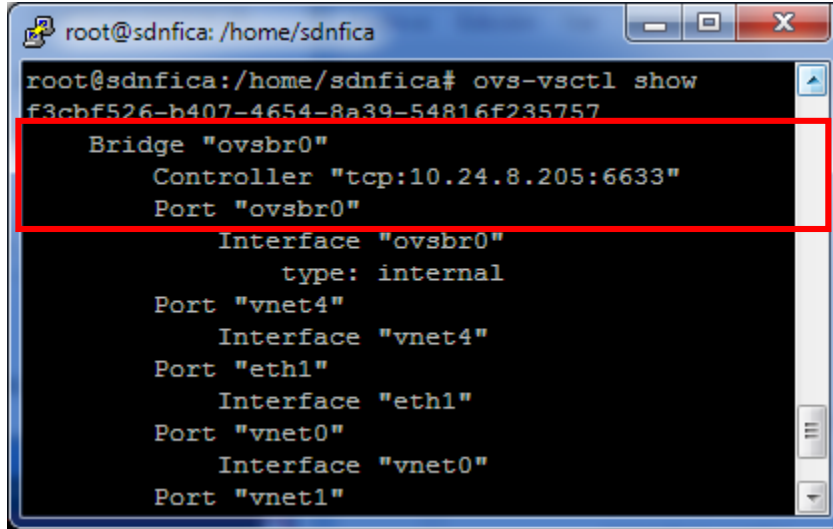


```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src# ./nox_core -i tcp:6633 switch
NOX 0.9.2~core~beta (nox core), compiled Jun 23 2016 18:38:30
00001|connection_manager|WARN:connected: 10.24.8.205:6633<->10.24.8.77:40768
00002|openflow-datapath|WARN:Negotiated OpenFlow version 0x01
```

**Figura 17.** Conexión entre NOX y switch  
**Fuente:** Máquina virtual controlador NOX



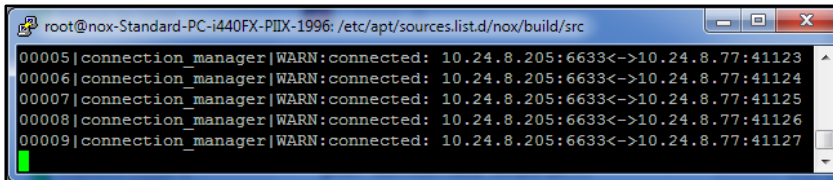
En la Figura 18 se observa la ruta hacia el controlador desde el switch.



```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ovs-vsctl show
f3cbf526-b407-4654-8a39-54816f235757
Bridge "ovsbr0"
  Controller "tcp:10.24.8.205:6633"
  Port "ovsbr0"
  Interface "ovsbr0"
    type: internal
  Port "vnet4"
    Interface "vnet4"
  Port "eth1"
    Interface "eth1"
  Port "vnet0"
    Interface "vnet0"
  Port "vnet1"
```

**Figura 18.** Comunicación entre el switch y NOX  
**Fuente:** Servidor SDN

NOX genera mensajes los cuales se ven en la Figura 19



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
00005|connection_manager|WARN:connected: 10.24.8.205:6633<->10.24.8.77:41123
00006|connection_manager|WARN:connected: 10.24.8.205:6633<->10.24.8.77:41124
00007|connection_manager|WARN:connected: 10.24.8.205:6633<->10.24.8.77:41125
00008|connection_manager|WARN:connected: 10.24.8.205:6633<->10.24.8.77:41126
00009|connection_manager|WARN:connected: 10.24.8.205:6633<->10.24.8.77:41127
```

**Figura 19.** Mensajes generados por NOX  
**Fuente:** Máquina virtual controlador NOX

## POX

La instalación y configuración del controlador POX se especifica en el anexo D, para iniciar el controlador se debe utilizar el comando `./pox.py log level -DEBUG samples.pretty_log Forwarding.l2_learning,`

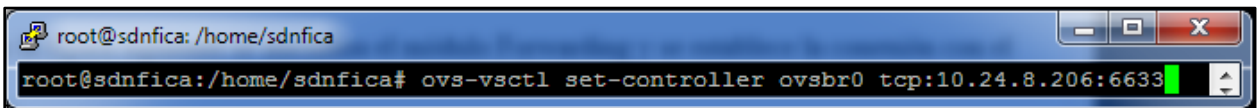
Para iniciar el controlador de forma general se utiliza:

```
./pox.py --[opciones] [OpenFlow_ versión] - [ip_controlador] - [puerto] - [otros]
```

En donde, el campo de opciones permite activar el debug, luego se ingresa la versión de OpenFlow, el siguiente campo representa la IP asignada a la máquina virtual donde está insta-

lado el controlador, luego el puerto de escucha del Switch y en el campo de *otros*, se puede ingresar las funcionalidades de acuerdo a los requerimientos, en este caso se ingresará con una función de aprendizaje L2.

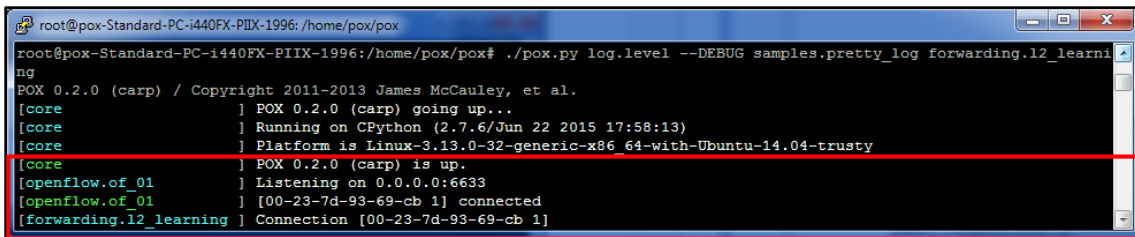
En la Figura 20 se establece el datapath en el switch con la dirección IP del controlador y el puerto de escucha del switch., que en este caso es el 6633.



```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ovs-vsctl set-controller ovsbr0 tcp:10.24.8.206:6633
```

**Figura 20.** Asignación de Datapath POX  
**Fuente:** Servidor SDN

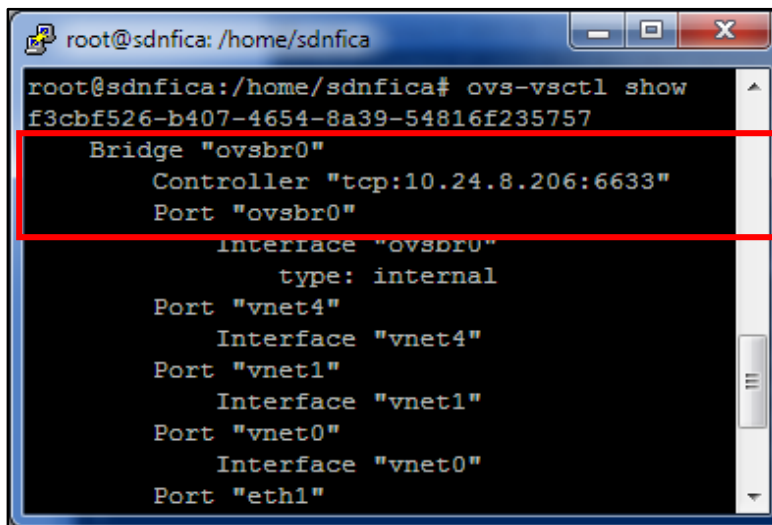
En la Figura 21 se inicializa el módulo Forwarding y se establece la conexión con el switch.



```
root@pox-Standard-PC-i440FX-PIIX-1996:/home/pox/pox
root@pox-Standard-PC-i440FX-PIIX-1996:/home/pox/pox# ./pox.py log.level --DEBUG samples.pretty_log forwarding.l2_learning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
[core ] POX 0.2.0 (carp) going up...
[core ] Running on CPython (2.7.6/Jun 22 2015 17:58:13)
[core ] Platform is Linux-3.13.0-32-generic-x86_64-with-Ubuntu-14.04-trusty
[core ] POX 0.2.0 (carp) is up.
[openflow.of_01 ] Listening on 0.0.0.0:6633
[openflow.of_01 ] [00-23-7d-93-69-cb 1] connected
[forwarding.l2_learning ] Connection [00-23-7d-93-69-cb 1]
```

**Figura 21.** Conexión entre POX y Switch  
**Fuente:** Máquina Virtual controlador POX

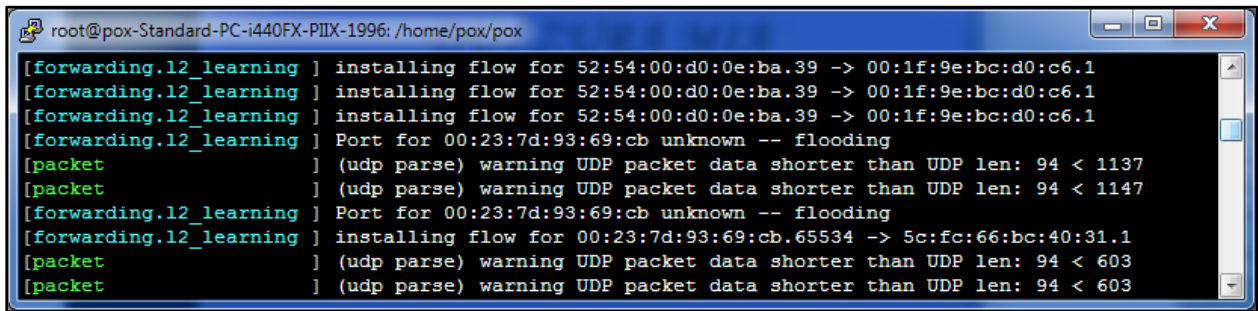
En la Figura 22 se observa la ruta hacia el controlador desde el switch.



```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ovs-vsctl show
f3cbf526-b407-4654-8a39-54816f235757
    Bridge "ovsbr0"
        Controller "tcp:10.24.8.206:6633"
        Port "ovsbr0"
    Interface "ovsbr0"
        type: internal
    Port "vnet4"
        Interface "vnet4"
    Port "vnet1"
        Interface "vnet1"
    Port "vnet0"
        Interface "vnet0"
    Port "eth1"
```

**Figura 22.** Comunicación entre el switch y POX  
**Fuente:** Servidor SDN

POX genera mensajes los cuales se ven en la Figura 23:



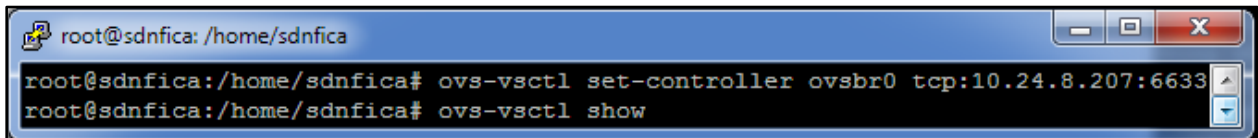
```
root@pox-Standard-PC-i440FX-PIIX-1996: /home/pox/pox
[forwarding.l2_learning ] installing flow for 52:54:00:d0:0e:ba.39 -> 00:1f:9e:bc:d0:c6.1
[forwarding.l2_learning ] installing flow for 52:54:00:d0:0e:ba.39 -> 00:1f:9e:bc:d0:c6.1
[forwarding.l2_learning ] installing flow for 52:54:00:d0:0e:ba.39 -> 00:1f:9e:bc:d0:c6.1
[forwarding.l2_learning ] Port for 00:23:7d:93:69:cb unknown -- flooding
[packet ] (udp parse) warning UDP packet data shorter than UDP len: 94 < 1137
[packet ] (udp parse) warning UDP packet data shorter than UDP len: 94 < 1147
[forwarding.l2_learning ] Port for 00:23:7d:93:69:cb unknown -- flooding
[forwarding.l2_learning ] installing flow for 00:23:7d:93:69:cb.65534 -> 5c:fc:66:bc:40:31.1
[packet ] (udp parse) warning UDP packet data shorter than UDP len: 94 < 603
[packet ] (udp parse) warning UDP packet data shorter than UDP len: 94 < 603
```

**Figura 23.** Mensajes generados por POX  
**Fuente:** Máquina virtual controlador POX

## Beacon

El proceso de instalación y configuración del controlador Beacon se encuentra especificado en el anexo E, el modulo q se ejecuta es *Beacon* por medio de eclipse, pero hay q recalcar que posee dos módulos más que son: *Beacon LearningSwitch Only* y *Beacon Tutorial LearningSwitch*


En la Figura 24 se establece el datapath en el switch con la dirección IP del controlador y el puerto de escucha del switch.



```
root@sdfica: /home/sdfica
root@sdfica: /home/sdfica# ovs-vsctl set-controller ovsbr0 tcp:10.24.8.207:6633
root@sdfica: /home/sdfica# ovs-vsctl show
```

**Figura 24.** Asignación del Datapath Beacon  
**Fuente:** Máquina virtual controlador Beacon

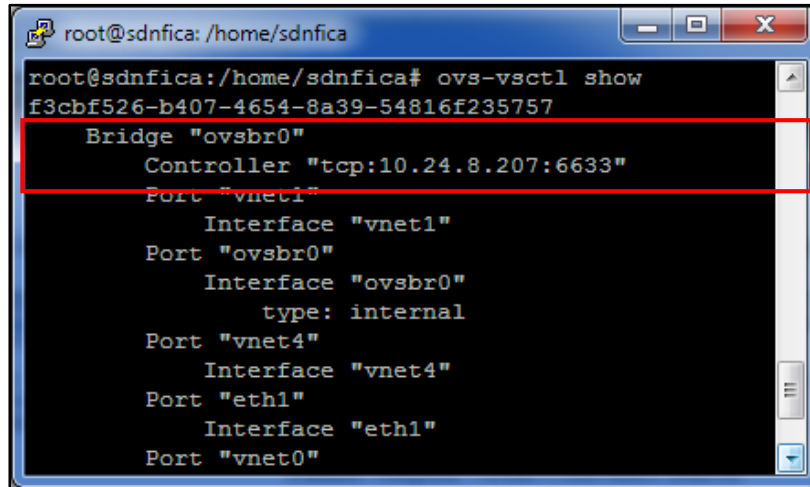
En la Figura 25 se puede ver el puerto por el que Beacon espera los paquetes:



```
beacon [OSGi Framework] /home/controlador/Escritorio/eclipse/jre/bin/java (23/04/2013 9:23:16)
er - Started thread Thread[pool-2-thread-1,5, spring-osgi-extender[34b6a6d6]-threads] for IOLoop IOLoop [id=0 s
itroller - Beacon Core Started
ontroler - Controller listening on *:6633
```

**Figura 25.** Conexión Beacon y Switch  
**Fuente:** Máquina Virtual Controlador Beacon

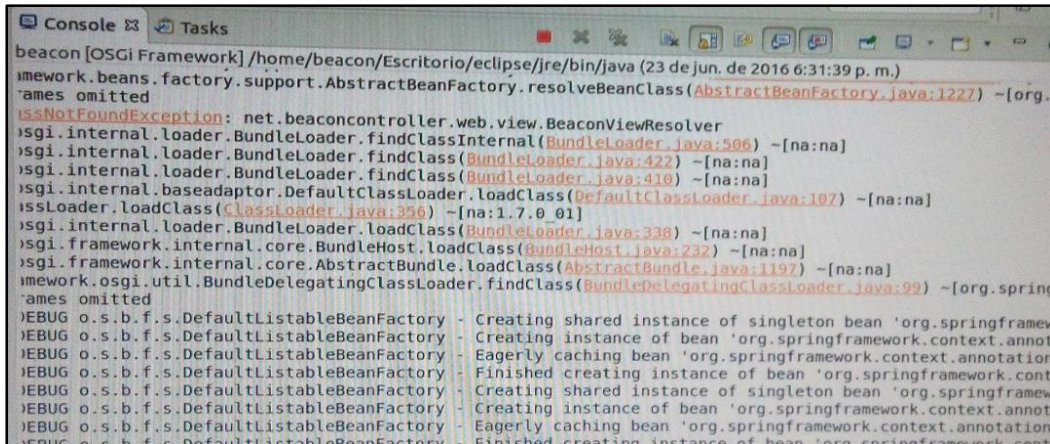
En la Figura 26 se observa la ruta hacia el controlador desde el switch:



```
root@sdnfica: /home/sdnfica
root@sdnfica: /home/sdnfica# ovs-vsctl show
f3cbf526-b407-4654-8a39-54816f235757
Bridge "ovsbr0"
Controller "tcp:10.24.8.207:6633"
Port "vnet1"
Interface "vnet1"
Port "ovsbr0"
Interface "ovsbr0"
type: internal
Port "vnet4"
Interface "vnet4"
Port "eth1"
Interface "eth1"
Port "vnet0"
```

**Figura 26.** Comunicación entre el switch y Beacon  
**Fuente:** Servidor SDN

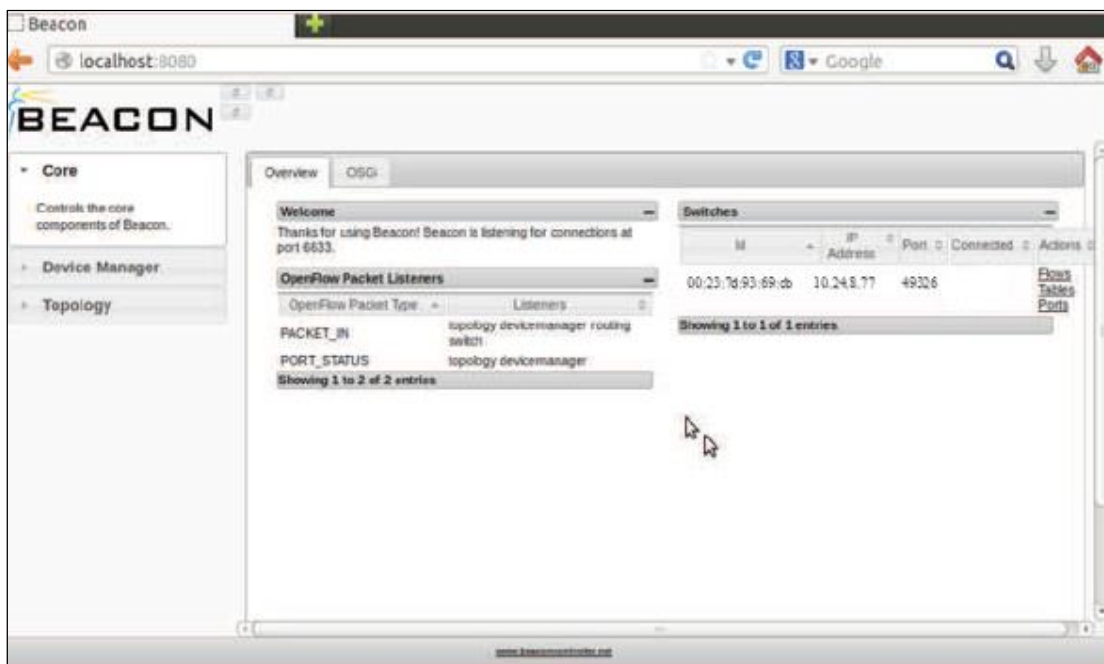
Beacon genera mensajes los cuales se ven en la Figura 27:



```
beacon [OSGi Framework] /home/beacon/Escritorio/eclipse/jre/bin/java (23 de jun. de 2016 6:31:39 p. m.)
amework.beans.factory.support.AbstractBeanFactory.resolveBeanClass(AbstractBeanFactory.java:1227) ~[org.
ames omitted
ClassNotFoundException: net.beaconcontroller.web.view.BeaconViewResolver
sgi.internal.loader.BundleLoader.findClassInternal(BundleLoader.java:566) ~[na:na]
sgi.internal.loader.BundleLoader.findClass(BundleLoader.java:422) ~[na:na]
sgi.internal.loader.BundleLoader.findClass(BundleLoader.java:410) ~[na:na]
sgi.internal.baseadaptor.DefaultClassLoader.loadClass(DefaultClassLoader.java:107) ~[na:na]
ssLoader.loadClass(ClassLoader.java:356) ~[na:1.7.0_01]
sgi.internal.loader.BundleLoader.loadClass(BundleLoader.java:338) ~[na:na]
sgi.framework.internal.core.BundleHost.loadClass(BundleHost.java:232) ~[na:na]
sgi.framework.internal.core.AbstractBundle.loadClass(AbstractBundle.java:1197) ~[na:na]
amework.osgi.util.BundleDelegatingClassLoader.findClass(BundleDelegatingClassLoader.java:99) ~[org.spring
ames omitted
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating instance of bean 'org.springframework.context.annot
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Eagerly caching bean 'org.springframework.context.annotation
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Finished creating instance of bean 'org.springframework.cont
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframew
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating instance of bean 'org.springframework.context.annot
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Eagerly caching bean 'org.springframework.context.annotation
DEBUG o.s.b.f.s.DefaultListableBeanFactory - Finished creating instance of bean 'org.springframew cont
```

**Figura 27.** Mensajes generados por Beacon  
**Fuente:** Máquina Virtual Controlador Beacon

A través de un navegador se puede acceder a la interfaz de usuario de Beacon con la dirección: <http://localhost:8080>, en la Figura 28 se pueden visualizar los equipos conectados por Beacon en la red, donde se detalla la dirección física, la dirección lógica y el puerto.



**Figura 28.** Interfaz gráfica de Beacon  
**Fuente:** Máquina virtual controlador Beacon

## Floodlight

El proceso de instalación y de configuración del controlador Floodlight se encuentra especificado en el Anexo F, en la Figura 29, al ejecutar el comando `java -jar target/floodlight.jar` se observa el puerto por el cual el controlador escucha y espera los paquetes, en este caso es el puerto 6653

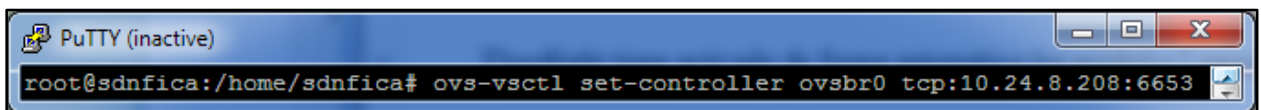
```

root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
12:29:42.008 INFO [n.f.c.i.OFSwitchManager:main] Computed OpenFlow version bitmap as [62]
12:29:42.012 INFO [n.f.c.i.Controller:main] OpenFlow port set to 6653
12:29:42.013 INFO [n.f.c.i.Controller:main] Number of worker threads set to 8
12:29:42.013 INFO [n.f.c.i.Controller:main] ControllerId set to 1
12:29:42.014 INFO [n.f.c.i.Controller:main] Controller role set to ACTIVE
12:29:42.086 INFO [n.f.l.i.LinkDiscoveryManager:main] Link latency history set to 10 LLDP data points
12:29:42.092 INFO [n.f.l.i.LinkDiscoveryManager:main] Latency update threshold set to +/-0.5 (50.0%) of rolling
historical average
12:29:42.143 INFO [n.f.f.Forwarding:main] Default hard timeout not configured. Using 0.
12:29:42.144 INFO [n.f.f.Forwarding:main] Default idle timeout set to 5.
12:29:42.145 INFO [n.f.f.Forwarding:main] Default priority not configured. Using 1.
12:29:42.145 INFO [n.f.f.Forwarding:main] Default flags will be set to SEND_FLOW_REM.
12:29:42.146 INFO [n.f.f.Forwarding:main] Default flow matches set to: VLAN=true, MAC=true, IP=true, TPPT=true
12:29:42.146 INFO [n.f.f.Forwarding:main] Not flooding ARP packets. ARP flows will be inserted for known destinations
12:29:42.146 INFO [n.f.f.Forwarding:main] Flows will be removed on link/port down events
12:29:42.147 INFO [n.f.s.StatisticsCollector:main] Statistics collection disabled
12:29:42.147 INFO [n.f.s.StatisticsCollector:main] Port statistics collection interval set to 10s
12:29:42.279 INFO [o.s.s.i.SyncManager:main] [1] Updating sync configuration ClusterConfig [allNodes={1=Node [hostname=192.168.1.100, port=6642, nodeId=1, domainId=1], 2=Node [hostname=192.168.1.100, port=6643, nodeId=2, domainId=1]}, authScheme=CREDENTIAL_RESPONSE, keyStorePath=/etc/floodlight/keys.jks, keyStorePassword_is_set}
12:29:42.820 INFO [o.s.s.i.r.RPCService:main] Listening for internal floodlight RPC on 0.0.0.0/0.0.0.0:6642
12:29:42.877 INFO [n.f.c.i.OFSwitchManager:main] Listening for switch connections on /0.0.0.0:6653
12:29:42.916 INFO [n.f.l.i.LinkDiscoveryManager:main] Setting autoportfast feature to OFF

```

**Figura 29.** Resultado de `java -jar target/floodlight.jar`  
**Fuente:** Máquina Virtual controlador Floodlight

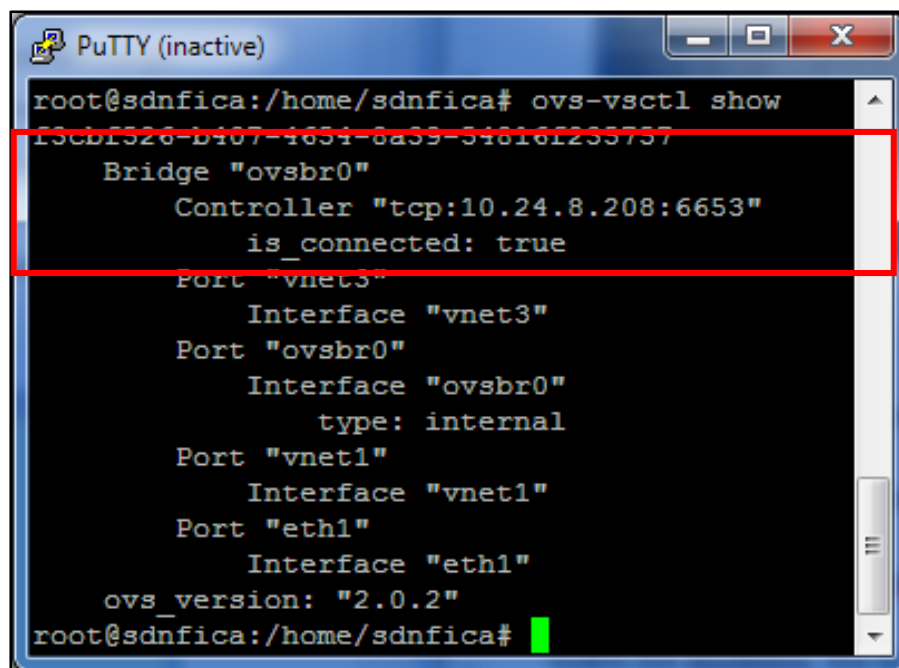
Floodlight tiene activado de forma automática los módulos LearningSwitch y Forwarding, los cuales se encargan del envío de paquetes entre dispositivos, se puede editar el archivo que se encuentra en /src/main/resources llamado Floodlightdefault.properties en el cual se encuentran todos los módulos. En la Figura 30 se establece el datapath en el switch con la dirección Ip del controlador y el puerto de escucha del switch.



```
root@sdnfica:/home/sdnfica# ovs-vsctl set-controller ovsbr0 tcp:10.24.8.208:6653
```

**Figura 30.** Asignación del Datapath Floodlight  
**Fuente:** Servidor SDN

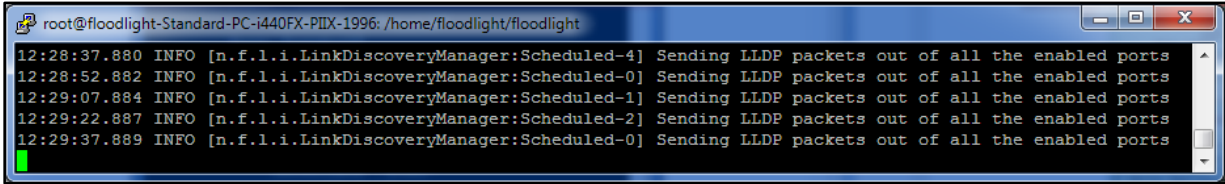
En la Figura 31 se observa la ruta hacia el controlador desde el switch:



```
root@sdnfica:/home/sdnfica# ovs-vsctl show
13cbr528-d407-4634-8a39-34816f233737
  Bridge "ovsbr0"
    Controller "tcp:10.24.8.208:6653"
    is_connected: true
  Port "vnet3"
    Interface "vnet3"
  Port "ovsbr0"
    Interface "ovsbr0"
    type: internal
  Port "vnet1"
    Interface "vnet1"
  Port "eth1"
    Interface "eth1"
  ovs_version: "2.0.2"
root@sdnfica:/home/sdnfica#
```

**Figura 31.** Comunicación entre el Switch y Floodlight  
**Fuente:** Servidor SDN

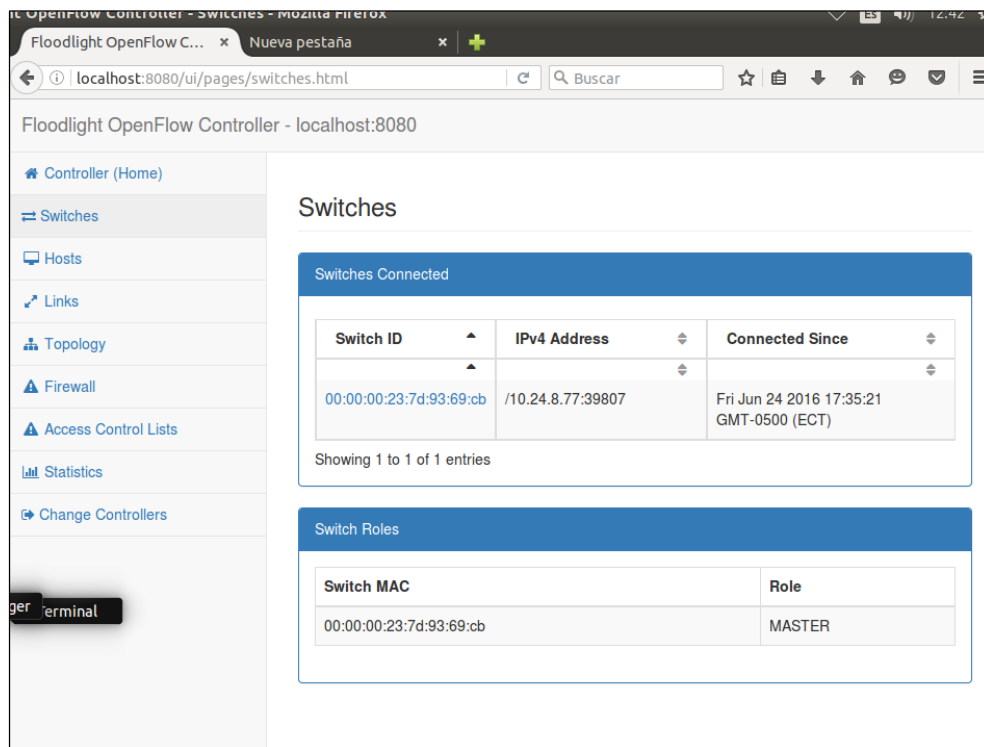
Una vez que se han cargado los módulos, Floodlight genera mensajes LLDP que se envía de forma periódica para descubrir los dispositivos que se conecten a la red, como se muestra en la figura 32.



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
12:28:37.880 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-4] Sending LLDP packets out of all the enabled ports
12:28:52.882 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-0] Sending LLDP packets out of all the enabled ports
12:29:07.884 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-1] Sending LLDP packets out of all the enabled ports
12:29:22.887 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-2] Sending LLDP packets out of all the enabled ports
12:29:37.889 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-0] Sending LLDP packets out of all the enabled ports
```

**Figura 32.** Mensajes LLDP  
**Fuente:** Máquina Virtual controlador Floodlight

Floodlight tiene una interfaz web a la que se puede acceder por medio de la dirección: <http://localhost:8080/ui/index.html>, la Figura 33 muestra la interfaz en la que se puede ver los dispositivos detectados por el controlador en la red y la topología de la misma.



**Figura 33.** Interfaz Web Floodlight  
**Fuente:** Máquina virtual controlador Floodlight

### 3.1.3. Evaluación de los controladores

Cada uno de los controladores se integra de forma sencilla con el Open vSwitch, una vez que se aclaró la función de las interfaces. Cabe mencionar que el software Open vSwitch posee un gran soporte debido a que es muy utilizado en la infraestructura de SDN.

- El controlador NOX es simple para instalar, debido a que es el primer controlador que se desarrolló para las redes SDN, no brinda todos los componentes requeridos para las aplicaciones, el soporte del mismo ya no se encuentra actualizado, además existen problemas de compatibilidad con las versiones actuales de Ubuntu.
- El controlador POX esta creado a partir de NOX , por lo que su función es cumplir con los requerimientos de SDN, brindando ejemplos de aplicaciones y de su desarrollo, es necesario conocer el lenguaje de programación Python para su manipulación , el soporte del controlador esta poco desarrollado y la documentación del mismo es escasa.
- El controlador Beacon posee muchos módulos que permiten el desarrollo de SDN, está escrito en el lenguaje de programación Java, para su instalación se necesitan muchas librerías y recursos para su funcionamiento, es necesario la utilización del software eclipse para administrarlo.
- El controlador Floodlight está escrito en el lenguaje de programación Java, la instalación del mismo es muy sencilla, posee una interfaz web muy interactiva, permite obtener funcionalidades de ruteo y conmutación, brinda una serie de módulos, entre ellos uno esencial que es Firewall, el cual servirá para la realización de la aplicación para NAC, este controlador tiene una amplia documentación y el soporte del mismo es muy desarrollado.

### **3.2. ELECCIÓN DE SOFTWARE BASÁNDOSE EN LA NORMA**

#### **ISO/IEC/IEEE 29148:201**

La norma ISO/IEC/IEEE 29148 en su última modificación del 2011, presenta los requisitos y parámetros que se debe seguir para realizar la selección de software en un proyecto, dichos parámetros se encuentran especificados en la sección “9.5 Especificación de los Requisitos de Software (SRS)” (ISO/IEC/IEEE, 2011) de la norma, esta norma se adjunta en el anexo H.



### **3.2.1. Switch virtual**

Open vSwitch es el software con el cual se va implementar la red SDN, debido a que posee muchas funciones para los controladores, este software tiene la habilidad de adaptarse a distintas plataformas, soportando diversas interfaces de gestión, está desarrollado para funcionar en ambientes con máquinas virtuales.

Open vSwitch soporta varias tecnologías de virtualización en Linux como Xen, KVM. VirtualBox, Promox, la tecnología utilizada para la implementación de la red es KVM, debido a que permite una virtualización completa sobre el sistema operativo Linux.

### **3.2.2. Servidor controlador**

Para elegir el software del servidor controlador se siguió los siguientes pasos basados en la norma ISO/IEC/IEEE 29148.

#### **3.2.2.1. Propósito**

El propósito que debe cumplir el software es controlar la red SDN, permitiendo que su administración resulte sencilla y desde el mismo se pueda verificar y observar las conexiones y la infraestructura de dicha red.

#### **3.2.2.2. Alcance**

Administrar y controlar la red definida por software, por medio del ingreso de tráfico de datos y de reglas de control, dando paso a la creación de una infraestructura de red definida y a la monitorización de los equipos.

#### **3.2.2.3. Perspectiva del producto**

La perspectiva que se tiene del producto es la instalación de un controlador para la red SDN, que se apto para su administración y control, el cual brinde funciones de monitoreo de acuerdo a los requerimientos de la red misma.

#### **3.2.2.4. Funciones del producto**

El controlador de la red deberá cumplir con las siguientes funciones:

- Permitir el ingreso de tráfico de datos de forma manual, de acuerdo a los requerimientos y peticiones de los clientes.
- Permitir la conexión directa al switch virtual por medio de un puerto específico.
- Determinar los equipos que forman la red SDN y las características de los mismos.
- Determinar la conexión y desconexión de un equipo que forme parte de la red.
- Realizar el descubrimiento de la topología de la red SDN
- Permitir el desarrollo de aplicaciones para las pruebas de funcionalidad de la red SDN.

#### **3.2.2.5. Características de los usuarios**

Este software debe presentar características de nivel superior, ya que los usuarios de la red serán el personal de administración de redes, las cuales tiene experiencia en el manejo de las mismas, pero no de la infraestructura en su totalidad.

#### **3.2.2.6. Limitaciones**

Las limitaciones del software controlador en la red SDN son las siguientes:

- Se cuenta con un solo servidor para el desarrollo de la red SDN, por lo cual los clientes se desarrollaran como máquinas virtuales de un solo switch virtual.
- El consumo de recursos del servidor es limitado
- El acceso al servidor se realiza desde dentro del campus universitario ubicado en la ciudadela el Olivo.

#### **3.2.2.7. Suposiciones y dependencias**

En el software controlador de la red SDN se supone lo siguiente:

- El controlador debe ser compatible con el sistema operativo Linux, ya que sobre este sistema operativo se va desarrollar la infraestructura de la red.

- El servidor donde está alojada la red siempre deberá tener conexión a Internet.
- La dirección IP asignada a la máquina virtual del controlador es privada y parte de toda la infraestructura de red de la Universidad Técnica del Norte.
- El acceso del controlador a internet lo realizara por medio de una interfaz virtual conectada al switch virtual, el cual estará conectado por medio de un puente a la interfaz física del servidor.

### 3.2.2.8. Requisitos comunes de las interfaces

En esta sección se detallan los requisitos de las interfaces de acuerdo a las necesidades de software y hardware.

- **Interfaces de usuario:** el software que se implementa como controlador debe tener una interfaz web, en la que se pueda visualizar la red y sus características.
- **Interfaces de hardware:** el servidor en el que se implementa la red SDN consta de una sola interfaz física, la cual es utilizada para la conexión a internet del servidor y como acceso a la infraestructura de red general de la Universidad.
- **Interfaces de software:** estas interfaces son creadas y determinadas por el switch virtual, para lo cual el controlador será asignado a una de estas interfaces virtuales.
- **Restricción de memoria:** el software debe consumir una mínima cantidad de la capacidad de memoria del servidor físico en el que instala la red.

### 3.2.2.9. Requisitos funcionales

En la tabla 3, se describe los requisitos, para la elección del software del controlador de la red SDN en la red SDN, la prioridad esta designada por números del 1 al 3, en donde 3 es la máxima y 1 la mínima, al final se calcula un valor total, que permita determinar el controlador más adecuado para la implementación.

**Tabla 3.** Requerimientos Funcionales

| Nº de Requisito | Nombre              | Característica  | Descripción   | Prioridad |     |        |            |
|-----------------|---------------------|---|---|-----------|-----|--------|------------|
|                 |                     |   |   | NOX       | POX | Beacon | Floodlight |
| REQ 01          | OpenFlow            | Soporte OpenFlow en versión 1.0                       | El controlador debe ser parte de OpenFlow.  | 3         | 3   | 3      | 3          |
| REQ 02          | Plataforma          | Soporte en OpenStack                                  | El controlador debe permitir software libre   | 0         | 0   | 0      | 3          |
| REQ 03          | Virtualización      | Permitir los softwares virtuales Minimet y OpenSwitch | El controlador se debe conectar con los softwares de virtualización por medio de puertos específicos.               | 3         | 3   | 3      | 3          |
| REQ 04          | Sistemas Operativos | Soportar varios sistemas Operativos                   | El controlador debe soportar los sistemas operativos de software libre y licenciado (Linux, Mac, Windows y Android) | 1         | 2   | 3      | 2          |
| REQ 05          | REST API            | Arquitectura de desarrollo web apoyada en HTTP        | El controlador debe poseer REST API, para el desarrollo de la red   | 0         | 0   | 0      | 3          |
| REQ 06          | Bucles              | Repeticiones de paquetes de datos                     | El controlador debe manejar los bucles de datos y tratar de evitarlos   | 0         | 0   | 0      | 3          |
| REQ 07          | Multiprocesos       | Ejecución de uno o más procesos                       | El controlador debe ser capaz de manejar varios procesos ala ves.   | 3         | 0   | 3      | 3          |
| <b>TOTAL</b>    |                     |   |   | 10        | 8   | 12     | 17         |

**Fuente:** Elaborado por el autor

### 3.2.2.10. Requisitos no funcionales

En la Tabla 4 se describen los requisitos no funcionales que los controladores deben cumplir, para elegir uno de ellos como el más adecuado para la red.

**Tabla 4.** Requerimientos no Funcionales

| N <sup>o</sup> de Requisito | Nombre           | Característica  | Descripción   | Prioridad |           |           |            |
|-----------------------------|------------------|---|---|-----------|-----------|-----------|------------|
|                             |                  |   |   | NOX       | POX       | Beacon    | Floodlight |
| REQ 08                      | Java             | Lenguaje de programación                              | El controlador debe estar desarrollado en Java  | 0         | 0         | 3         | 3          |
| REQ 09                      | Licencia         | Licencia de software libre                            | El controlador debe manejar código abierto y software libre GPL y FOSS                              | 1         | 1         | 3         | 3          |
| REQ 10                      | Interfaz Gráfica | Permite la comunicación entre el usuario y la máquina | El controlador debe poseer una interfaz gráfica que permita su fácil utilización (Python, QT4, Web) | 2         | 3         | 1         | 1          |
| REQ 11                      | Instalación      | Transferencia de programas a un equipo                | El controlador debe ser fácil de instalar   | 1         | 2         | 1         | 3          |
| REQ 12                      | Simplicidad      | Facilidad de manejo                                   | El controlador debe ser fácil de manipular  | 2         | 2         | 3         | 3          |
| REQ 13                      | Comunidad Activa | Personas que actualizan y modifican                   | El controlador debe ser actualizado y tener suficiente documentación                                | 1         | 1         | 2         | 3          |
| REQ 14                      | Mercado          | Tiempo que está en proceso                            | El controlador debe ser de los más actuales   | 1         | 3         | 1         | 3          |
| <b>TOTAL</b>                |                  |   |   | <b>8</b>  | <b>12</b> | <b>14</b> | <b>19</b>  |

**Fuente:** elaborado por el autor

### 3.2.2.11. Requisitos de rendimiento

El controlador se instalará en una máquina virtual, el cual debe monitorear toda la red SDN, los equipos que se conectan a la misma y el tráfico de datos que se genere, sin afectar su funcionamiento.

### 3.2.2.12. Justificación

Floodlight presenta las mejores funciones y características en base a su funcionamiento y a los servicios que brinda, ya que tiene varios beneficios como su funcionamiento con switches físicos y virtuales, presenta una interfaz web propia, posee REST API, tiene soporte para varios siste-

mas operativos, su instalación es simple, es compatible con versiones actuales de Java, su desarrollo es de código abierto y tiene un gran documentación y desarrolladores activos, además presenta la mayor prioridad de los requisitos tanto funcionales como no funcionales para la implementación de la red SDN, lo cual genera una instalación simple, pero robusta en software, floodlight garantiza ser el software optimo y eficiente para cubrir con las necesidades de la infraestructura de red SDN.

### **3.2.2.13. Características**

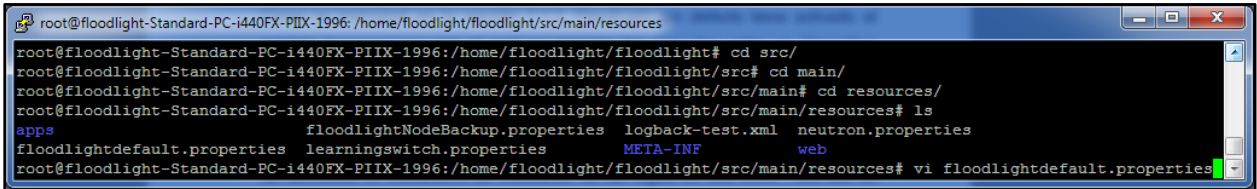
Para el controlador de la red SDN se eligió Floodlight, debido a que después de realizar las configuraciones y pruebas con todos los controladores propuestos, se determinó que Floodlight es el más adecuado para la infraestructura de red, basándose en las siguientes características:

- Adaptabilidad, está escrito en Java por lo que permite el desarrollo de aplicaciones y acoplar diferentes softwares, además de que su instalación y ejecución es relativamente fácil.
- Funcionalidad, se desarrolla en switch físicos y virtuales, además maneja dos tipos de flujos: reactivo y proactivo.
- Documentación, posee una amplia información acerca de su desarrollo, configuración y aplicación del mismo, esta soportado por una gran comunidad de desarrolladores y aportes sobre su implementación.

## **3.3. DEFINICIÓN DE REGLAS DE CONTROL**

En la elección del software se seleccionó a Floodlight como controlador de la red, el cual tiene activado automáticamente en su configuración el módulo Forwarding, que se encarga del reenvío de datos entre dispositivos manejando flujos reactivos, de forma inmediata al encender el

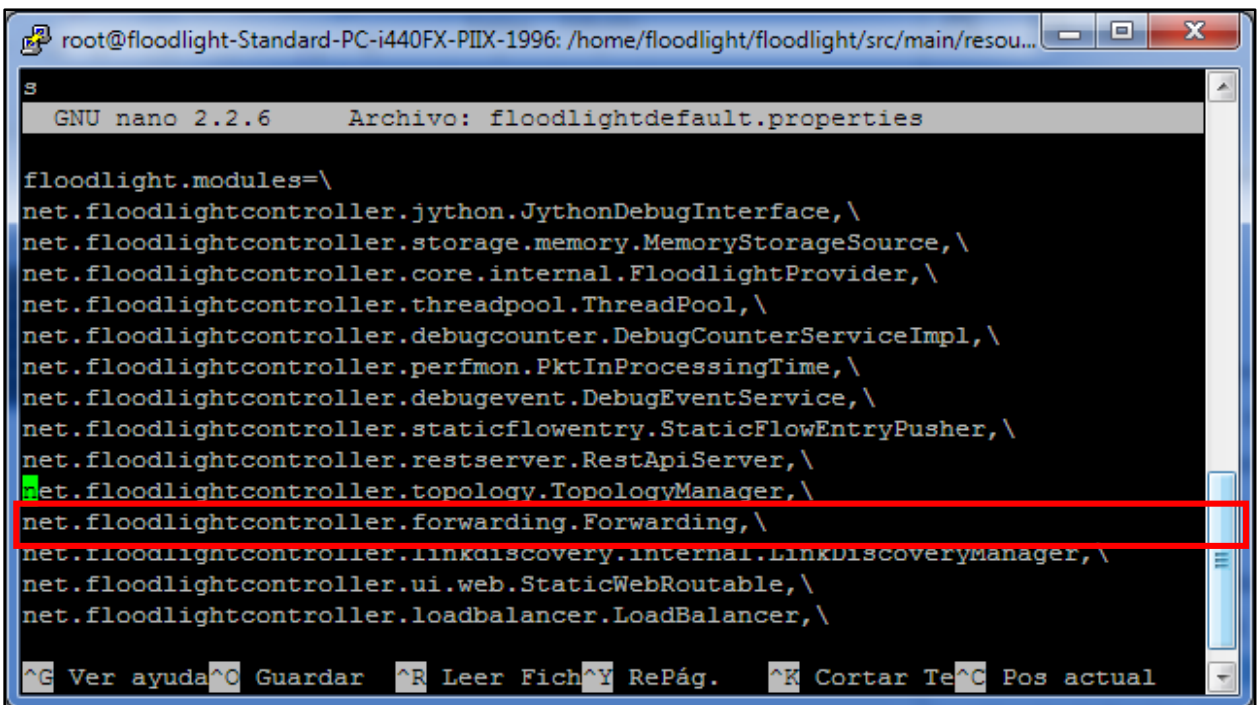
controlador, para la realización del proyecto se va a utilizar flujos estáticos, por lo que es necesario cambiar la configuración del módulo Forwarding, en la Figura 34 se elimina la línea donde se llama dicho módulo.



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight/src/main/resources
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight/src# cd main/
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight/src/main# cd resources/
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight/src/main/resources# ls
apps                floodlightNodeBackup.properties  logback-test.xml  neutron.properties
floodlightdefault.properties  learningswitch.properties        META-INF          web
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight/src/main/resources# vi floodlightdefault.properties
```

**Figura 34.** Ingreso al Archivo floodlightdefault.properties  
**Fuente:** Máquina virtual controlador Floodlight

Eliminar *net.floodlightcontroller.forwarding.Forwarding* y verificar la línea: *net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher*, como en la Figura 35.



```
GNU nano 2.2.6 Archivo: floodlightdefault.properties

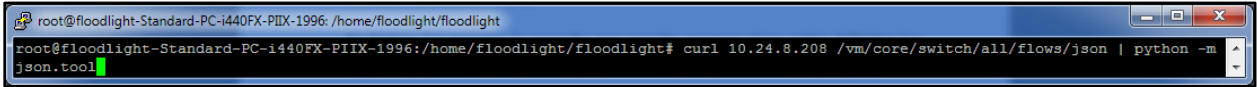
floodlight.modules=\
net.floodlightcontroller.jython.JythonDebugInterface, \
net.floodlightcontroller.storage.memory.MemoryStorageSource, \
net.floodlightcontroller.core.internal.FloodlightProvider, \
net.floodlightcontroller.threadpool.ThreadPool, \
net.floodlightcontroller.debugcounter.DebugCounterServiceImpl, \
net.floodlightcontroller.perfmon.PktInProcessingTime, \
net.floodlightcontroller.deugevent.DebugEventService, \
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher, \
net.floodlightcontroller.restserver.RestApiServer, \
net.floodlightcontroller.topology.TopologyManager, \
net.floodlightcontroller.forwarding.Forwarding, \
net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager, \
net.floodlightcontroller.ui.web.StaticWebRoutable, \
net.floodlightcontroller.loadbalancer.LoadBalancer, \

^G Ver ayuda ^C Guardar ^R Leer Fich ^Y RePág. ^K Cortar Te ^C Pos actual
```

**Figura 35.** Modificación de archivo floodlightdefault.properties  
**Fuente:** Máquina virtual controlador Floodlight

### 3.3.1. Estadísticas de flujo

Para la transferencia de archivos con formato URL se utiliza el comando *curl*, para conocer los flujos, su ubicación y las estadísticas del mismo se utiliza el comando, mostrado en la Figura 36:

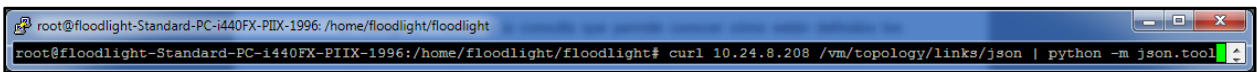


```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996:/home/floodlight/floodlight# curl 10.24.8.208 /vm/core/switch/all/flows/json | python -m json.tool
```

**Figura 36.** Comando para ver las características de los flujos  
**Fuente:** Máquina virtual controlador Floodlight

### 3.3.2. Información de la topología

Para conocer la información de los distintos switches conectados se utiliza el comando de la Figura 37:



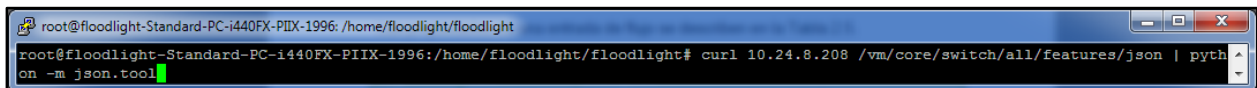
```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996:/home/floodlight/floodlight# curl 10.24.8.208 /vm/topology/links/json | python -m json.tool
```

**Figura 37.** Comando para ver los switches conctados a la red  
**Fuente:** Máquina virtual controlador Floodlight

Para obtener información de switches no OpenFlow se anexa el modulo `net.floodlightcontroller.topology.TopologyManager`.

### 3.3.3. Características del switch

Para conocer las características del switch se usa el comando de la Figura 38:



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996:/home/floodlight/floodlight# curl 10.24.8.208 /vm/core/switch/all/features/json | python -m json.tool
```

**Figura 38.** Comando para verlas características del switch  
**Fuente:** Máquina virtual controlador Floodlight

### 3.3.4. Flujos estáticos e ingreso de los mismos

Los flujos estáticos permiten crear la tabla de flujo de forma manual, debido a que se insertan de acuerdo a las necesidades de los usuarios, los flujos son insertados por el controlador antes de que los paquetes de datos lleguen, una vez que los paquetes llegan al switch no son enviados al controlador para ser evaluados, debido a que coinciden con el flujo insertado.

En la Tabla 4 se detallan ciertas características de una tabla de flujo



**Tabla 5.** Propiedades tabla de flujo

| Key           | Valor           | Descripción  |
|---------------|-----------------|--|
| Switch        | <ID_switch>     | Formato hexadecimal  |
| Name          | <cadena>        | Nombre de la entrada de flujo, es única                                      |
| Action        | <key>=<valor>   | Si no hay acción se descarta el paquete, se usa coma para múltiples acciones |
| Priority      | <número>        | Puede ser 0-32767, por defecto 32767   |
| Active        | <booleano>      | Regla activa o no, true o false  |
| ingress-port  | <número>        | Puerto por el que se recibe paquetes   |
| src-mac       | <dirección_mac> | Origen   |
| dst-mac       | <dirección_mac> | Destino  |
| vlan-id       | <número>        | ID vlan  |
| vlan-priority | <número>        | Prioridad de la vlan   |
| ether-type    | <número>        | Tipo de protocolo Ethernet   |
| Protocol      | <número>        | Protocolo implementado   |
| src-ip        | <dirección_ip>  | Origen   |
| dst-ip        | <dirección_ip>  | Destino  |
| src-port      | <número>        | TCP/UDP origen   |
| dst-port      | <número>        | TCP/UDP destino  |

**Fuente:** Diana Morillo. (2014). *Implementación de un prototipo de una Red Definida por Software (SDN) empleando una solución basada en software.*

En el campo de acción existen varias alternativas, las cuales se presentan en la Tabla 5.

**Tabla 6.** Opciones campo acción

| Acción                   | Valor                       | Descripción  |
|--------------------------|-----------------------------|--|
| <b>Output</b>            | <número>                    | Puerto de salida del flujo   |
|                          | All                         | All: envío a todos los dispositivos  |
|                          | Cotroller                   | Controller: envía por puerto conectado a controlador                             |
|                          | Local                       | Local: envío local   |
|                          | Ingress-port<br>Normalflood | Ingress-port: puerto de ingreso del flujo<br>Flood: inundación todos los puertos |
| <b>Enqueue</b>           | <número>:<número>           | Primero el puerto y el segundo es el ID de la cola                               |
| <b>set-vlan-id</b>       | <número>                    | Valor ID de vlan   |
| <b>set-vlan-priority</b> | <número>                    | Prioridad de vlan  |
| <b>set-src-mac</b>       | <dirección_mac>             | Origen   |
| <b>set-dst-mac</b>       | <dirección_mac>             | Destino  |
| <b>set-tos-bits</b>      | <número>                    | Bits de ToS  |
| <b>set-src-ip</b>        | <dirección_ip>              | Origen   |
| <b>set-dst-ip</b>        | <dirección_ip>              | Destino  |
| <b>set-src-port</b>      | <número>                    | Puerto de capa transporte origen   |

**Fuente:** Diana Morillo. (2014). *Implementación de un prototipo de una Red Definida por Software (SDN) empleando una solución basada en software.*

Para insertar un flujo estático es necesario poner los campos: Switch, name, priority, ingress-port, de la siguiente forma:

```
curl -d '{"switch": "ID switch", "name": "nombre del flujo", "priority": "prioridad", "ingres-port": "número puerto", "active": "Booleano", "actions": "Acciones"}'
```

<http://<IP controlador>:8080/wm/staticflowentrypusher/json>

### 3.3.5. Eliminación de flujos estáticos

Para eliminar un flujo estático es necesario poner solo el campo name de la siguiente forma:

```
curl -X DELETE -d '{"name": "nombre flujo"}'
```

<http://<IP controlador>:8080/wm/staticflowentrypusher/json>

Para la inserción de flujos estáticos se pueden utilizar distintos argumentos, los cuales se describen en la Tabla 6.

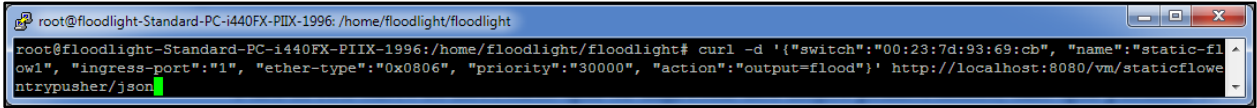
**Tabla 7.** Argumentos de flujos estáticos

| URI   | Descripción              | Argumento                |
|---|--------------------------|--------------------------|
| /wm/staticflowentrypusher/json                | Agrega y elimina         | HTTP POST<br>HTTP DELETE |
| /wm/staticflowentrypusher/list/<switch>/json  | Lista todos los flujos   | DPID del switch          |
| /wm/staticflowentrypusher/clear/<switch>/json | Elimina todos los flujos | DPID del switch          |

**Fuente:** Diana Morillo. (2014). *Implementación de un prototipo de una Red Definida por Software (SDN) empleando una solución basada en software.*

Para la red SDN es necesario que se ingresen dos flujos primordiales, uno que permita el tráfico ARP y otro para la conexión de cada cliente a través del puerto, para el flujo de ARP el campo ether-type, especifica el protocolo ARP en hexadecimal y en el campo action se realiza un flooding del flujo por los puertos.

El ingreso del flujo estático para el tráfico ARP en los switches se muestra en la Figura 39.



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996:/home/floodlight/floodlight# curl -d '{"switch":"00:23:7d:93:69:cb", "name":"static-flow1", "ingress-port":"1", "ether-type":"0x0806", "priority":"30000", "action":"output=flood"}' http://localhost:8080/vm/staticflowentrypusher/json
```

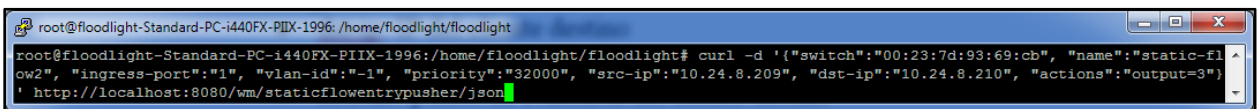
**Figura 39.** Inserción de flujo para tráfico ARP  
**Fuente:** Máquina Virtual Floodlight

Los flujos para la comunicación entre clientes, es necesario definir dos flujos uno en cada dirección, es decir que son bidireccionales, en ellos se incluye la dirección IP de cada uno de los host, por lo que se debe ingresar dos flujos para cada par de host de la red.

Los campos que se utilizan en la inserción de flujos estáticos para los host son:

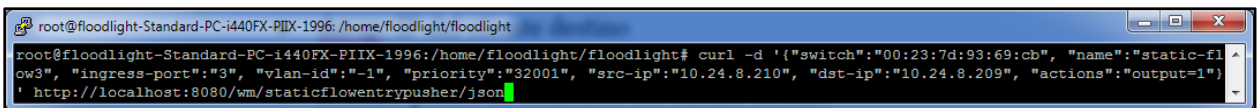
- switch: identifica al switch
- ingress-port: puerto de conexión del cliente origen
- vlan-id: tiene el valor de -1 para Forwarding
- priority: diferente en cada flujo
- src-ip: IP del cliente origen
- dst-ip: IP del cliente destino
- actions: puerto de conexión del cliente destino

El ingreso del flujo estático para la comunicación entre pc se muestra en la Figura 40 y 41.



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996:/home/floodlight/floodlight# curl -d '{"switch":"00:23:7d:93:69:cb", "name":"static-flow2", "ingress-port":"1", "vlan-id":"-1", "priority":"32000", "src-ip":"10.24.8.209", "dst-ip":"10.24.8.210", "actions":{"output":3}}' http://localhost:8080/vm/staticflowentrypusher/json
```

**Figura 40.** Ingreso de flujo cliente1  
**Fuente:** Máquina Virtual Floodlight



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996:/home/floodlight/floodlight# curl -d '{"switch":"00:23:7d:93:69:cb", "name":"static-flow3", "ingress-port":"3", "vlan-id":"-1", "priority":"32001", "src-ip":"10.24.8.210", "dst-ip":"10.24.8.209", "actions":{"output":1}}' http://localhost:8080/vm/staticflowentrypusher/json
```

**Figura 41.** Ingreso flujo cliente2  
**Fuente:** Máquina Virtual Floodlight

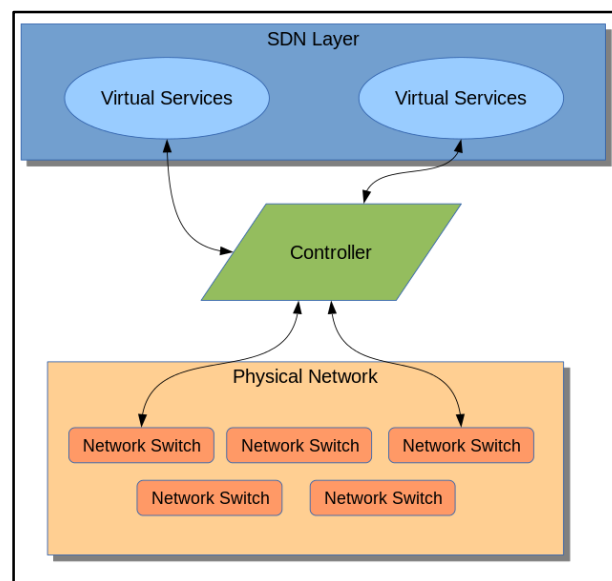
## 3.4. DESARROLLO DE LA APLICACIÓN PARA NAC

### 3.4.1. Introducción

Para que exista comunicación en toda la red, es necesario contar con dos interfaces las cuales dan paso a la comunicación tanto a un nivel superior como inferior de partes específicas de la red, las interfaces son northbound y southbound.

Northbound es una interfaz que por medio de las aplicaciones de control permite gestionar la red, en la parte alta del stack de SDN, además ofrece la facilidad de configurar una sola aplicación para las funciones de seguridad, enrutamiento y calidad de servicio.

Southbound es una interfaz que permite la comunicación con la parte inferior del stack SDN, además verifica la conducta de los switches, la estructura se muestra en la Figura 42.



**Figura 42.** Aplicación Northbound y Southbound

**Fuente:** Li-Ji Hong. (2014). *Showipprotocols*. Obtenido de: <http://goo.gl/QOiWAQ>

La aplicación para NAC que se desarrollara para el presente proyecto está basada en el modelo cliente-servidor, por medio de la cual se controla los dispositivos que acceden a la red, utilizando reglas en las que si el dispositivo cuenta con seguridad se utiliza la acción *ALLOW* y si no posee ninguna seguridad se utiliza *DENY*.

La aplicación *Cliente* se encarga de comunicar un dispositivo que accede a la red con el servidor, dicha comunicación la realiza por medio de un puerto TCP, la aplicación permite definir si un equipo tiene establecido algún mecanismo de seguridad, en la aplicación desarrollada para el proyecto se detectará un archivo que permita verificar las políticas de seguridad.

La aplicación *ServidorNAC* está a la espera de las peticiones realizadas por el *Cliente*, cuando se realiza la conexión, dicho servidor envía las reglas al controlador, para permitir o denegar el acceso a un dispositivo.

### 3.4.2. Diseño del software

Para el diseño se debe considerar los siguientes criterios:

**Cliente:** se deben considerar las siguientes acciones:

- Establecer la comunicación con el servidor
- Mecanismo para detectar el archivo que indica el cumplimiento de las reglas
- Envío de un aviso al servidor para determinar la existencia de seguridad

**ServidorNAC:** se deben considerar las siguientes acciones:

- Establecer la comunicación con el cliente
- Establecer la comunicación con el controlador
- Diferencia entre equipos de conectividad y usuarios, debido a que las reglas son implementadas en base a requerimientos de usuario
- Determinar un switch principal en caso de implementar el switch virtual en más equipos
- Evitar flujos redundantes, evitando el ingreso de reglas repetidas
- Eliminar reglas de equipos que se desconecten de la red

- Si un equipo se conecta al switch principal se define las reglas solo de dicho switch, pero si se conecta a un switch secundario se determinan las reglas del principal y del secundario.
- Capacidad de ensamblar las reglas, ya que deben estar serializados y concatenados.

### Comunicación servidor-cliente:

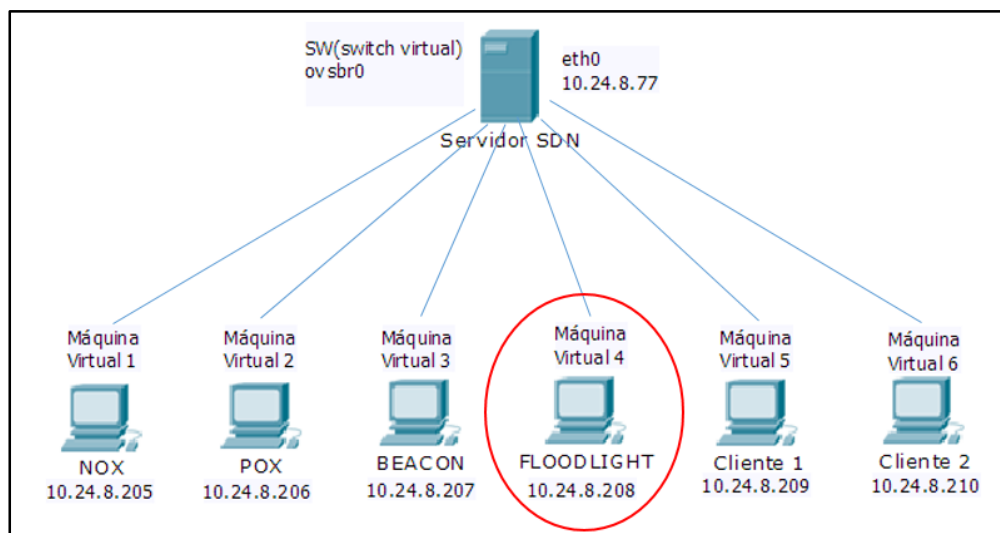
Se utiliza el protocolo TCP para la comunicación entre el servidor y el cliente, evitando las retransmisiones y brindando una transmisión de datos segura, se emplea el puerto 5000.

### Controlador:

El controlador Floodlight cuenta con dos módulos específicos que son Firewall y Forwarding, los cuales se encargan de las características de manejo y reenvío de datos, además de que se insertaran las reglas definidas e iniciales.

### 3.4.3. Componentes del software y del hardware

Para el desarrollo del proyecto en la parte de hardware se cuenta con un equipo servidor, del cual se describen las características en la Tabla 1. La infraestructura sobre la cual se desarrollara la aplicación es la que se muestra en la Figura 43.



**Figura 43.** Infraestructura de red  
Fuente: Criterio de diseño

Para la parte del software se utiliza Eclipse como plataforma y Avior para el servidor, tomando en cuenta los requisitos de la aplicación NAC.

### 3.4.4. Diagrama de flujo

Para la aplicación Cliente, es importante determinar un archivo en el que se verifica si un equipo tiene o no algún mecanismo de seguridad, para buscar el archivo es necesario identificar la ruta de ubicación del mismo.

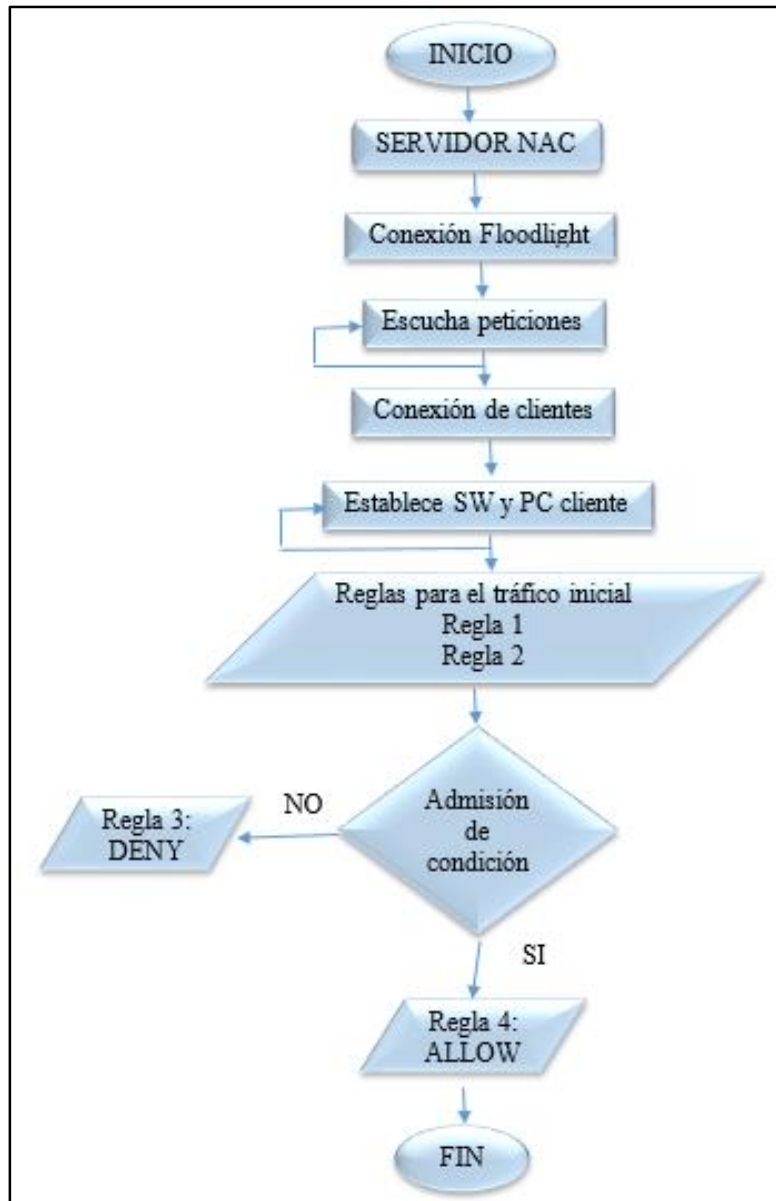
Se envía una condición con un valor 1 o 0, en los cuales el 1 indica q hay seguridad y 0 no existe seguridad.

El diagrama de flujo utilizado se muestra en la Figura 44.



**Figura 44.** Diagrama de flujo aplicación cliente  
**Fuente:** Criterio de diseño del proyecto

Para la aplicación del Servidor NAC, en el diagrama de flujo se muestra la Regla 1, la cual permite el tráfico ARP, la Regla 2 es la encargada de la conexión entre cliente y servidor, las Reglas 3 y 4 son las encargadas de permitir o denegar el acceso de un cliente a la red, el diagrama se muestra en la Figura 45.



**Figura 45.** Diagrama de flujo aplicación ServidorNAC  
**Fuente:** Criterio de diseño



### 3.5.5. Módulos Floodlight empleados

Los módulos utilizados son Firewall y Forwarding, de los muchos que ofrece Floodlight, debido a que permiten el envío de paquetes, por medio de las acciones ALLOW y DENY.

#### **FORWARDING**

Es el modulo encargado del envío de paquetes entre dispositivos de origen y de destinos, dichos dispositivos se diferencian por el identificador de servicio, las dependencias de los servicios son:

- IDeviceService
- IFloodlightProviderService
- IRestApiService
- IRoutingService
- ITopologyService
- ICounterStoreService

Dicho modulo se encarga de determinar una ruta para el flujo, está activado de forma automática, si se ha desactivado anteriormente, es necesario verificar el archivo */src/main/resources/floodlightdefault.properties* en la línea *net.floodlightcontroller.forwarding.Forwarding*.  
(floodlight.atlassian.net, 2015)

#### **FIREWALL**

Es un módulo que se encarga del cumplimiento de las Acces-list, por medio de la adherencia de flujos, las cuales son determinadas de acuerdo a si se permite o no la entrada de un flujo de datos, las reglas de acceso se ordenan de acuerdo a la prioridad de las mismas en el momento de su creación, dicho modulo maneja los métodos de HTTP: GET, POST y DELETE.

Si un paquete del tipo *Packet-In* ingresa, se verifica la tabla de flujo y si existe una coincidencia, se determina una prioridad y se almacena la regla de permitir o denegar en un objeto, se transmite el paquete hasta el módulo Forwarding para el reenvío del paquete, en el proyecto que se desarrolla se utiliza una entrada de reenvío si la acción es permitir y se descarta el paquete si la acción es denegar. (floodlight.atlassian.net, 2015)

### 3.5.6. Código desarrollado

Para el desarrollo de las aplicaciones es necesario instalar el software AVIOR, del cual se modificaron algunas clases, para cumplir con los requerimientos de la aplicación NAC, los cambios realizados en cada clase se describen a continuación:

- **Rule.java:** es una clase de Avior y se encuentra en /floodlight/Avior-master/src/model/tools/firewall, se encarga de establecer los parámetros de las reglas de tráfico, se adaptó el método *Rule()* para que se incluya el ID del switch al inicio de cada regla
- **FloodlightProvider.java:** es una clase de Avior, se encuentra en /floodlight/Avior-master/src/controller/floodlightprovider, muestra los switches de la red y las reglas insertadas, se modificó para que se haga la llamada al método *getDeviceSummaries()* y permita observar los clientes conectados a la red.
- **FirewallPusher.java:** es una clase de Avior, se encuentra en /floodlight/Avior-master/src/controller/tools/firewall/push, se encarga de establecer las funciones *push()* y *remove()*, las cuales permiten adherir o quitar flujos.
- **Firewall.java:** es una clase del controlador Floodlight, se encuentra en /floodlight/src/Main/java/net/floodlightcontroller/firewall, se encarga de establecer el funcionamiento del módulo en el controlador, se modificó para que si un tráfico no coincide con ninguna regla se realice una acción.

- **Switch.java:** es una clase de Avior, ubicada en /floodlight/ Avior-master/src/model/overview, se encarga de establecer las características de los switches de la red.
- **DevicesSummary.java:** es una clase de Avior, ubicada en /floodlight/ Avior-master/src/model/overview, se encarga de establecer las características de los clientes de la red.

### Aplicación ServidorNAC

La aplicación que se encarga de articular y enviar las reglas al controlador, dependiendo de los requerimientos enviados por el cliente es la de *ServidorNAC.java*, la cual está ubicada en /src/servidor/control, consta de 4 métodos principales *loadSwitches( )*, *loadDevices( )*, *Filtro( )*, *conexionCliente( )*, los dos primeros métodos son realizados de acuerdo a un hilo de ejecución, dicho hilo se encuentra verificando los nuevos equipos que se conectan a la red, al mismo tiempo la aplicación posee otro hilo de ejecución que está pendiente de las peticiones de conexión, creando un nuevo hilo para cada cliente que solicite una conexión.

En la Figura 46 se muestra la clase Hilo, que es la encargada de recopilar la información de los switches y clientes, las líneas de estructura de dicha clase son:

- Línea 1: definición de clase
- Línea 5: creación de objeto *Escuchar*
- Línea 6: inicio del objeto *Escuchar* por medio del método *start( )*
- Línea 7: definición de lazo
- Línea 8: carga de los datos del switch por medio de *loadSwitchData*
- Línea 9: carga de los datos de los clientes por medio de *loadDeviceData*
- Línea 10: llamada al método *sleep( )*

```

1:     private static class Hilo implements Runnable {
2:         public void run()
3:         {
4:             try{
5:                 Thread s = new Thread (new Escuchar ());
6:                 s.start();
7:                 while(true) {
8:                     loadSwitchData();
9:                     loadDeviceData();
10:                    Thread.sleep(5000);
11:                }
12:            }catch(InterruptedException e)
13:            }
14:        }

```

**Figura 46.** Código de la clase Hilo  
**Fuente:** Máquina virtual controlador Floodlight

Para el método escuchar se crea un método que permita la conexión del cliente como en la Figura 47.

```

1:     private static class Escuchar implements Runnable {
2:         public void run()
3:         {
4:             conexionCliente();
5:         }}

```

**Figura 47.** Código clase Escuchar  
**Fuente:** Máquina virtual controlador Floodlight

La clase principal de la aplicación ServidorNAC, establece la comunicación con el controlador, además controla el estado del Módulo Firewall, además crea un nuevo objeto Hilo, como ya se había menciona para la inicialización de los anteriores, se desarrolla como en la Figura 48.

```

1: private static void main (String[] args) throws InterruptedException {
2: FloodlightProvider.setIP(controllerIP);
3: String status = "";
4: try{
5:     if (!FirewallJSON.isEnabled()) {
6:         try {
7:             status = FirewallJSON.enable(true);
8:             System.out.println(status);
9:         } catch (JSONException e) {
10:            e.printStackTrace();
11:        }
12:    }
13: }
14: Thread t = new Thread (new Hilo())
15: t.start (); }

```

**Figura 48.** Código clase principal ServidorNAC  
**Fuente:** Máquina virtual controlador Floodlight

El método loadSwitches, se detecten los switches por medio de los paquetes LLDP, si se tiene más de dos switches virtuales, es necesario hacer una diferencia entre el que está conectado al controlador y el resto de los mismos, la estructura de este método se desarrolló como en la Figura 49.

```

1: public static void loadSwitchData() {
2: switches = Floodlightprovider.getSwitches(true);
3: for(Switch sw : switches)
4: {
5:     if (sw.getDpid().equals(macAddress))
6:         principal=sw;
7: }
8: }

```

**Figura 49.** Código método loadSwitchData  
**Fuente:** Máquina virtual controlador Floodlight

El método Filtro contiene las características para la inserción de reglas especificadas en la clase Rule.java, también almacena una variable con la información del switch actual, en el cual se procesan los datos, la estructura de dicho método se desarrolló como se muestra en la Figura 50.

```

1: public static void Filtro(String in_port, String dl_src, String dl_dst, String dl_type,
String nw_src_prefix, String nw_src_maskbits, String nw_dst_prefix, String
nw_dst_maskbits, String nw_proto, String tp_src, String tp_dst, String action, String
priority)
2: {
3:     if(currSwitch != null) {
4:         Rule rule = new Rule(currSwitch.getDpid());
5:         rule.setIn_port(in_port);
6:         rule.Dl_src(dl_src);
7:         rule.Dl_dst(dl_dst);
8:         rule.Dl_type(dl_type);
9:         if(!nw_src_prefix.equals(""))
10:            rule.setNw_src_prefix(nw_src_prefix);
11:         if(!nw_src_maskbits.equals(""))
12:            rule.setNw_src_maskbits(nw_src_maskbits);
13:         rule.setNw_proto(nw_proto);
14:         rule.setTp_src(tp_src);
15:         rule.setTp_dst(tp_dst);
16:         rule.setAction(action);
17:         rule.setPriority(priority);
18:
19:         if (rule.getDpid() != null) {
20:             String response;
21:             try {
22:                 response = FirewallPusher.push(rule);
23:                 if(response.equals("Regla insertada en Switch")) {
24:                     System.out.println("Regla insertada exitosamente");
25:                 }
26:             }catch (IOException | JSONException el) {
27:                 System.out.println("Problema al insertar la regla");
28:                 el.printStackTrace();
29:             }
30:             }else {
31:                 System.out.println("No existe ninguna regla para insertar");
32:             }
33:         }

```

**Figura 50.** Código método Filtro  
**Fuente:** Máquina virtual controlador Floodlight

Para el método loadDeviceData , se elimina los dispositivos que se desconectan de la red al igual que sus reglas de tráfico, para lo cual se verifica el que el puerto al que estaba conectado un equipo sea cero y la existencia de reglas para dicho dispositivo, utilizando su dirección MAC, en este método también se evita la inserción de reglas replicadas comparando la MAC con la lista *obj*, es necesario hacer diferencia entre los switches en el caso de existir más de uno, declarando como principal al que contiene el controlador y secundarios a los demás, por lo que el ingreso de

las reglas es diferente para cada uno, el desarrollo de este método se muestra en la Figura 51 y 52.

```
1: public static void loadDeviceData()
2: {
3:     devices = FloodlightProvider.getDevices(true);
4:     for(Switch lista : switches)
5:     {
6:         reglas = FloodlightProvider.getRules(lista.getDpid(), true);
7:         for(DeviceSummary aux : devices)
8:         {
9:             if(aux.getSwitchPort() == 0 && !reglas.isEmpty())
10:            {
11:                try {
12:                    System.out.println(reglas.size());
13:                    for(Rule rl : reglas)
14:                    {
15:                        if(!rl.getDl_src().equals(""))
16:                        {
17:                            if(!aux.getMacAddress().equals(""))
18:                            {
19:                                if(rl.getDl_src().toLowerCase().equals(aux.getMacAddress()))
20:                                {
21:                                    { System.out.println(MAC.remove(aux.getMacAddress()));
22:                                    String response = FirewallPusher.remove(rl);
23:                                    System.out.println(response);
24:                                    return;
25:                                }
26:                            }
27:                        }
28:                    }
29:                } catch (TOException | JSONException e) {
30:                    e.printStackTrace();
31:                } catch (NullPointerException e)
32:                {
33:                    System.out.println(e.getMessage());
34:                }
35:            }
36:        }
37:        for(DeviceSummary obj : devices)
38:        {
39:            int repeticion = 0;
40:            for(String mac : MAC)
41:            {
42:                if (obj != null)
43:                if(obj.getMacAddress().equals(mac))
44:                    repeticion++;
45:            }
46:            if(repeticion == 0)
47:            {
48:                for(Switch equipo : Switches)
```

**Figura 51.** Código método loadDeviceData  
**Fuente:** Máquina virtual controlador Floodlight

```

43:         for(Switch equipo : Switches)
44:         }
45:         if(equipo.getDpid().equals(obj.getAttachedSwitch()))
46:         {
47:             currSwitch = equipo;
48:             if(obj.getIpv4() != switches.contains(obj) && !obj.getIpv4().equals(controllerIP))
49:             {
50:                 MAC.add(obj.getMacAddress());
51:                 if(obj.getAttachedSwitch().equals(macAddress))
52:                 { //Reglas Switch
53:                     Filtro("", "", "", "2054", "", "", "", "", "", "", "", "", "");
54:                     Filtro("", "", "", "2048", "", "", "", "", "TCP", "5000", "", "", "");
55:                     Filtro("", "", "", "2048", "", "", "", "", "TCP", "", "5000", "", "");
56:                 }

```

**Figura 52.** Continuación código método loadDeviceDat  
**Fuente:** Máquina virtual controlador Floodlight

Para el método `conexionCliente`, es necesario que se realice la comunicación con el cliente para que se permita o deniegue su conexión, dicha conexión se realiza por el puerto 5000, para lo que se define una clase *Comunicación*, la cual recibe la información del cliente, el desarrollo de dicho método se especifica en la Figura 53, 54 y 55



```

1: public static void conexionCliente() {
2:     try{
3:         ServerSocket escucha = null ;
4:         Socket cliente = null ;
5:         escucha = new ServerSocket(50000);
6:         while (true) {
7:             cliente = escucha.accept() ;
8:             if(cliente.isConnected())
9:             {
10:                System.out.println("Cliente conectado");
11:                Thread hilocliente = new Thread(new Comunicacion(cliente)) ;
12:                hilocliente.start() ;
13:            }
14:        }
15:    }catch (Exception e) {
16:        Logger.getLogger(ServidorNAC.class.getName()).log(Level.SEVERE, null, e) ;
17:    }
18: }
19: private static class Comunicacion implements Runnable {
20:     Socket cliente ,
21:     InputStream bufferEntrada ;
22:     DataInputStream datos ;
23:     public Comunicacion(Socket cliente)
24:     {
25:         this.cliente = cliente ;
26:     }
27:     public void run() {
28:         try {
29:             DeviceSummary host = null ;
30:             System.out.println(" ");
31:             System.out.println(cliente.getInetAddress().getHostAddress());
32:             System.out.println(" ");
33:             for (DeviceSummary obj : devices)
34:                 if(obj.getIpv4() !=null)
35:                 {
36:                     if(obj.getIpv4().equals(cliente.getInetAddress().getHostAddress()))
37:                     {
38:                         System.out.println(obj.getIpv4());
39:                         host = obj ;
40:                         break ;
41:                     }
42:                 }
43:         }

```

**Figura 53.** Código método conexionCliente  
**Fuente:** Máquina virtual controlador Floodlight

```

44:      System.out.println(host.getIpv4());
45:      bufferEntrada = cliente.getInputStream();
46:      datos = new DataInputStream(bufferEntrada);
47:      Integer cadena = new Integer(datos.readInt());
48:      {
49:          Integer men=cadena;
50:          if (men == 0) {
51:              System.out.println(String.valueOf(host.getSwitchPort()));
52:              System.out.print("POSIBLE FILTRACION\n");
53:              System.out.print("Bloquear trafico entrante\n");
54:              if(host.getAttachedSwitch().equals(principal.getDpid()))
55:                  {
56:                      currSwitch = principal
57:                      Filtro("", host.getMacAddress(), "", "", host.getIpv4(), "32", "", "", "", "", "", "DENY", "");
58:                      for(Switch dispositivo : switches)
59:                          {
60:                              if(!dispositivo.getDpid().equals(principal.getDpid()))
61:                                  {
62:                                      currSwitch = dispositivo
63:                                      Filtro("", host.getMacAddress(), "", "", host.getIpv4(), "32", "", "", "", "", "", "DENY", "");
64:                                  }
65:                              }
66:                          }
67:                      else if (men == 1)
68:                          {
69:                              System.out.print("SEGURIDAD ESTABLECIDA");
70:                              System.out.print("Permitir ingreso");
71:                              if (host.getAttachedSwitch().equals(principal.getDpid()))
72:                                  {
73:                                      currSwitch = principal;
74:                                      Filtro("", host.getMacAddress(), "", "", host.getIpv4(), "32", "", "", "", "", "", "", "");
75:                                      for(Switch dispositivo : switches)
76:                                          {
77:                                              if(!dispositivo.getDpid().equals(principal.getDpid()))
78:                                                  {
79:                                                      currSwitch = dispositivo;
80:                                                      Filtro("", host.getMacAddress(), "", "", host.getIpv4(), "32", "", "", "", "", "", "", "");
81:                                                  }
82:                                              }
83:                                          }
84:                                      datos.close();
85:                                      bufferEntrada.close();
86:                                      cliente.close();

```

**Figura 54.** Continuación código método conexionCliente

**Fuente:** Máquina virtual controlador Floodlight

```

87:      }
88:      } catch (Exception e) {
89:          Logger.getLogger(ServidorNAC.class.getName()).log(Level.SEVERE, null, e);
90:      }
91:      }
92:      }
93:      }

```

**Figura 55.** Continuación código método conexionCliente

**Fuente:** Máquina virtual controlador Floodlight

## Cliente

En la clase Cliente se establece la conexión con el servidor, utilizando el puerto TCP y la dirección IP del mismo, como se había mencionado la seguridad se establece a través de un archivo, el desarrollo de dicha clase se muestra en la Figura 56.

```
1: public class Cliente {
2:     public static void main(String[] args) {
3:         try {
4:             Socket canalComunicacion = null ;
5:             OutputStream bufferSalida ;
6:             DataOutputStream datos ;
7:             File archivo = null ;
8:             FileReader fr = null ;
9:             BufferedReader br = null ;
10:            canalComunicacion = new Socket("10.24.8.208", 5000) ;
11:            bufferSalida = canalComunicacion.getOutputStream() ;
12:            datos = new DataOutputStream(bufferSalida) ;
13:            archivo = new File ("/home/floodlight/floodlight/archivo_conf") ;
14:            fr = new FileReader (archivo) ;
15:            br = new BufferedReader(fr) ;
16:            String sfichero = String.valueOf(br.readLine()) ;
17:            File fichero = new File(sfichero) ;
18:            Integer mensaje = 5 ;
19:            if (fichero.exists())
20:                mensaje = 1 ;
21:            else
22:                mensaje = 0 ;
23:            datos.writeInt(mensaje) ;
24:            datos.close() ;
25:            bufferSalida.close() ;
26:            canalComunicacion.close() ;
27:        } catch (UnknownHostException ex) {
28:            Logger.getLogger(Cliente.class.getName()).log(Level.SEVERE, null, ex) ;
29:        }
    }
```

**Figura 56.** Código Aplicación Cliente  
**Fuente:** Máquina virtual controlador Floodlight

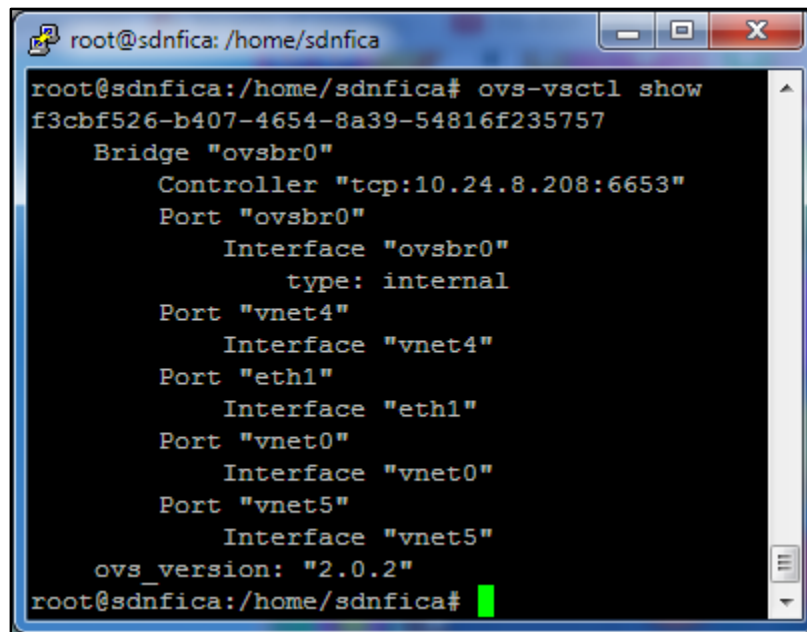
# CAPÍTULO IV

## PRUEBAS Y RESULTADOS

En este capítulo se presentan las pruebas desarrolladas para la demostración del funcionamiento de la red SDN y de la aplicación de acceso al medio, además de la configuración final de switches y la topología de la red.

### 4.1. CONFIGURACIÓN FINAL DE SWITCH

En la Figura 57, 58, 59 y 60 se puede observar un resumen de la configuración final de switch de la red SDN, en las que constan las interfaces, los servicios iniciados, las características de las interfaces y las rutas.



```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ovs-vsctl show
f3cbf526-b407-4654-8a39-54816f235757
  Bridge "ovsbr0"
    Controller "tcp:10.24.8.208:6653"
    Port "ovsbr0"
      Interface "ovsbr0"
        type: internal
    Port "vnet4"
      Interface "vnet4"
    Port "eth1"
      Interface "eth1"
    Port "vnet0"
      Interface "vnet0"
    Port "vnet5"
      Interface "vnet5"
  ovs_version: "2.0.2"
root@sdnfica:/home/sdnfica#
```

**Figura 57.** Resultado de ovs-vsctl show  
Fuente: Servidor SDN

```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ps -ea | grep ovs
22959 ?          00:03:18 ovsdb-server
22970 ?          00:38:11 ovs-vswitchd
root@sdnfica:/home/sdnfica#
```

**Figura 58.** Resultado de ps -ea | grep ovs  
**Fuente:** Servidor SDN

```
root@sdnfica: /home/sdnfica
root@sdnfica:/home/sdnfica# ovs-ofctl show ovsbr0
OFPT_FEATURES_REPLY (xid=0x2): dpid:000000237d9369cb
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS Q
UEUE_STATS ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_V
LAN SET_DL_SRC SET_DL_DST SET_NW_SRC SET_NW_DST S
ET_NW_TOS SET_TP_SRC SET_TP_DST ENQUEUE
1(eth1): addr:00:23:7d:93:69:cb
  config: 0
  state: 0
  current: 100MB-FD COPPER AUTO_NEG
  advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-F
D 1GB-HD 1GB-FD COPPER AUTO_NEG AUTO_PAUSE
  supported: 10MB-HD 10MB-FD 100MB-HD 100MB-F
D 1GB-HD 1GB-FD COPPER AUTO_NEG
  speed: 100 Mbps now, 1000 Mbps max
37(vnet4): addr:fe:54:00:98:fa:81
  config: 0
  state: 0
  current: 10MB-FD COPPER
  speed: 10 Mbps now, 0 Mbps max
38(vnet5): addr:fe:54:00:41:0c:f6
  config: 0
  state: 0
  current: 10MB-FD COPPER
  speed: 10 Mbps now, 0 Mbps max
56(vnet0): addr:fe:54:00:59:59:eb
  config: 0
  state: 0
  current: 10MB-FD COPPER
  speed: 10 Mbps now, 0 Mbps max
LOCAL(ovsbr0): addr:00:23:7d:93:69:cb
  config: 0
  state: 0
  speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal mis
s_send_len=0
root@sdnfica:/home/sdnfica#
```

**Figura 59.** Resultado de ovs-ofctl show ovsbr0  
**Fuente:** Servidor SDN

```
root@sdnfica: /home/sdnfica
root@sdnfica: /home/sdnfica# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use
Iface
0.0.0.0 10.24.8.1 0.0.0.0 UG 0 0 0
ovsbr0
10.24.8.0 0.0.0.0 255.255.255.0 U 0 0 0
ovsbr0
169.254.0.0 0.0.0.0 255.255.0.0 U 1000 0 0
ovsbr0
root@sdnfica: /home/sdnfica#
```

Figura 60. Resultado de route -n

Fuente: Servidor SDN

## 4.2. TOPOLOGÍA

Por medio de la interfaz de Floodlight se puede observar la topología de la red, como se muestra en la Figura 61.

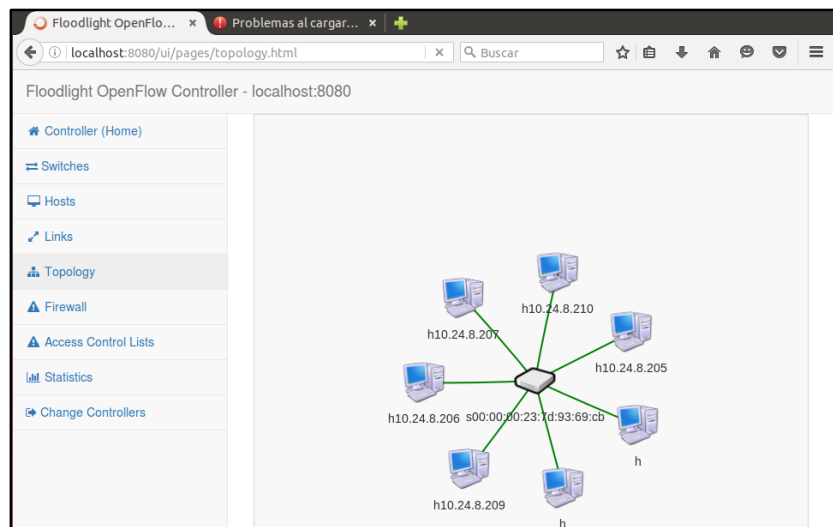
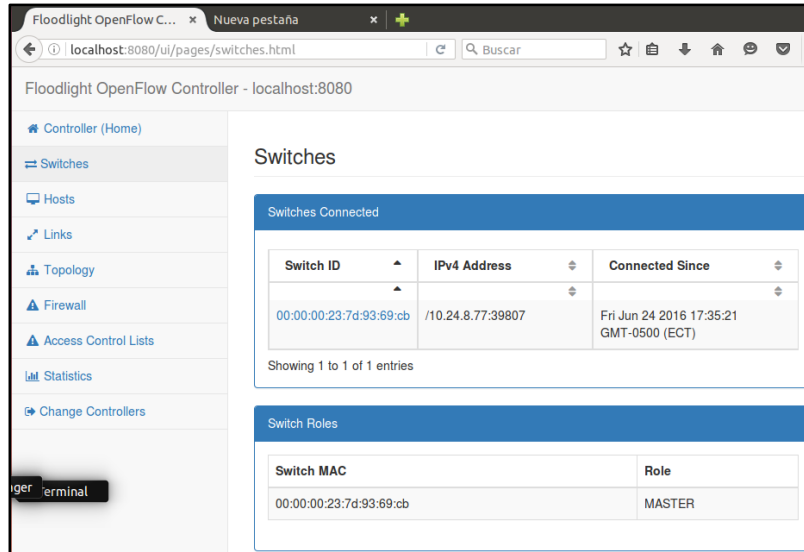


Figura 61. Topología de la red

Fuente: Máquina virtual controlador Floodlight

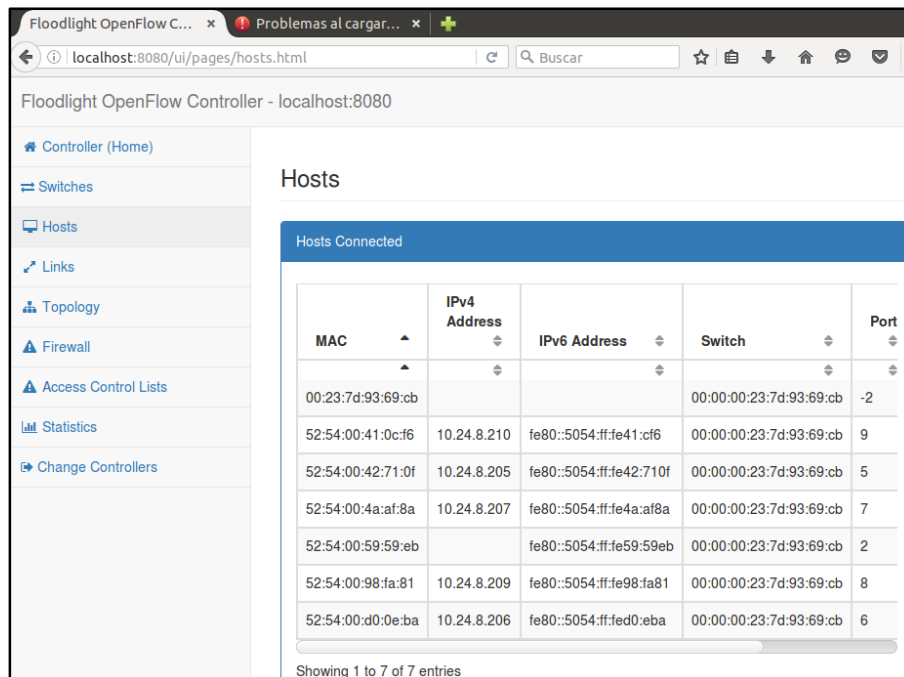
## 4.3. SWITCHES Y HOST

Por medio de la interfaz de Floodlight se observa a detección del switch y sus características, como en la Figura 62:



**Figura 62.** Switch virtual de la red  
**Fuente:** Máquina virtual controlador Floodlight

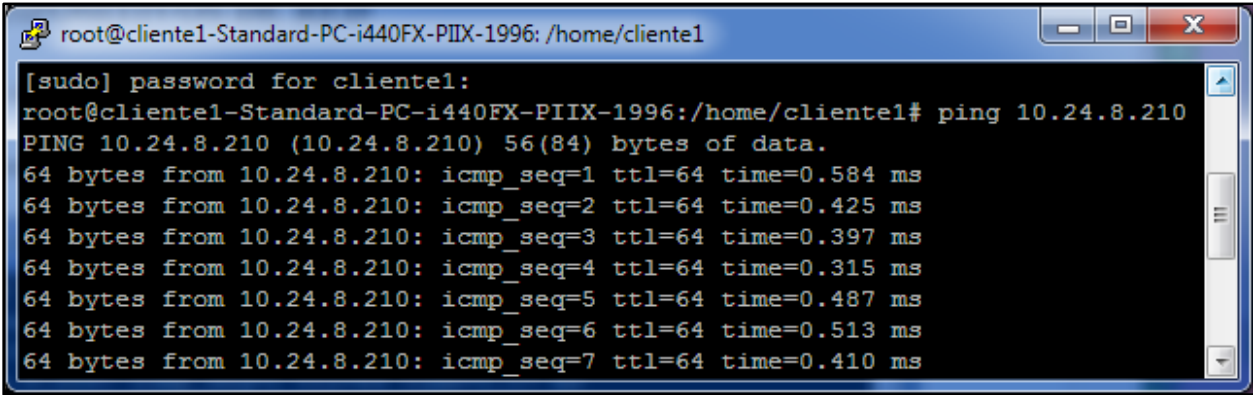
Por medio de la interfaz de Floodlight se puede observar los host conectados a la red y las características de los mismos, como se muestra en la Figura 63.



**Figura 63.** Host de la red  
**Fuente:** Máquina virtual controlador Floodlight

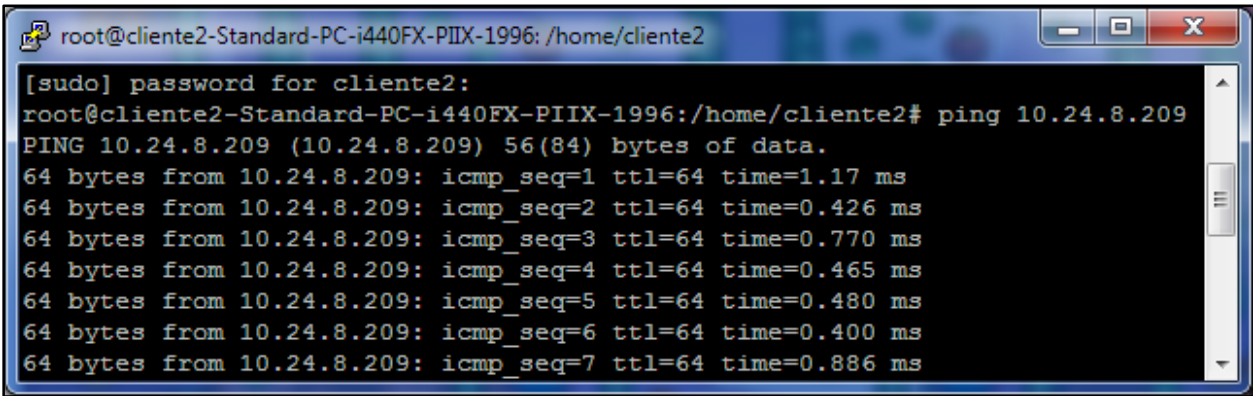
## 4.4. PING ENTRE DISPOSITIVOS DE RED

En la Figura 64 y 65 se puede observar la correcta conexión entre los clientes que conforman la red, por medio del uso del comando ping:

A terminal window titled 'root@cliente1-Standard-PC-i440FX-PIIX-1996: /home/cliente1'. The user enters '[sudo] password for cliente1:' and then 'ping 10.24.8.210'. The output shows a successful ping to 10.24.8.210 with 56(84) bytes of data and seven successful responses with varying times between 0.315 ms and 0.584 ms.

```
root@cliente1-Standard-PC-i440FX-PIIX-1996: /home/cliente1
[sudo] password for cliente1:
root@cliente1-Standard-PC-i440FX-PIIX-1996: /home/cliente1# ping 10.24.8.210
PING 10.24.8.210 (10.24.8.210) 56(84) bytes of data.
64 bytes from 10.24.8.210: icmp_seq=1 ttl=64 time=0.584 ms
64 bytes from 10.24.8.210: icmp_seq=2 ttl=64 time=0.425 ms
64 bytes from 10.24.8.210: icmp_seq=3 ttl=64 time=0.397 ms
64 bytes from 10.24.8.210: icmp_seq=4 ttl=64 time=0.315 ms
64 bytes from 10.24.8.210: icmp_seq=5 ttl=64 time=0.487 ms
64 bytes from 10.24.8.210: icmp_seq=6 ttl=64 time=0.513 ms
64 bytes from 10.24.8.210: icmp_seq=7 ttl=64 time=0.410 ms
```

**Figura 64.** Resultado ping Cliente1-Cliente2  
Fuente: Máquina Virtual Cliente1

A terminal window titled 'root@cliente2-Standard-PC-i440FX-PIIX-1996: /home/cliente2'. The user enters '[sudo] password for cliente2:' and then 'ping 10.24.8.209'. The output shows a successful ping to 10.24.8.209 with 56(84) bytes of data and seven successful responses with varying times between 0.400 ms and 1.17 ms.

```
root@cliente2-Standard-PC-i440FX-PIIX-1996: /home/cliente2
[sudo] password for cliente2:
root@cliente2-Standard-PC-i440FX-PIIX-1996: /home/cliente2# ping 10.24.8.209
PING 10.24.8.209 (10.24.8.209) 56(84) bytes of data.
64 bytes from 10.24.8.209: icmp_seq=1 ttl=64 time=1.17 ms
64 bytes from 10.24.8.209: icmp_seq=2 ttl=64 time=0.426 ms
64 bytes from 10.24.8.209: icmp_seq=3 ttl=64 time=0.770 ms
64 bytes from 10.24.8.209: icmp_seq=4 ttl=64 time=0.465 ms
64 bytes from 10.24.8.209: icmp_seq=5 ttl=64 time=0.480 ms
64 bytes from 10.24.8.209: icmp_seq=6 ttl=64 time=0.400 ms
64 bytes from 10.24.8.209: icmp_seq=7 ttl=64 time=0.886 ms
```

**Figura 65.** Resultado ping Cliente2-Cliente1  
Fuente: Máquina Virtual Cliente2

## 4.5. TRAFICO EN WIRESHARK

Por medio del sniffer Wireshark se puede visualizar los paquetes enviados entre los clientes al momento que se establece la comunicación entre los mismos, en la Figura 66 se observa los paquetes de TCP, ARP, ICMP, OFP.



| No.    | Time     | Source            | Destination   | Protocol | Length | Info   |
|--------|----------|-------------------|---------------|----------|--------|--|
| 127.30 | 1.133319 | 192.168.1.117     | 192.168.1.10  | TCP      | 66     | 6633 > 34139 [ACK] Seq=2032 Ack=11516 Win=2641 Len=0 |
| 128.30 | 1.425248 | 192.168.1.117     | 192.168.1.10  | DPF      | 78     | Stats Request (CSM) (128)                            |
| 129.30 | 1.428895 | 192.168.1.10      | 192.168.1.117 | DPF      | 1134   | Stats Reply (CSM) (10608)                            |
| 130.30 | 1.817862 | 192.168.1.117     | 192.168.1.10  | TCP      | 66     | 6633 > 34139 [ACK] Seq=2044 Ack=11584 Win=2641 Len=0 |
| 131.30 | 2.716594 | RealtekU 8f:cb:37 | Broadcast     | ARP      | 42     | Who has 192.168.1.1? Tell 192.168.1.190              |
| 132.30 | 7.723113 | 192.168.1.119     | 192.168.1.191 | ICMP     | 98     | Echo (ping) request 10=0x0072, seq=2/512, ttl=64     |
| 133.30 | 7.732297 | 192.168.1.191     | 192.168.1.119 | ICMP     | 98     | Echo (ping) reply 10=0x0072, seq=2/512, ttl=64       |
| 134.30 | 9.900519 | RealtekU 99:3c:40 | Broadcast     | ARP      | 60     | Who has 192.168.1.1? Tell 192.168.1.119              |
| 135.31 | 1.271668 | RealtekU 8f:cb:37 | Broadcast     | ARP      | 42     | Who has 192.168.1.1? Tell 192.168.1.190              |
| 136.31 | 1.773854 | 192.168.1.119     | 192.168.1.191 | ICMP     | 98     | Echo (ping) request 10=0x0072, seq=3/768, ttl=64     |
| 137.31 | 1.773915 | 192.168.1.191     | 192.168.1.119 | ICMP     | 98     | Echo (ping) reply 10=0x0072, seq=3/768, ttl=64       |

**Figura 66.** Paquetes TCP, ARP, ICMP Y OSPF  
**Fuente:** Máquina Virtual controlador Floodlight

Wireshark permite verificar el segmento TCP, como en la Figura 67, en el que se verifica el puerto de escucha del controlador y la IP del mismo:

```

Frame 127: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
on interface 0
Ethernet II, Src: RealtekU_02:f2:6c (52:54:00:02:f2:6c), Dst: NonMPLP_be:31:7c (00:27:00:0e:31:7c)
Internet Protocol Version 4, Src: 192.168.1.117 (192.168.1.117), Dst: 192.168.1.10 (192.168.1.10)
Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 34139 (34139), Seq: 2032, Ack: 11516, Len: 0
Source port: 6633 (6633)
Destination port: 34139 (34139)
[Stream Index: 0]
Sequence number: 2032 (relative sequence number)
Acknowledgement number: 11516 (relative ack number)
Header length: 32 bytes
Flags: 0x010 (ACK)
Window size value: 2641
[Calculated window size: 2641]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x7061 (validation disabled)
  
```

**Figura 67.** Segmento TCP  
**Fuente:** Máquina Virtual controlador Floodlight

En la Figura 68 y 69 se observa los paquetes de OPF, el cual se muestra la conexión con OpenFlow:

```

OpenFlow Protocol
Header
  Version: 0x01
  Type: Stats Request (CSM) (16)
  Length: 12
  Transaction ID: 1806
Stats Request
  Type: Description of this OpenFlow switch (0x0000)
  Flags: 0x0000
  Body: 0x1557400
  
```

**Figura 68.** Conexión OpenFlow  
**Fuente:** Máquina Virtual controlador Floodlight

```

OpenFlow Protocol
Header
  Version: 0x01
  Type: Stats Reply (CSM) (17)
  Length: 1068
  Transaction ID: 1806
Stats Reply
  Type: Description of this OpenFlow switch (0x0000)
  Flags: 0
Desc Stats Reply
  Mfr Desc: Nicira Networks, Inc.
  Hw Desc: Open vSwitch
  Sw Desc: 1.4.0+build0
  Serial Num: None
  Dp Desc: None
  
```

**Figura 69.** Conexión OpenFlow  
**Fuente:** Máquina Virtual controlador Floodlight

En la Figura 70 se observa el tráfico ARP, en el cual se visualiza las direcciones MAC e IP de los clientes:

```
Hardware type: Ethernet (1)
Protocol type: IP (0x8000)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
[Is gratuitous: False]
Sender MAC address: Realtek_0f:cb:37 (52:54:00:0f:cb:37)
Sender IP address: 192.168.1.190 (192.168.1.190)
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.1.1 (192.168.1.1)
```

**Figura 70.** Tráfico ARP  
**Fuente:** Máquina Virtual controlador Floodlight

En la Figura 71 y 72 se muestran los paquetes de ICMP en los que se observan las direcciones IP de destino y origen de los equipos conectados:

```
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0xc1f9 [correct]
Identifier (BE): 2162 (0x0872)
Identifier (LE): 29192 (0x7208)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)
[Response In: 133]
Data (36 bytes)
```

**Figura 71.** Paquetes ICMP  
**Fuente:** Máquina Virtual controlador Floodlight

```
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0xc9f9 [correct]
Identifier (BE): 2162 (0x0872)
Identifier (LE): 29192 (0x7208)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)
[Response To: 132]
[Response Time: 0,984 ms]
Data (36 bytes)
```

**Figura 72.** Paquetes ICMP  
**Fuente:** Máquina Virtual controlador Floodlight

## 4.6. MENSAJES ICMP

En la Figura 73 se observa las peticiones de ICMP, de todos los dispositivos que son parte de la red SDN:

| No.  | Time       | Source            | Destination       | Protocol | Length | Info   |
|------|------------|-------------------|-------------------|----------|--------|--|
| 480  | 97.973215  | 192.168.1.10      | 192.168.1.117     | TCP      | 66     | 42949 > 8633 [ACK] Seq=37491 Ack=6842 Win=115 Len=0 TSval=389662 TSecr=425838  |
| 489  | 97.973228  | 192.168.1.10      | 192.168.1.117     | TCP      | 66     | 42949 > 8633 [ACK] Seq=37491 Ack=6842 Win=115 Len=0 TSval=389662 TSecr=425838  |
| 490  | 97.973244  | Internet II       | 80.134.21         | OutIAT   | 49     | Internet II  |
| 491  | 98.838517  | 192.168.1.10      | 192.168.1.119     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=41536, ttl=64                                  |
| 492  | 98.838566  | 192.168.1.119     | 192.168.1.10      | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=41536, ttl=64                                    |
| 493  | 98.305509  | 192.168.1.119     | 192.168.1.10      | ICMP     | 98     | Echo (ping) request 16=0x0, seq=41536, ttl=64                                  |
| 494  | 98.305873  | 192.168.1.10      | 192.168.1.119     | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=41536, ttl=64                                    |
| 711  | 143.946265 | 192.168.1.117     | 192.168.1.10      | OFF      | 122    | Stats Request (CSM) (516)  |
| 712  | 143.946531 | 192.168.1.10      | 192.168.1.117     | OFF      | 102    | Stats Reply (CSM) (308)  |
| 713  | 143.988199 | 192.168.1.117     | 192.168.1.10      | TCP      | 66     | 6633 > 42949 [ACK] Seq=6692 Ack=55895 Win=2574 Len=0 TSval=426742 TSecr=406556 |
| 714  | 144.841878 | 192.168.1.10      | 192.168.1.119     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=3706, ttl=64                                   |
| 715  | 144.841523 | 192.168.1.119     | 192.168.1.10      | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=3706, ttl=64                                     |
| 716  | 145.843114 | 192.168.1.10      | 192.168.1.119     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=41024, ttl=64                                  |
| 717  | 145.843555 | 192.168.1.119     | 192.168.1.10      | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=41024, ttl=64                                    |
| 718  | 146.845171 | 192.168.1.10      | 192.168.1.119     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=51288, ttl=64                                  |
| 719  | 146.845051 | 192.168.1.119     | 192.168.1.10      | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=51288, ttl=64                                    |
| 1828 | 211.194698 | RealtekU 9f:76:5a | RealtekU 8f:cb:37 | ARP      | 68     | 192.168.1.118 is at 52:54:00:9f:76:5a  |
| 1829 | 211.195174 | 192.168.1.190     | 192.168.1.118     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=1256, ttl=64                                   |
| 1830 | 211.195444 | 192.168.1.118     | 192.168.1.190     | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=1256, ttl=64                                     |
| 1831 | 212.195860 | 192.168.1.190     | 192.168.1.118     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=2512, ttl=64                                   |
| 1832 | 212.196333 | 192.168.1.118     | 192.168.1.190     | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=2512, ttl=64                                     |
| 1838 | 338.245880 | 192.168.1.10      | 192.168.1.118     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=41024, ttl=64                                  |
| 1839 | 338.246259 | 192.168.1.118     | 192.168.1.10      | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=41024, ttl=64                                    |
| 1839 | 338.249016 | 192.168.1.10      | 192.168.1.118     | ICMP     | 98     | Echo (ping) request 16=0x0, seq=51288, ttl=64                                  |
| 1839 | 338.249478 | 192.168.1.118     | 192.168.1.10      | ICMP     | 98     | Echo (ping) reply 16=0x0, seq=51288, ttl=64                                    |

**Figura 73.** Mensajes ICMP  
**Fuente:** Máquina Virtual controlador Floodlight

## 4.7. INSERCIÓN DE REGLAS

En la Figura 74 se observa las reglas insertadas en el switch, las cuales permiten el tráfico ARP, y la conexión bidireccional entre el cliente y el controlador, por medio del puerto 5000:

```

1 [{"ruleid":1641349361,"dpid":"00:00:e8:03:9a:ab:ff:b2","in_port":0,"dl_src":"00:00:00:00:00:00","dl_dst":"00:00:00:00:00:00","dl_type":2054,"nw_src_prefix":"0.0.0.0","nw_src_maskbits":0,"nw_dst_prefix":"0.0.0.0","nw_dst_maskbits":0,"nw_proto":0,"tp_src":0,"tp_dst":0,"wildcard_dpid":false,"wildcard_in_port":true,"wildcard_dl_src":true,"wildcard_dl_dst":true,"wildcard_dl_type":false,"wildcard_nw_src":true,"wildcard_nw_dst":true,"wildcard_nw_proto":true,"wildcard_tp_src":true,"wildcard_tp_dst":true,"priority":0,"action":"ALLOW"}]

2 [{"ruleid":1952039499,"dpid":"00:00:e8:03:9a:ab:ff:b2","in_port":0,"dl_src":"00:00:00:00:00:00","dl_dst":"00:00:00:00:00:00","dl_type":2048,"nw_src_prefix":"0.0.0.0","nw_src_maskbits":0,"nw_dst_prefix":"0.0.0.0","nw_dst_maskbits":0,"nw_proto":6,"tp_src":0,"tp_dst":5000,"wildcard_dpid":false,"wildcard_in_port":true,"wildcard_dl_src":true,"wildcard_dl_dst":true,"wildcard_dl_type":false,"wildcard_nw_src":true,"wildcard_nw_dst":true,"wildcard_nw_proto":false,"wildcard_tp_src":true,"wildcard_tp_dst":false,"priority":0,"action":"ALLOW"}],

3 [{"ruleid":1739665884,"dpid":"00:00:e8:03:9a:ab:ff:b2","in_port":0,"dl_src":"00:00:00:00:00:00","dl_dst":"00:00:00:00:00:00","dl_type":2048,"nw_src_prefix":"0.0.0.0","nw_src_maskbits":0,"nw_dst_prefix":"0.0.0.0","nw_dst_maskbits":0,"nw_proto":6,"tp_src":5000,"tp_dst":0,"wildcard_dpid":false,"wildcard_in_port":true,"wildcard_dl_src":true,"wildcard_dl_dst":true,"wildcard_dl_type":false,"wildcard_nw_src":true,"wildcard_nw_dst":true,"wildcard_nw_proto":false,"wildcard_tp_src":false,"wildcard_tp_dst":true,"priority":0,"action":"ALLOW"}],

4 [{"ruleid":983025134,"dpid":"00:00:e8:03:9a:ab:ff:b2","in_port":0,"dl_src":"52:54:00:8f:cb:37","dl_dst":"00:00:00:00:00:00","dl_type":0,"nw_src_prefix":"0.0.0.0","nw_src_maskbits":0,"nw_dst_prefix":"0.0.0.0","nw_dst_maskbits":0,"nw_proto":0,"tp_src":0,"tp_dst":0,"wildcard_dpid":false,"wildcard_in_port":true,"wildcard_dl_src":false,"wildcard_dl_dst":true,"wildcard_dl_type":true,"wildcard_nw_src":true,"wildcard_nw_dst":true,"wildcard_nw_proto":true,"wildcard_tp_src":true,"wildcard_tp_dst":true,"priority":0,"action":"ALLOW"}],

```

**Figura 74.** Reglas insertadas en el Switch  
**Fuente:** Máquina Virtual controlador Floodlight

Otra forma de verificar las reglas insertadas es por medio de un explorador con la siguiente dirección: <http://10.24.8.208:8080/wm/firewall/rules/json>, las reglas se muestran en la Figura 75.

```

SEGUIRIDAD ESTABLECIDA Permitir ingreso a la red rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:e8:03:9a:ab:ff:b2", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:d0:27:88:be:31:7c", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:d0:27:88:be:31:7c", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:d0:27:88:be:31:7c", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":
rule("switchid":"00:00:d0:27:88:be:31:7c", "src-mac":"52:54:00:8f:cb:37", "dst-mac": "", "nw-prot": "", "dl-type":

```

**Figura 75.** Verificación de reglas en el explorador  
**Fuente:** Máquina Virtual controlador Floodlight

## 4.8. CONECTIVIDAD

Para verificar la conexión de los clientes es necesario ver las reglas, al principio se muestra las reglas existentes iniciales como en la Figura 76.

```

ServidorNAC (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (02/12/2013 11:35:46)
null
Numero de re
7
*****
52:54:00:8f:cb:37
52:54:00:8f:cb:37
Dispositivo i
52:54:00:99:3c:48
52:54:00:0a:8c:9c
52:54:00:0a:8c:9c
Dispositivo en aux
52:54:00:99:3c:48
52:54:00:99:3c:48
52:54:00:99:3c:48
Dispositivo en aux
52:54:00:99:3c:48
true
Rule deleted

```

**Figura 76.** Reglas iniciales  
**Fuente:** Máquina Virtual controlador Floodlight

En la Figura 77 se muestra las reglas después de la desconexión de un cliente:

```

Console (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (02/12/2013 11:35:46)
52:54:00:99:3c:48
null
Numero de reglas:
0
*****
52:54:00:8f:cb:37
52:54:00:8f:cb:37
Dispositivo en aux
52:54:00:99:3c:48
52:54:00:0a:8c:9c
52:54:00:0a:8c:9c
Dispositivo en aux
52:54:00:99:3c:48
52:54:00:9f:78:5a
52:54:00:9f:78:5a
Dispositivo en aux
52:54:00:99:3c:48
null

```

**Figura 77.** Reglas después de desconexión  
**Fuente:** Máquina Virtual controlador Floodlight

# CAPÍTULO V

## CONCLUSIONES Y RECOMENDACIONES

### 5.1. CONCLUSIONES

- Se realizó la implementación de una red SDN, con una topología constituida por switch virtual, el controlador y los clientes, por medio del manejo de flujos estáticos la persona encargada de la administración de red puede controlar el tráfico de datos, tomando en cuenta las necesidades de la red, además se desarrolló la aplicación NAC, con la cual se gestiona el acceso a la red por medio de reglas de tráfico, principalmente ARP y TCP.
- Se implementó una SDN con solución por software, que ofrece una virtualización económica, demostrando que este tipo de implementación es una alternativa para espacios en los que no se puede realizar una SDN basada en hardware.
- Se tomaron 4 posibles opciones para el controlador : NOX, POX, Beacon y Floodlight , en la elección de uno de ellos para la red SDN se verificó que no existe información necesaria de algunos de los mismos, por lo que la elección del más idóneo no resulta fácil, sin embargo la realización de pruebas con cada controlador permitió que se establezcan ciertos criterios, por ejemplo POX es un controlador muy acertado para redes SDN, debido a que posee muchos módulos, pero maneja un lenguaje poco conocido, Python el cual resultó en un inconveniente, generando que se explore otros controladores.

- El controlador NOX es el primero que se desarrolló para la creación de SDN, este se tomó como base para el desarrollo de POX, por lo que se ha dejado de lado a NOX, estancando su avance, debido a esta limitación se dice que no es el más idóneo para el proyecto, además de que presenta problemas de compatibilidad con la versión de sistema operativo Ubuntu 14.04
- El controlador Floodlight al igual que Beacon están desarrollados en Java, con la diferencia de que en Floodlight el número de paquetes es menor que en Beacon, debido a que este último necesita de más librerías y paquetes para su trabajo ,exigiendo a los programadores mayor conocimiento en el lenguaje Java, resulta más sencillo el manejo de Floodlight .
- StaticFlowEntryPusher y Firewall son módulos de Floodlight, los cuales se emplearon para el desarrollo de la aplicación NAC y para la culminación de la red SDN, llegándose a establecer que Floodlight posee muchos módulos con los que se pueden realizar diversas pruebas, para que el switch se comporte como uno de capa2, capa 3 o incluso hub, es necesario modificar una línea de código lo que implican un manejo sencillo de los módulos.
- Se eligió Floodlight como el controlador idóneo para la red SDN, debido a que está escrito en el lenguaje Java y posee una API la cual maneja JSON/REST, por medio de misma es posible utilizar diversas acciones que permitan determinar una política de seguridad por medio de la inserción de reglas de flujo.

- Se implementó una aplicación NAC para que cumpla con la función del administrador y determine si un equipo se conecta o no a la red, permitiendo que la red tenga sus propias características.
- El desarrollo de la aplicación NAC en la red es uno de los objetivos principales del proyecto, ya que permite controlar el acceso y dar opciones de seguridad a la misma, se demuestra que es posible crear y establecer diversos tipos de aplicaciones para mejorar el servicio, ingeniería de tráfico y movilidad, y de esta forma cubrir las necesidades y requisitos de los usuarios.
- Las diversas pruebas realizadas permiten verificar el correcto funcionamiento de la red, y la conectividad de la misma, las pruebas también permiten que se visualicen las reglas de tráfico insertadas, de acuerdo a si un cliente se ha conectado o desconectado de la red, haciendo uso de la aplicación.

## 5.2. RECOMENDACIONES

- Si se desea incursionar en SDN y no se dispone de la infraestructura física adecuada, se recomienda una solución por software, la cual permite dar muchas características a la red, puesto que se virtualiza los elementos que la componen, manejando tráfico de datos diferentes y reglas de seguridad.
- Es necesario borrar el kernel de Linux “bridge” para que el switch funcione de forma correcta, debido a que este se carga antes que el de Open vSwitch y se genera un conflicto en el arranque.
- Para elegir el software adecuado en la infraestructura de la red, es recomendable verificar las características de cada una de las opciones y basado en las facilidades y ventajas que ofrecen, tomar la decisión del más idóneo.
- Es recomendable que se estudie un poco de los lenguajes de programación en los que se desarrolla cada controlador, para que el manejo de dichos controladores no resulte complicado.
- Es necesario que se verifique la compatibilidad entre los controladores y la versión del Sistema operativo, debido a que en el caso de NOX no existe el soporte actualizado del mismo.



- Cuando se realiza la inserción de las reglas de flujo es necesario que se identifique de forma correcta las direcciones IP, direcciones MAC y puertos de cada uno de los equipos y de esta forma evitar confusiones.
- Se recomienda conocimientos del idioma Inglés en un nivel medio, debido a que el desarrollo de redes SDN es un tanto nuevo y la mayoría de información se encuentra en dicho idioma.
- Se recomienda que en un futuro se mejore la aplicación NAC, permitiendo que el cliente tenga mayor funcionalidad en cuanto a la búsqueda de seguridad.

## BIBLIOGRAFÍA

- Archive OpenFlow*. (2011). Retrieved from <http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf>
- CISCO*. (2015). Retrieved from <http://www.cisco.com/web/LA/soluciones/trends/sdn/index.html>
- citrix*. (2013). Retrieved from [https://www.citrix.com/content/dam/citrix/en\\_us/documents/oth/sdn-101-an-introduction-to-software-defined-networking-es.pdf](https://www.citrix.com/content/dam/citrix/en_us/documents/oth/sdn-101-an-introduction-to-software-defined-networking-es.pdf)
- cmqueue*. (2013, Noviembre 30). Retrieved from <http://queue.acm.org/detail.cfm?id=2560327>
- cpqd.github*. (2014). Retrieved from <http://cpqd.github.io/RouteFlow/>
- Dans, E. (2011, octubre 19). *enriquedans*. Retrieved from <https://www.enriquedans.com/2011/10/big-data-una-pequena-introduccion.html>
- dit*. (2012). Retrieved from [http://www.dit.upm.es/~posgrado/doc/TFM/TFMs2011-2012/TFM\\_Vanessa\\_Moreno\\_2012.pdf](http://www.dit.upm.es/~posgrado/doc/TFM/TFMs2011-2012/TFM_Vanessa_Moreno_2012.pdf)
- docs.oracle*. (2014). Retrieved from [https://docs.oracle.com/cd/E37929\\_01/html/E36563/gfkbw.html](https://docs.oracle.com/cd/E37929_01/html/E36563/gfkbw.html)
- Erickson, D. (2013). *openflow.stanford*. Retrieved from <https://openflow.stanford.edu/display/Beacon/Benchmarking>
- filotecnologa*. (2013). Retrieved from <https://filotecnologa.wordpress.com/tag/plano-de-control/>
- floodlight.atlassian*. (2015, Abril). Retrieved from <https://floodlight.atlassian.net/wiki>
- floodlight.atlassian.net*. (2015). Retrieved from <https://floodlight.atlassian.net/wiki/display/FlowScale>
- floodlight.openflow*. (2015). Retrieved from <http://floodlight.openflowhub.org/>
- Gandarillas, P. D. (2013, Mayo 27). Analisis de Factibilidad para la implementacion del protocolo OpenFlow en redesdefinidas por software. Jalisco, Guadalajara, Mexico.
- git.openvswitch*. (2012). Retrieved from [http://git.openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob\\_plain;f=FAQ;hb=HEAD](http://git.openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=FAQ;hb=HEAD)
- github*. (2014, Diciembre 2). Retrieved from <https://github.com/openvswitch/ovs/blob/master/README.md>
- HP*. (2014). Retrieved from <http://h17007.www1.hp.com/ec/es/solutions/technology/openflow/index.aspx>
- hp.com*. (2014). Retrieved from <http://www8.hp.com/h20195/v2/GetPDF.aspx/4AA5-7552ESE.pdf>
- informaticahoy*. (2013). Retrieved from <http://www.informatica-hoy.com.ar/la-nube/Cloud-Computing-Servicios-actuales-en-la-Nube.php>
- Jimenez, J. C. (2013, Agosto). Implementacion de un prototipo de una Red Definida por Software (SDN), empleando una solucion basada en Hardware. Quito, Pichincha, Ecuador.
- launchpad.net*. (2013). Retrieved from <https://launchpad.net/floodlight-openflow>

Millan, R. (2015). *ramonmillan*. Retrieved from <http://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php#referencias>

MorilloFuentala, D. G. (2014, Mayo). Implementacion de un prototipode una red definida por software (SDN), empleando una solucion basada ensoftware. Quito, Pichincha, Ecuador.

Nicira, M. C. (2013). *yuba.stanford*. Retrieved from <http://yuba.stanford.edu/~casado/virt-presto.pdf>

*nongnu*. (2014). Retrieved from <http://www.nongnu.org/quagga/>

*noxrepo.org*. (2014). Retrieved from <http://www.noxrepo.org/nox/about-nox/>

*noxrepo.org*. (2014). Retrieved from <http://www.noxrepo.org/pox/about-pox/>

*onlab.us*. (2015). Retrieved from [http://onlab.us/tools\\_nox.html](http://onlab.us/tools_nox.html)

*openflow.stanford*. (2013, Abril). Retrieved from <https://openflow.stanford.edu/display/Beacon/Home>

*openflow.stanford*. (2015, Marzo 5). Retrieved from <https://openflow.stanford.edu/display/ONL/POX+Wiki#POXWiki-ComponentsinPOX>

*opennetworking*. (2013, Abril 13). Retrieved from <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>

*opennetworking*. (2015). Retrieved from <https://www.opennetworking.org/sdn-resources/onf-specifications/OpenFlow>

Porras, P. (2012). *openflowsec*. Retrieved from [http://www.openflowsec.org/FortNOX\\_Sigcomm\\_HotSDN\\_2012.pdf](http://www.openflowsec.org/FortNOX_Sigcomm_HotSDN_2012.pdf)

*Princeton*. (2013). Retrieved from <http://www.cs.princeton.edu/~jrex/virtual.html>

*projectfloodlight*. (2014). Retrieved from <http://www.projectfloodlight.org/>

Rouse, M. (2013). *serachdatacenter*. Retrieved from <http://searchdatacenter.techtarget.com/es/definicion/Virtualizacion-de-las-funciones-de-red-NFV>

Rouse, M. (2014). *searchsdn*. Retrieved from <http://searchsdn.techtarget.com/definition/Network-hypervisor>

Santamaría, S. R. (2012, Septiembre). *Repositorio unican*. Retrieved from <http://repositorio.unican.es/xmlui/bitstream/handle/10902/1165/Sergio%20Rodriguez%20Santamaría.pdf?sequence=1>

*sdxcentral*. (2013). Retrieved from <https://www.sdxcentral.com/resources/sdn/what-the-definition-of-software-defined-networking-sdn/>

*sdxcentral*. (2014). Retrieved from <https://www.sdxcentral.com/resources/nfv/whats-network-functions-virtualization-nfv/>

*secutatis*. (2015). Retrieved from [http://www.secutatis.com/?page\\_id=62](http://www.secutatis.com/?page_id=62)

- Shin, S. (2013, Febrero). *openflowsec*. Retrieved from <http://www.openflowsec.org/FRESCO-NDSS2013.pdf>
- sites*. (2015). Retrieved from <https://sites.google.com/site/routeflow/>
- softsecuritycorp*. (2015). Retrieved from <http://softsecuritycorp.com/index.php/soluciones-y-productos/proteccion-de-redes-y-aplicaciones/control-de-acceso-a-la-red-nac>
- Spera, C. (2013, Marzo). *La Logicalis*. Retrieved from <http://www.la.logicalis.com/globalassets/latin-america/logicalisnow/revista-20/lnow20-nota-42-45.pdf>
- Stanford*. (2012, Diciembre). Retrieved from [http://onrc.stanford.edu/research\\_modern\\_sdn\\_stack.html](http://onrc.stanford.edu/research_modern_sdn_stack.html)
- Velez, J. (2012, Junio 1). *julyvezwordpress*. Retrieved from <https://julyvelez.wordpress.com/2012/06/01/ventajas-y-desventajas-de-la-tecnologia-en-la-sociedad/>
- vmware*. (2014). Retrieved from <http://www.vmware.com/latam/software-defined-datacenter/networking-security>

## ANEXOS

### ANEXO A

#### INSTALACIÓN VIRT MANAGER Y KVM

Verificar si el equipo donde se instalará los softwares, soporta la virtualización

```
root@SDN:/home/sdnflica# egrep -c 'lm' /proc/cpuinfo
4
root@SDN:/home/sdnflica# egrep -c '(svm|vmx)' /proc/cpuinfo
4
```

Código A-1. CPU disponibles para virtualización

```
root@SDN:/home/sdnflica# egrep '(vmx|svm)' --color=always /proc/cpuinfo
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant
_tsc arch_perfmon pebs bts rep_good nopl aperfmperf pni dtes64 monitor ds_cpl vm
x tm2 sse3 cx16 xtpr pdcm dca sse4_1 lahf_lm dtherm tpr_shadow vnmi flexpriorit
y
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant
_tsc arch_perfmon pebs bts rep_good nopl aperfmperf pni dtes64 monitor ds_cpl vm
x tm2 sse3 cx16 xtpr pdcm dca sse4_1 lahf_lm dtherm tpr_shadow vnmi flexpriorit
y
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant
_tsc arch_perfmon pebs bts rep_good nopl aperfmperf pni dtes64 monitor ds_cpl vm
x tm2 sse3 cx16 xtpr pdcm dca sse4_1 lahf_lm dtherm tpr_shadow vnmi flexpriorit
y
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant
_tsc arch_perfmon pebs bts rep_good nopl aperfmperf pni dtes64 monitor ds_cpl vm
x tm2 sse3 cx16 xtpr pdcm dca sse4_1 lahf_lm dtherm tpr_shadow vnmi flexpriorit
y
```

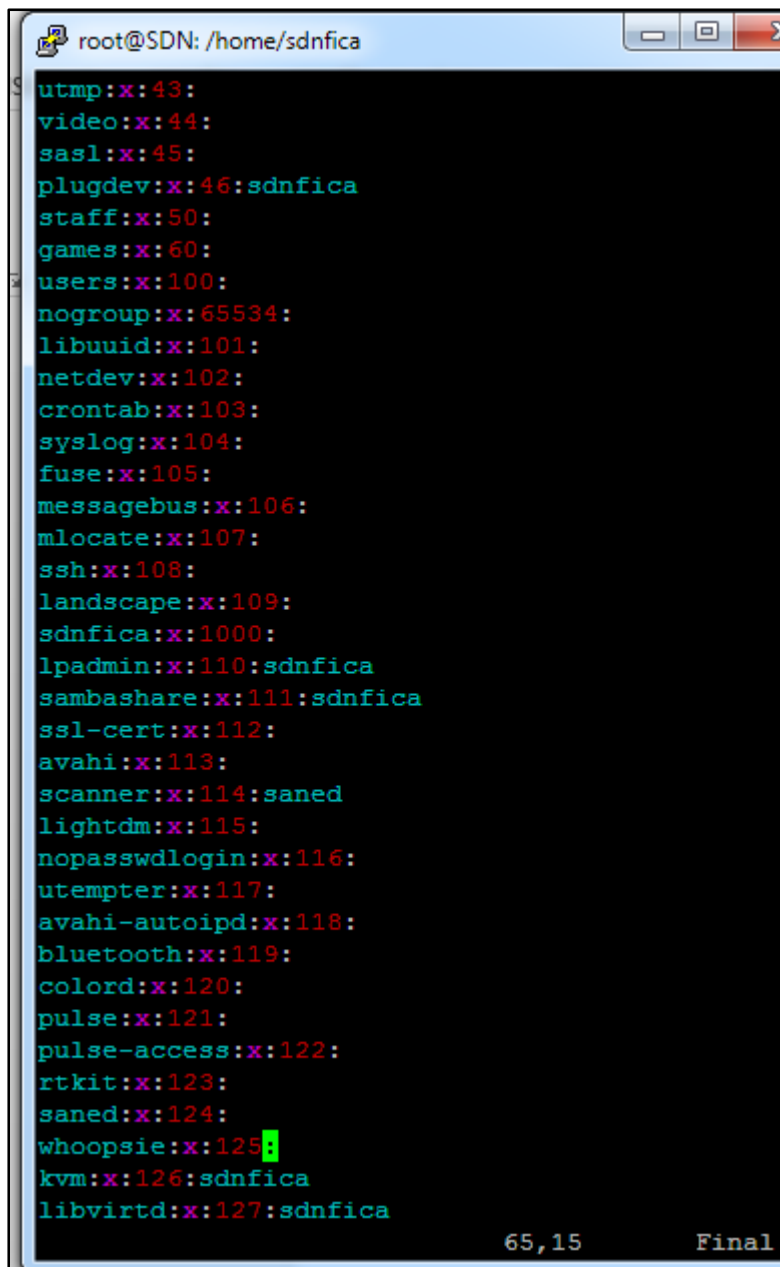
Código A-2. Soporte de hardware para virtualización

Escribir el comando de Instalación de KVM y virtual manager, además de la aplicación que permite administrar las máquinas virtuales

```
root@SDN:/home/sdnflica# apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils virt-manager virt-viewer
```

Código A-3. Comando de instalación de KVM

Solo el usuario root y los del grupo libvirt, pueden administrar las máquinas virtuales, anexar usuario a grupo libvirtd ubicado en etc/group, luego es necesario cerrar e iniciar de nuevo sesión.

A terminal window titled 'root@SDN: /home/sdnfica' displaying the contents of the /etc/group file. The output lists system and user groups with their respective GIDs and GECs. The groups listed are: utmp (43), video (44), sasl (45), plugdev (46, sdnfica), staff (50), games (60), users (100), nogroup (65534), libuuid (101), netdev (102), crontab (103), syslog (104), fuse (105), messagebus (106), mlocate (107), ssh (108), landscape (109), sdnfica (1000), lpadmin (110, sdnfica), sambashare (111, sdnfica), ssl-cert (112), avahi (113), scanner (114, saned), lightdm (115), nopasswdlogin (116), utempter (117), avahi-autoipd (118), bluetooth (119), colord (120), pulse (121), pulse-access (122), rtkit (123), saned (124), whoopsie (125), kvm (126, sdnfica), and libvirtd (127, sdnfica). The cursor is positioned at the end of the 'whoopsie' line. The terminal shows '65,15' and 'Final' at the bottom right.

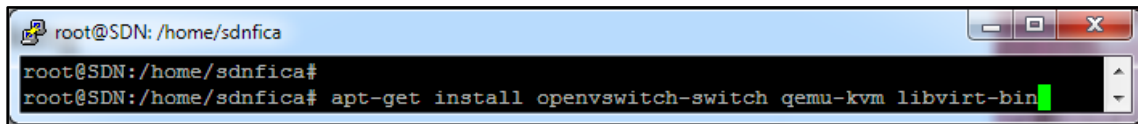
```
root@SDN: /home/sdnfica
utmp:x:43:
video:x:44:
sasl:x:45:
plugdev:x:46:sdnfica
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
libuuid:x:101:
netdev:x:102:
crontab:x:103:
syslog:x:104:
fuse:x:105:
messagebus:x:106:
mlocate:x:107:
ssh:x:108:
landscape:x:109:
sdnfica:x:1000:
lpadmin:x:110:sdnfica
sambashare:x:111:sdnfica
ssl-cert:x:112:
avahi:x:113:
scanner:x:114:saned
lightdm:x:115:
nopasswdlogin:x:116:
utempter:x:117:
avahi-autoipd:x:118:
bluetooth:x:119:
colord:x:120:
pulse:x:121:
pulse-access:x:122:
rtkit:x:123:
saned:x:124:
whoopsie:x:125:
kvm:x:126:sdnfica
libvirtd:x:127:sdnfica
65,15 Final
```

Código A-4. Archivo group

## ANEXO B

### INSTALACIÓN OPEN VSWITCH

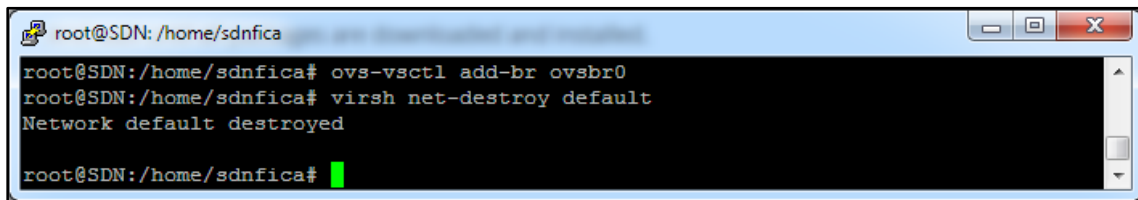
Para instalar open vSwitch es necesario instalar varias herramientas, por medio de los códigos que se muestran a continuación:



```
root@SDN: /home/sdnfica
root@SDN: /home/sdnfica#
root@SDN: /home/sdnfica# apt-get install openvswitch-switch qemu-kvm libvirt-bin
```

Código B-1. Instalación de paquetes para open vSwitch

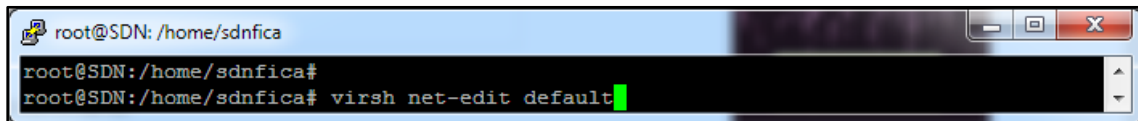
Creación del nuevo puente y es necesario eliminar el que viene instalado por defecto



```
root@SDN: /home/sdnfica
root@SDN: /home/sdnfica# ovs-vsctl add-br ovsbr0
root@SDN: /home/sdnfica# virsh net-destroy default
Network default destroyed
root@SDN: /home/sdnfica#
```

Código B-2. Anexar puente y destruir default

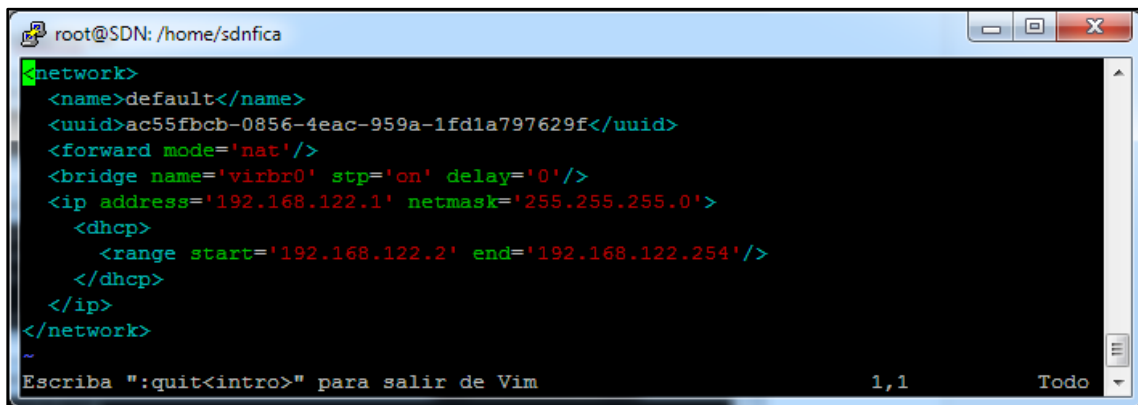
Se debe editar la configuración del archivo por defecto



```
root@SDN: /home/sdnfica
root@SDN: /home/sdnfica#
root@SDN: /home/sdnfica# virsh net-edit default
```

Código B-3. Ingreso a default

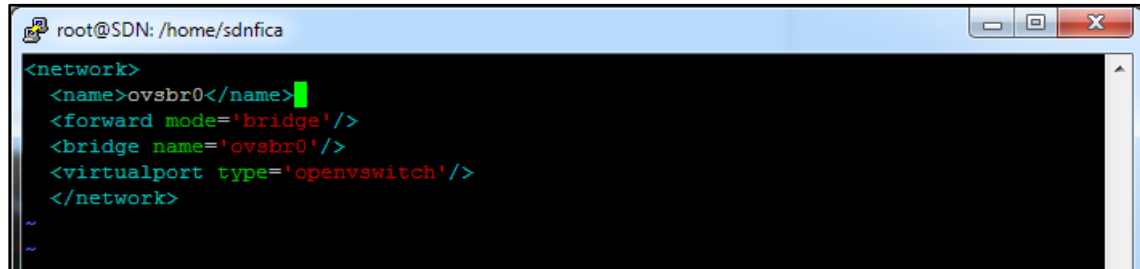
Configuración inicial, la cual se debe cambiar con los datos del nuevo puente



```
network>
<name>default</name>
<uuid>ac55fbc0-0856-4eac-959a-1fd1a797629f</uuid>
<forward mode='nat' />
<bridge name='virbr0' stp='on' delay='0' />
<ip address='192.168.122.1' netmask='255.255.255.0'>
  <dhcp>
    <range start='192.168.122.2' end='192.168.122.254' />
  </dhcp>
</ip>
</network>
~
Escriba ":quit<intro>" para salir de Vim 1,1 Todo
```

Código B-4. Contenido archivo default

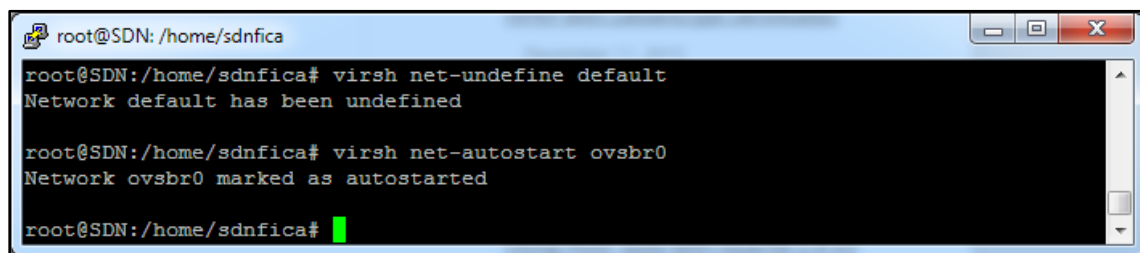
Configurar el nuevo puente



```
root@SDN: /home/sdnfica
<network>
  <name>ovsbr0</name>
  <forward mode='bridge' />
  <bridge name='ovsbr0' />
  <virtualport type='openvswitch' />
</network>
~
~
```

Código B-5. Nueva configuración para ovsbr0

Se debe quitar la definición de default y configurar el auto iniciación de ovsbr0



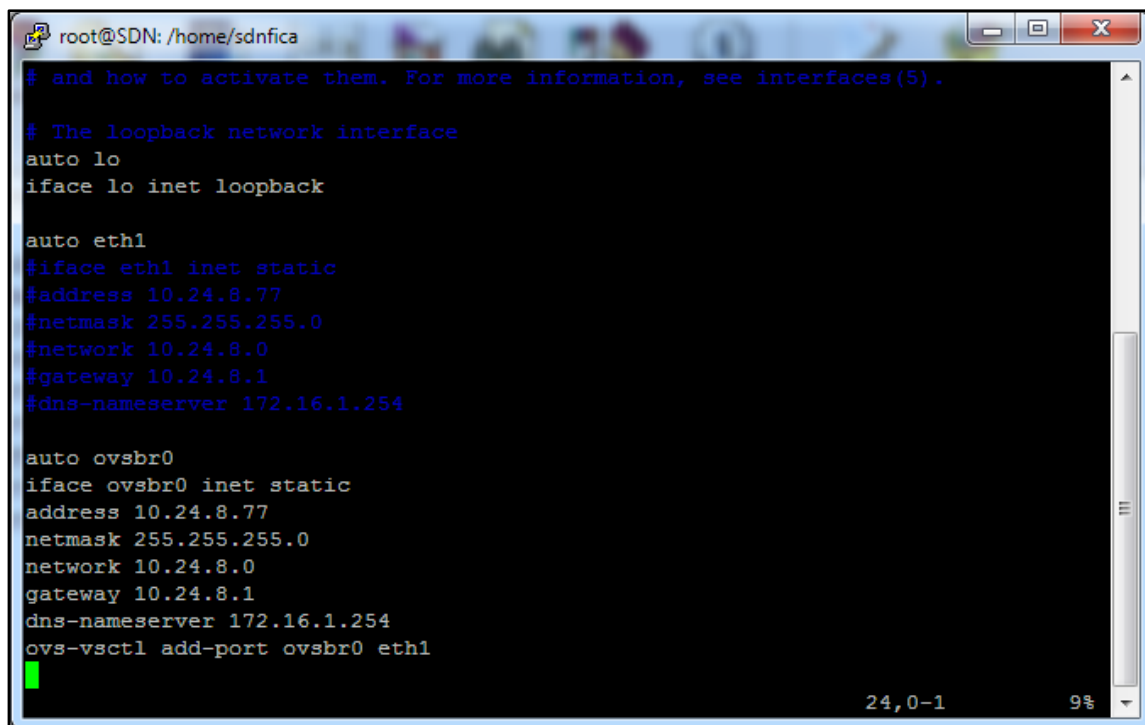
```
root@SDN: /home/sdnfica
root@SDN:/home/sdnfica# virsh net-undefine default
Network default has been undefined

root@SDN:/home/sdnfica# virsh net-autostart ovsbr0
Network ovsbr0 marked as autostarted

root@SDN:/home/sdnfica#
```

Código B-6. Auto iniciar ovsbr0

Configurar ovsbr0 como una nueva interfaz, en el archivo de configuración de las interfaces, utilizando la misma información de la interfaz física eth1



```
root@SDN: /home/sdnfica
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

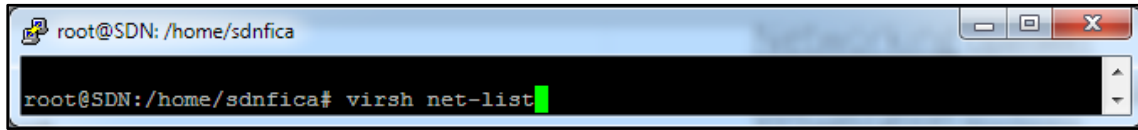
auto eth1
#iface eth1 inet static
#address 10.24.8.77
#netmask 255.255.255.0
#network 10.24.8.0
#gateway 10.24.8.1
#dns-nameserver 172.16.1.254

auto ovsbr0
iface ovsbr0 inet static
address 10.24.8.77
netmask 255.255.255.0
network 10.24.8.0
gateway 10.24.8.1
dns-nameserver 172.16.1.254
ovs-vsctl add-port ovsbr0 eth1
```

Código B-7. Configuración interfaz ovsbr0

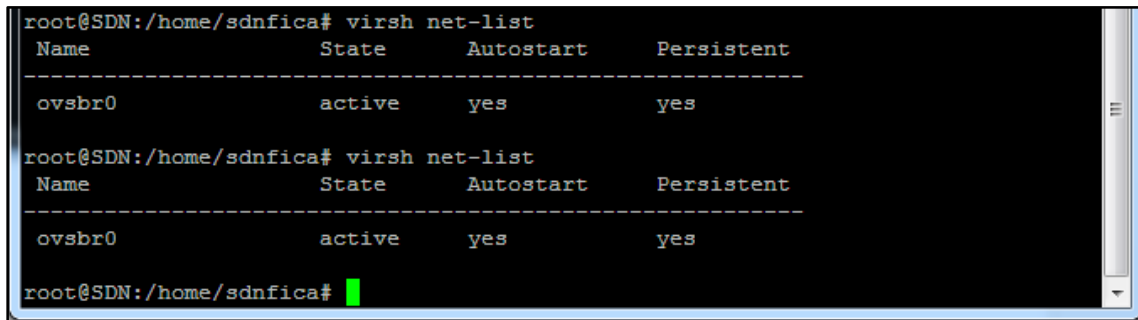


Verificar la creación de ovsbr0



```
root@SDN: /home/sdnfica
root@SDN:/home/sdnfica# virsh net-list
```

Código B-8. Comando de verificación

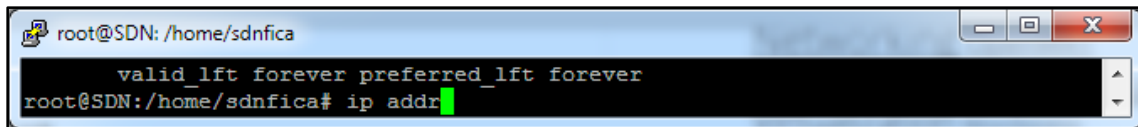


```
root@SDN:/home/sdnfica# virsh net-list
Name          State      Autostart   Persistent
-----
ovsbr0        active    yes         yes

root@SDN:/home/sdnfica# virsh net-list
Name          State      Autostart   Persistent
-----
ovsbr0        active    yes         yes

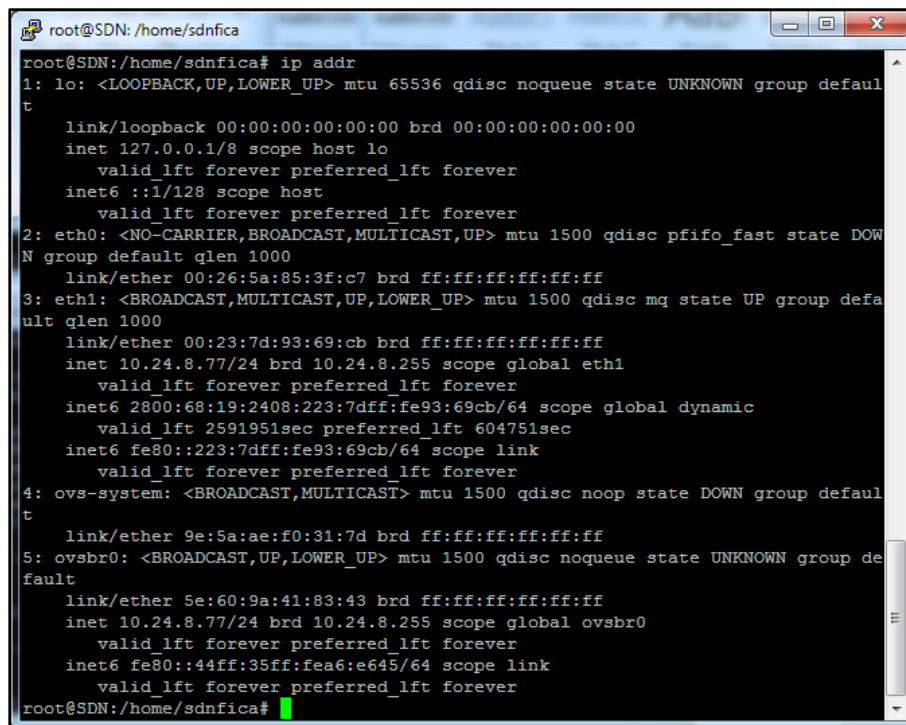
root@SDN:/home/sdnfica#
```

Código B-9. Resultado de virsh net-list



```
root@SDN: /home/sdnfica
valid_lft forever preferred_lft forever
root@SDN:/home/sdnfica# ip addr
```

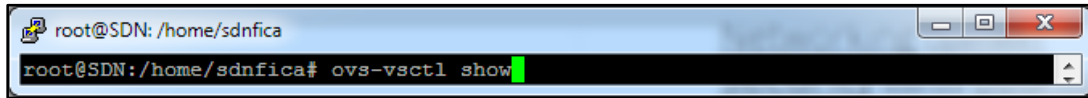
Código B-10. Comando de verificación



```
root@SDN: /home/sdnfica
root@SDN:/home/sdnfica# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether 00:26:5a:85:3f:c7 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:23:7d:93:69:cb brd ff:ff:ff:ff:ff:ff
    inet 10.24.8.77/24 brd 10.24.8.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 2800:68:19:2408:223:7dff:fe93:69cb/64 scope global dynamic
        valid_lft 2591951sec preferred_lft 604751sec
    inet6 fe80::223:7dff:fe93:69cb/64 scope link
        valid_lft forever preferred_lft forever
4: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
    link/ether 9e:5a:ae:f0:31:7d brd ff:ff:ff:ff:ff:ff
5: ovsbr0: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
    link/ether 5e:60:9a:41:83:43 brd ff:ff:ff:ff:ff:ff
    inet 10.24.8.77/24 brd 10.24.8.255 scope global ovsbr0
        valid_lft forever preferred_lft forever
    inet6 fe80::44ff:35ff:fea6:e645/64 scope link
        valid_lft forever preferred_lft forever
root@SDN:/home/sdnfica#
```

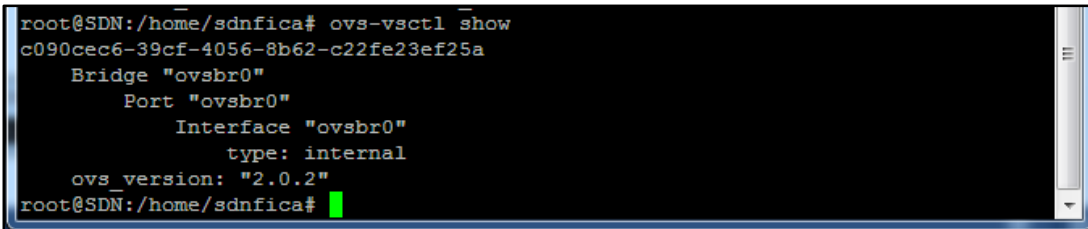
Código B-11. Resultado de ip addr

Se muestra toda la activación de ovsbr0 con el siguiente comando:



```
root@SDN: /home/sdnfica
root@SDN: /home/sdnfica# ovs-vsctl show
```

**Código B-12.** Comando para mostrar activación



```
root@SDN: /home/sdnfica# ovs-vsctl show
c090cec6-39cf-4056-8b62-c22fe23ef25a
  Bridge "ovsbr0"
    Port "ovsbr0"
      Interface "ovsbr0"
        type: internal
    ovs_version: "2.0.2"
root@SDN: /home/sdnfica#
```

**Código B-13.** Resultado de ovs-vsctl show

Para que la interfaz ovsbr0 funcione de forma correcta, es necesario encerrar la interfaz física eth1.



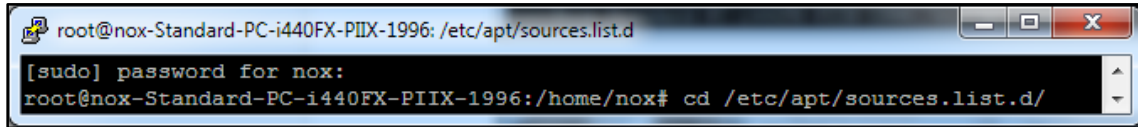
```
root@SDN: /home/sdnfica
root@SDN: /home/sdnfica# ifconfig eth1 0
```

**Código B-14.** Encerrar eth1

## ANEXO C

### INSTALACIÓN DE NOX

Ingresar a la carpeta donde se va instalar todos los paquetes



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d
[sudo] password for nox:
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox# cd /etc/apt/sources.list.d/
```

**Código C-1.** Ingreso etc/apt/sources.list.d/

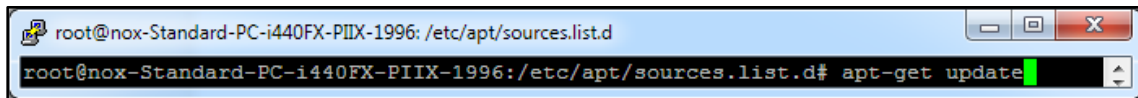
Instalación de los paquetes previos necesarios para NOX



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d# wget http://openflowswitch.org/downloads/debian/nox.list
```

**Código C-2.** Comando de instalación de paquetes

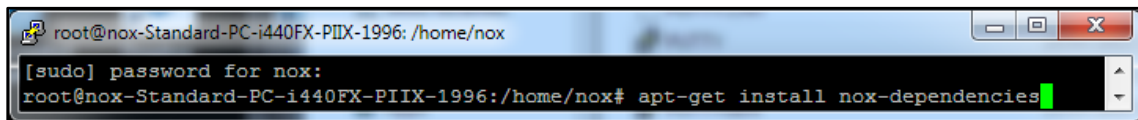
Actualizar los repositorios



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d# apt-get update
```

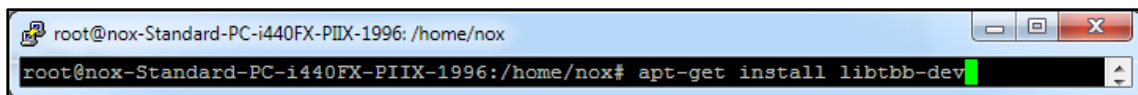
**Código C-3.** Comando de actualización

Instalar las dependencias para el controlador NOX



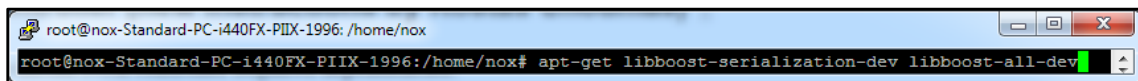
```
root@nox-Standard-PC-i440FX-PIIX-1996: /home/nox
[sudo] password for nox:
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox# apt-get install nox-dependencies
```

**Código C-4.** Comando de instalación de dependencias



```
root@nox-Standard-PC-i440FX-PIIX-1996: /home/nox
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox# apt-get install libtbb-dev
```

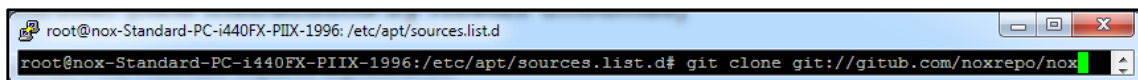
**Código C-5.** Comando de instalación de libtbb



```
root@nox-Standard-PC-i440FX-PIIX-1996: /home/nox
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox# apt-get install libboost-serialization-dev libboost-all-dev
```

**Código C-6.** Comando de instalación de libboost

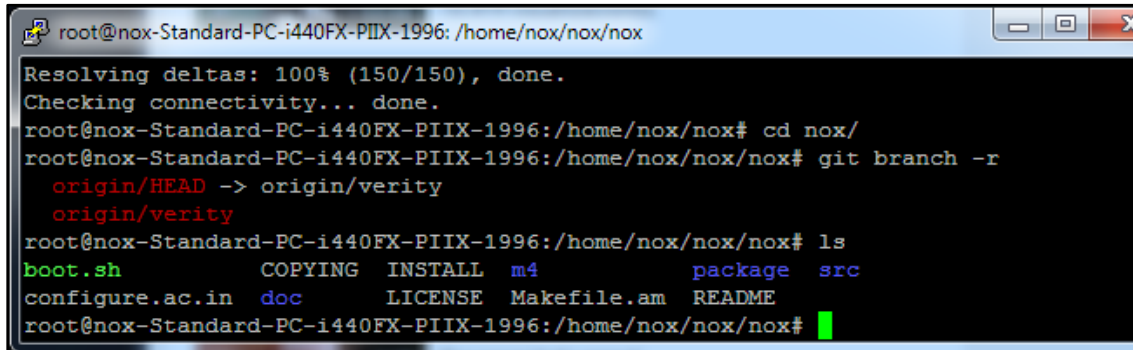
Clonar el proyecto en la carpeta de NOX



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d# git clone git://github.com/noxrepo/nox
```

**Código C-7.** Comando para clonar

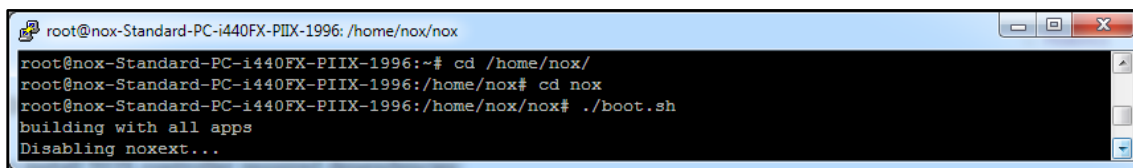
Verificar la clonación en la carpeta y buscar el archivo boot.sh



```
root@nox-Standard-PC-i440FX-PIIX-1996: /home/nox/nox/nox
Resolving deltas: 100% (150/150), done.
Checking connectivity... done.
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox/nox# cd nox/
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox/nox/nox# git branch -r
origin/HEAD -> origin/verity
origin/verity
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox/nox/nox# ls
boot.sh          COPYING  INSTALL  m4        package  src
configure.ac.in doc      LICENSE  Makefile.am  README
```

Código C-8. Listado de archivos en nox

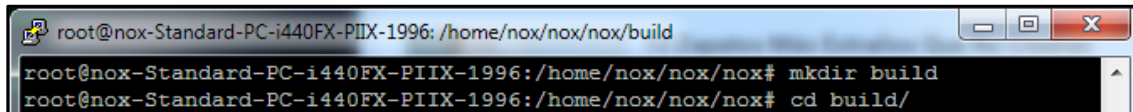
Es necesario ejecutar el archivo boot.sh



```
root@nox-Standard-PC-i440FX-PIIX-1996: /home/nox/nox
root@nox-Standard-PC-i440FX-PIIX-1996:~# cd /home/nox/
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox# cd nox
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox/nox# ./boot.sh
building with all apps
Disabling noxext...
```

Código C-9. Ejecución de boot.sh

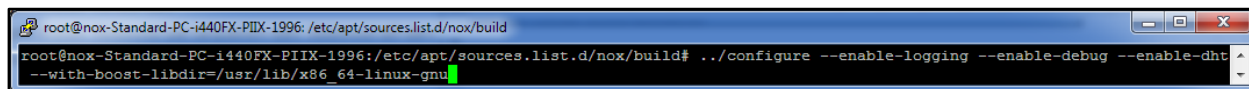
Crear carpeta build para construir los archivos



```
root@nox-Standard-PC-i440FX-PIIX-1996: /home/nox/nox/nox/build
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox/nox/nox# mkdir build
root@nox-Standard-PC-i440FX-PIIX-1996:/home/nox/nox/nox# cd build/
```

Código C-10. Crear carpeta

Descarga dependencias por medio del siguiente comando:



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build# ../configure --enable-logging --enable-debug --enable-dht
--with-boost-libdir=/usr/lib/x86_64-linux-gnu
```

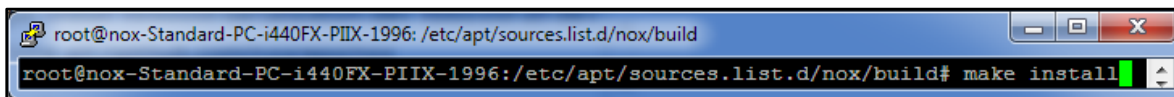
Código C-11. Comando configure

Se ejecutan los siguientes comandos:



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build# make
```

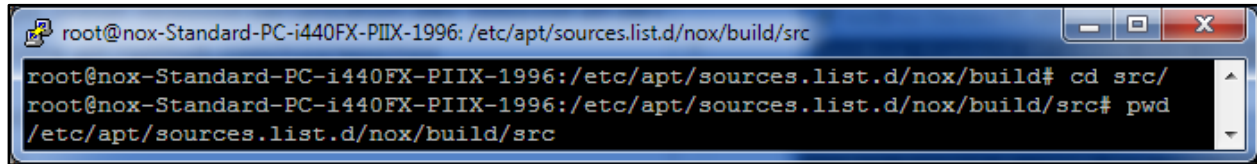
Código C-12. Comando make



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build# make install
```

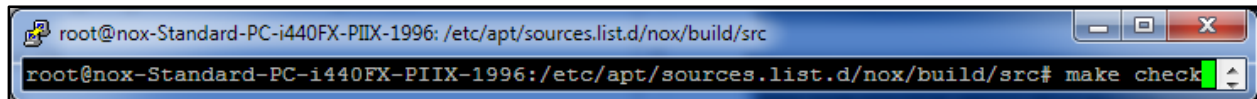
Código C-13. Comando make install

## Verificar la instalación



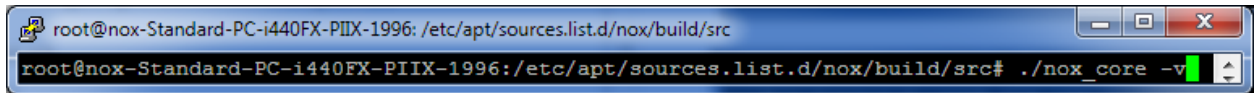
```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build# cd src/
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build/src# pwd
/etc/apt/sources.list.d/nox/build/src
```

**Código C-14.** Comando para verificar ubicación



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build/src# make check
```

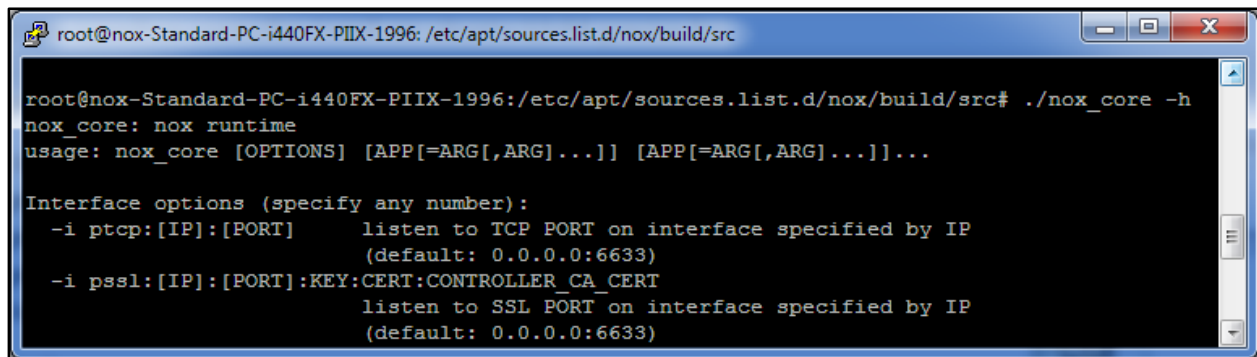
**Código C-15.** Comando para verificar make



```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build/src# ./nox_core -v
```

**Código C-16.** Comando para verificar el núcleo de NOX

## Verificación de puerto de escucha

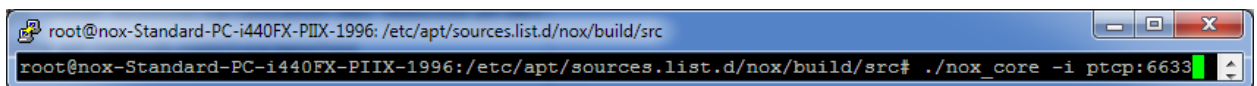


```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build/src# ./nox_core -h
nox_core: nox runtime
usage: nox_core [OPTIONS] [APP[=ARG[,ARG]...]] [APP[=ARG[,ARG]...]]...

Interface options (specify any number):
  -i ptcp:[IP]:[PORT]    listen to TCP PORT on interface specified by IP
                        (default: 0.0.0.0:6633)
  -i pssl:[IP]:[PORT]:KEY:CERT:CONTROLLER_CA_CERT
                        listen to SSL PORT on interface specified by IP
                        (default: 0.0.0.0:6633)
```

**Código C-17.** Comando para ver el puerto

## Ejecutar NOX por medio del puerto de escucha



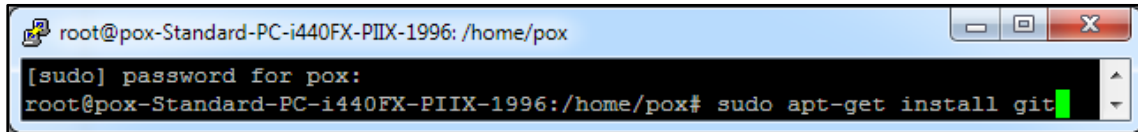
```
root@nox-Standard-PC-i440FX-PIIX-1996: /etc/apt/sources.list.d/nox/build/src
root@nox-Standard-PC-i440FX-PIIX-1996:/etc/apt/sources.list.d/nox/build/src# ./nox_core -i ptcp:6633
```

**Código C-18.** Comando ejecutar NOX

## ANEXO D

### INSTALACIÓN POX

Descargar los repositorios de la herramienta git



```
root@pox-Standard-PC-i440FX-PIIX-1996: /home/pox
[sudo] password for pox:
root@pox-Standard-PC-i440FX-PIIX-1996:/home/pox# sudo apt-get install git
```

**Código D-1.** Comando instalar git

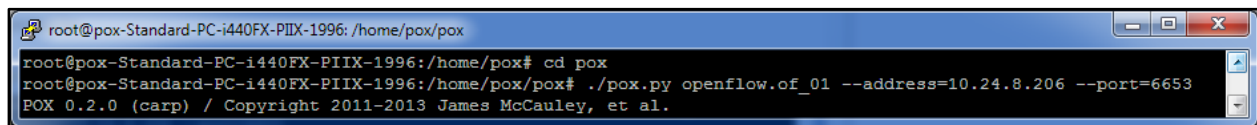
Clonar el proyecto de POX dentro de la carpeta pox



```
root@pox-Standard-PC-i440FX-PIIX-1996: /home/pox
root@pox-Standard-PC-i440FX-PIIX-1996:/home/pox# git clone https://github.com/noxrepo/pox
```

**Código D-2.** Comando clonar POX

Ejecución de POX por medio del siguiente comando:



```
root@pox-Standard-PC-i440FX-PIIX-1996: /home/pox/pox
root@pox-Standard-PC-i440FX-PIIX-1996:/home/pox# cd pox
root@pox-Standard-PC-i440FX-PIIX-1996:/home/pox/pox# ./pox.py openflow.of_01 --address=10.24.8.206 --port=6653
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
```

**Código D-3.** Comando ejecución POX

## ANEXO E

### INSTALACIÓN BEACON

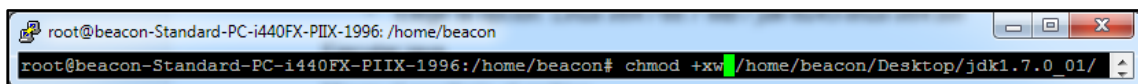
Para manejar Beacon es necesario instalar el software Eclipse y se lo puede descargar de la siguiente dirección: [www.eclipse.org/downloads](http://www.eclipse.org/downloads)

- Elegir la opción: *Eclipse for RCP and RAP Developers, para Linux de 64 bits*
- Ubicar la carpeta en el Escritorio y extraerlo.

También es necesario descargar JAVA, y se localiza en la siguiente dirección: <http://www.oracle.com/technetwork/java/javase/downloads/jdk6-jsp-136632.html>

- Descargar: *Java SE Development Kit 7*
- En la sección: *Java SE Development Kit 7u1*
- Elegir la opción: *Linux x64*

Para Ejecutar JAVA es necesario mover al escritorio el archivo descargado y darle los permisos necesarios

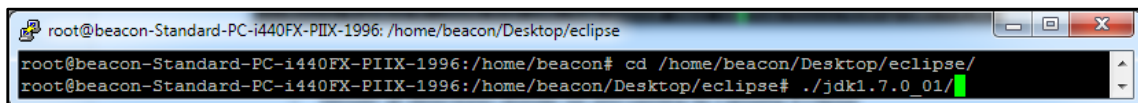


```
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon# chmod +x /home/beacon/Desktop/jdk1.7.0_01/
```

**Código E-1.** Permisos de ejecución y escritura archivo JAVA

- Mover el archivo al directorio donde se encuentra la carpeta Eclipse previamente descargada

Ir al directorio correspondiente donde están los archivos y ejecutar el archivo de java



```
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon/Desktop/eclipse
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon# cd /home/beacon/Desktop/eclipse/
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon/Desktop/eclipse# ./jdk1.7.0_01/
```

**Código D-2.** Ejecutar jdk

Se crea una carpeta con el nombre: jdk1.7.0\_01, copiar la subcarpeta con el nombre “jre” en la carpeta principal

```
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon/Desktop/eclipse/jdk1.7.0_01
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/Desktop/eclipse# ls
artifacts.xml  dropins  eclipse.ini  icon.xpm  p2  readme
configuration  eclipse  features  jdk1.7.0_01  plugins
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/Desktop/eclipse# cd jdk1.7.0_01/
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/Desktop/eclipse/jdk1.7.0_01# ls
bin  db  include  lib  man  release  src.zip
COPYRIGHT  demo  jre  LICENSE  README.html  sample  THIRDPARTYLICENSEREADME.txt
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/Desktop/eclipse/jdk1.7.0_01#
```

Código D-3. Verificación de carpetas

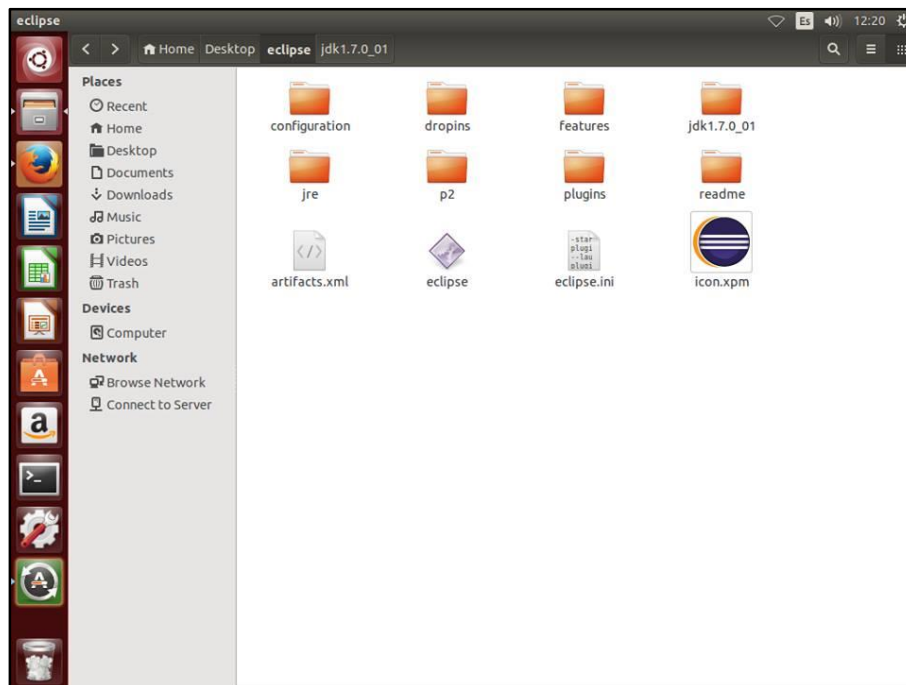


Figura D-1. Copiar carpeta

Verificar fuentes de Beacon y OpenFlowj creando carpetas

```
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon/git
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon# cd git/
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git# mkdir beacon
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git# mkdir openflowj
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git#
```

Código D-4. Creación de carpetas

Clonar el proyecto Beacon y Openflowj

```
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon/git/beacon
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git# cd beacon/
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git/beacon# git clone git://gitosis.stanford.edu/beacon.git
```

Código D-5. Comando clonar Beacon



```
root@beacon-Standard-PC-i440FX-PIIX-1996: /home/beacon/git/openflowj
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git# cd openflowj/
root@beacon-Standard-PC-i440FX-PIIX-1996:/home/beacon/git/openflowj# git clone git://gitorious.org/openflowj.git
```

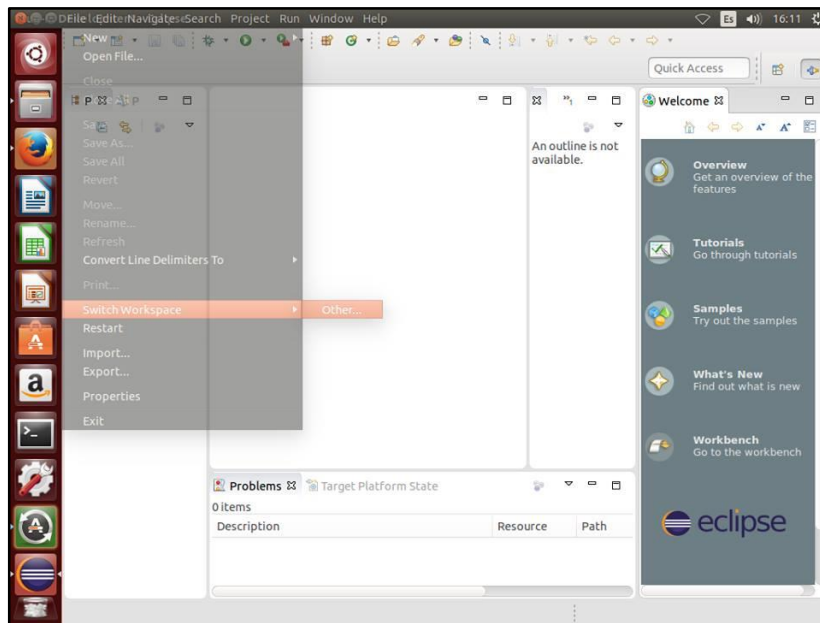
**Código D-6.** Comando clonar Openflowj

## DESARROLLO EN ECLIPSE

Acceder al entorno Eclipse mediante el ejecutable correspondiente.

- Crear un nuevo workspace: /home/beacon/**workspacebeacon**

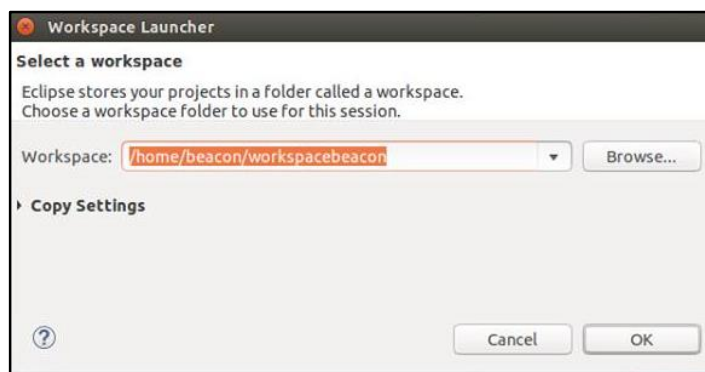
File – *Switch* Workspace – Other,



**Figura D-2.** Crear nuevo espacio

Seleccionar la nueva carpeta para el espacio de trabajo del host./home/beacon/**workspace-**

**Beacon**



**Figura D-3.** Ubicación del espacio

Modificar el nivel de “cumplimiento” de Eclipse a 1.6:

Window - Preferences - Java – Compiler, bajo JDK Compliance, cambiar el Compilador al nivel 1.6.

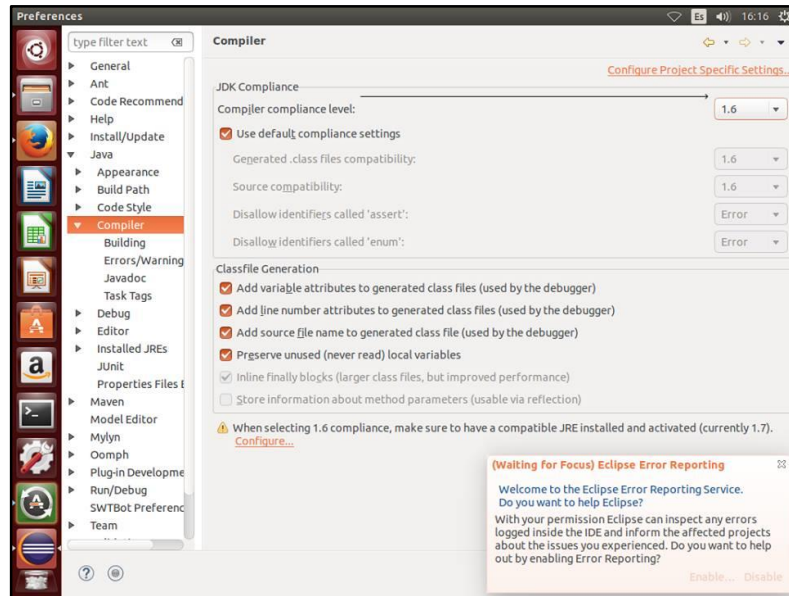


Figura D-4. Cambio del nivel compliance

Importar los proyectos OpenFlowJ y Beacon:

File - Import - General - Existing Projects into ~~W~~ Next.

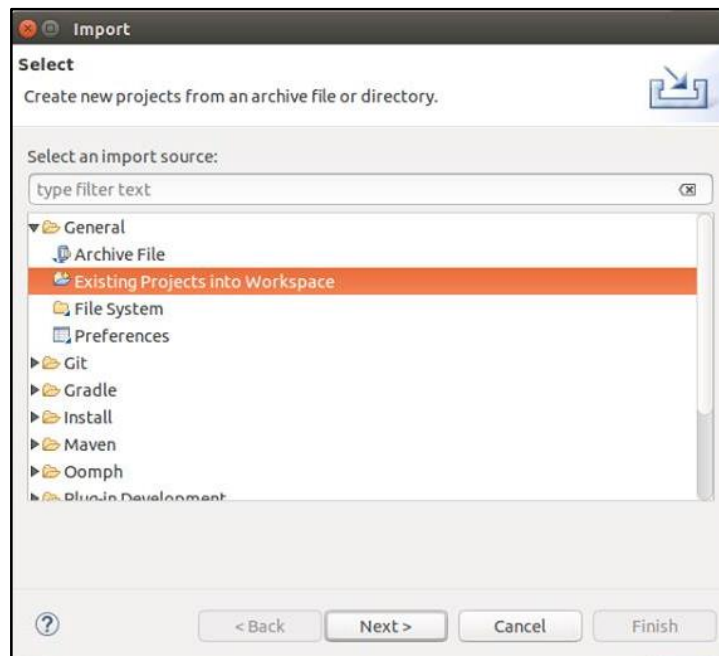
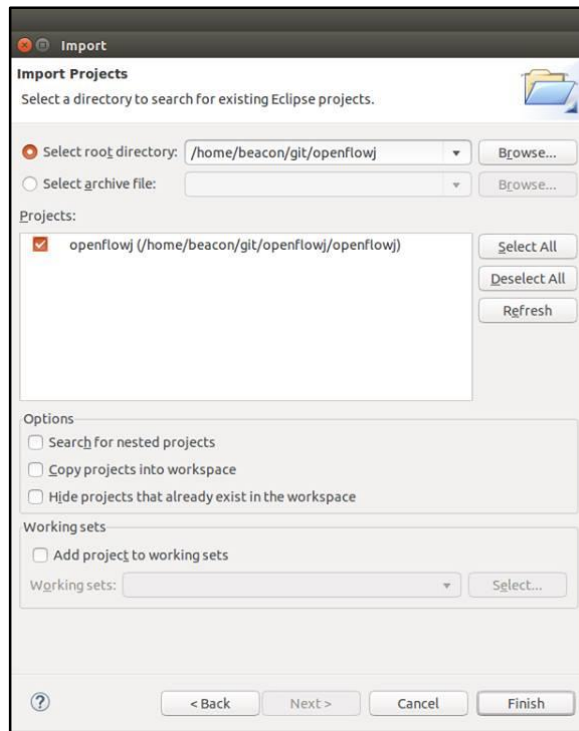


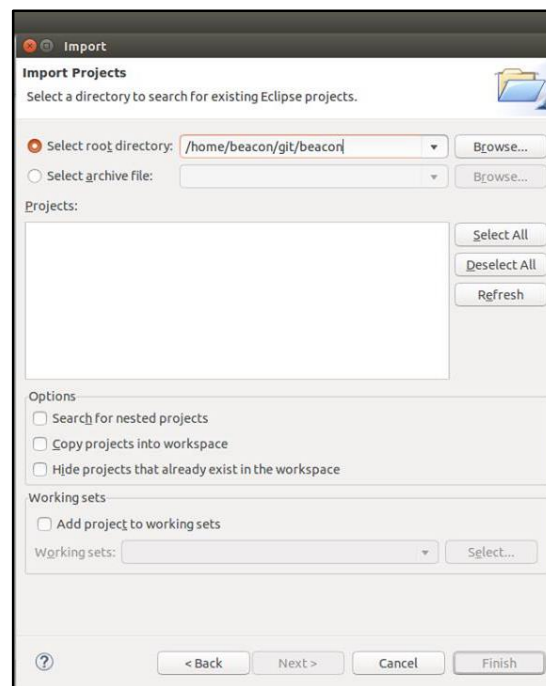
Figura D-5. Lugar donde se importa

En: Select Root Directory agregar el *path* donde se ubicaron los archivos de OpenFlowj:  
/home/controlador/git/openflowj



**Figura D-6.** Importar openflowj

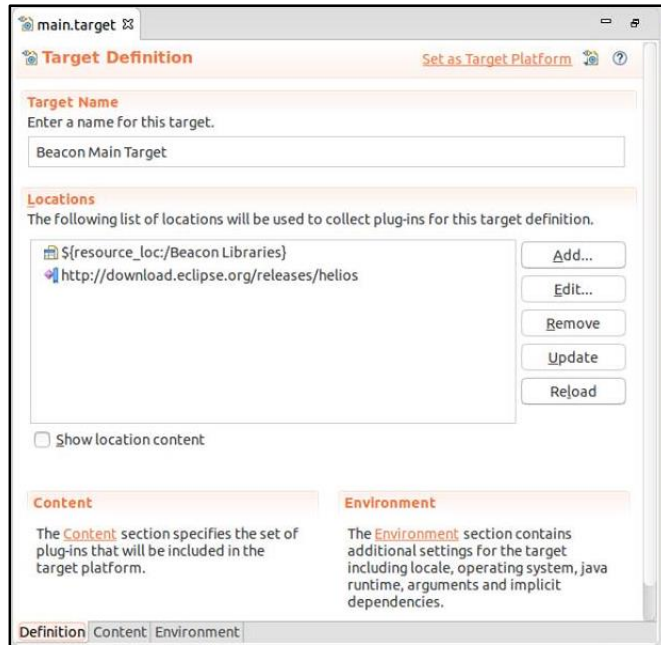
/home/controlador/git/beecon



**Figura D-7.** Importar Beacon

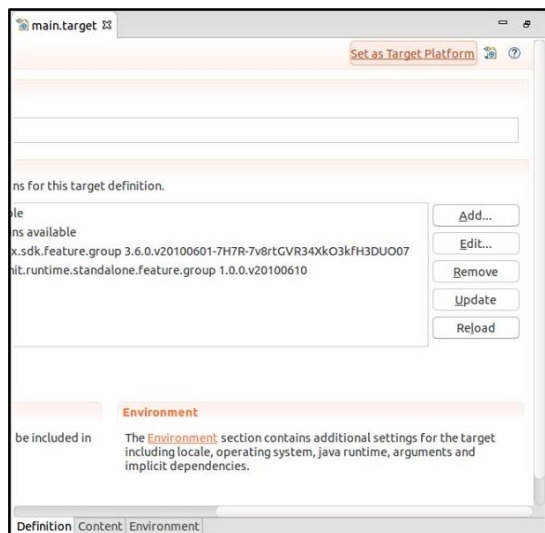
Establecer el destino (bibliotecas ejecutables)

- Abra el proyecto Beacon Main Target, hacer doble clic en el archivo *main.target*.



**Figura D-8.** Archivo main.target

- Una vez abierto, en la esquina inferior derecha de Eclipse se debería ver un mensaje que dice “*Resolving Target Definition*”, esperar a que complete esta operación, después de que el proceso se ha resuelto, hacer clic en “*Set as Target Platform*”, en la esquina superior derecha de la ventana *main.target*.



**Figura D-9.** Set as Target Platform

(Nota: Si se hace clic antes de que se haya resuelto, recibirá un error). En este punto, todos los errores de compilación se habrán ido.

Importar la configuración de estilo de código Beacon

- Hacer clic en: *Window - Preferences*.
- A continuación, en la columna izquierda, clic en: *Java - Code Style - Formatter*.

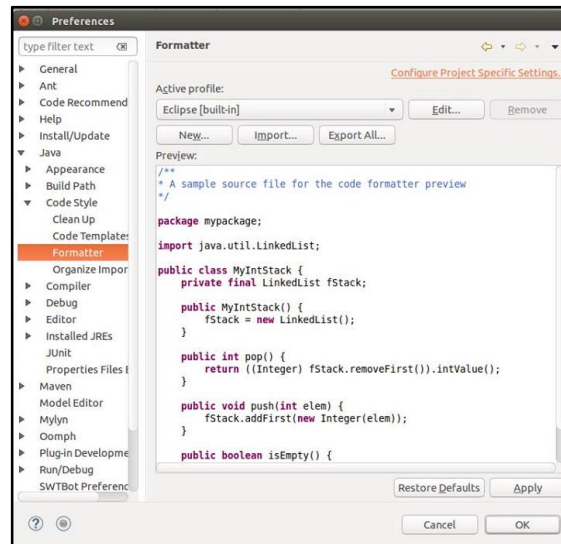


Figura D-10. Formatter

- Hacer clic en el botón *Import* y seleccionar: */git/beacon/beacon\_style\_settings.xml*

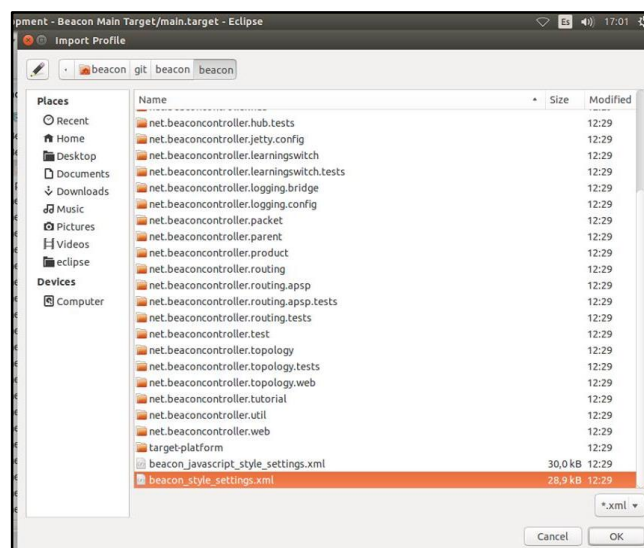


Figura D-11. Elección de archivo

## EJECUTAR BEACON

Ejecutar Beacon en modo de depuración:

- *Run - Debug Configurations*
- Buscar el *OSGi Framework* de la izquierda, expandir el directorio y seleccionar *Beacon*

Clic en *Debug*

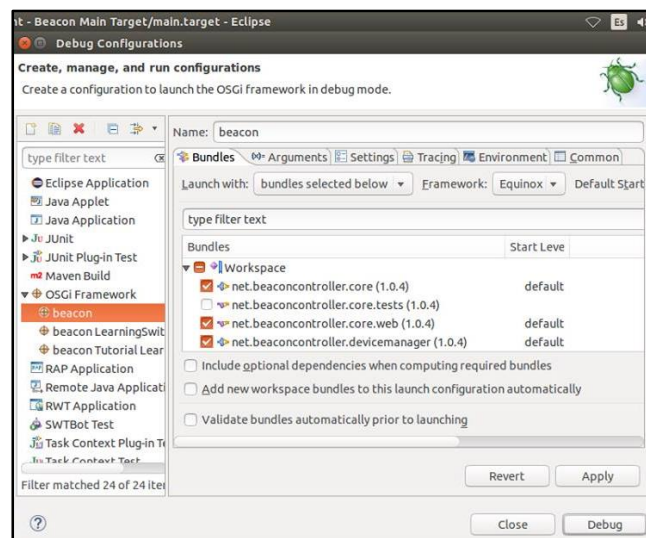
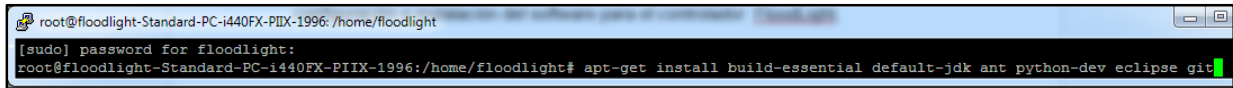


Figura D-12. Debug Beacon

## ANEXO F

### INSTALACIÓN FLOODLIGHT

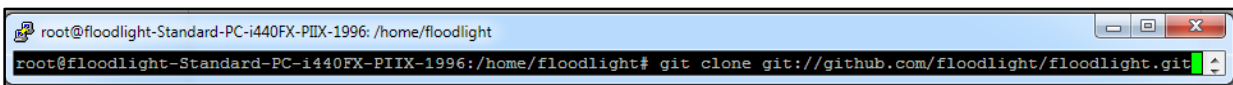
Descargar los repositorios del controlador



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight
[sudo] password for floodlight:
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight# apt-get install build-essential default-jdk ant python-dev eclipse git
```

**Código F-1.** Comando descargar repositorios

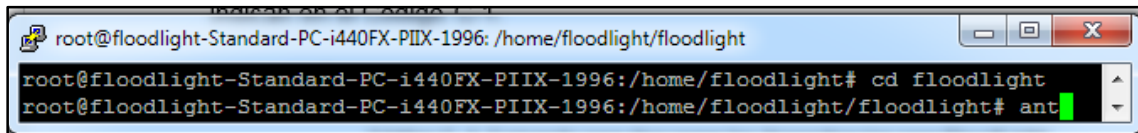
Clonar el proyecto en la carpeta floodlight



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight# git clone git://github.com/floodlight/floodlight.git
```

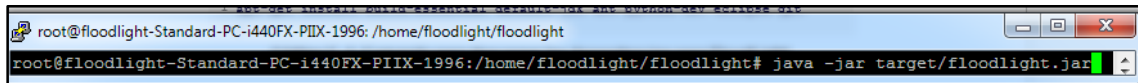
**Código F-2.** Comando clonar proyecto

Ejecución de floodlight



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight# cd floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight# ant
```

**Código F-3.** Ingreso carpeta floodlight



```
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight
root@floodlight-Standard-PC-i440FX-PIIX-1996: /home/floodlight/floodlight# java -jar target/floodlight.jar
```

**Código F-1.** Comando ejecución

## ANEXO G

### ACCESO AL SERVIDOR SDN

Para acceder al servidor SDN, se lo hace de dos formas:

La primera es directamente al servidor e ingresar la contraseña: sdnfica

La segunda es por medio del software Putty, para lo cual previamente se habilita SSH en el servidor y se modifica al puerto 22, luego se abre Putty y se ingresa la dirección del servidor

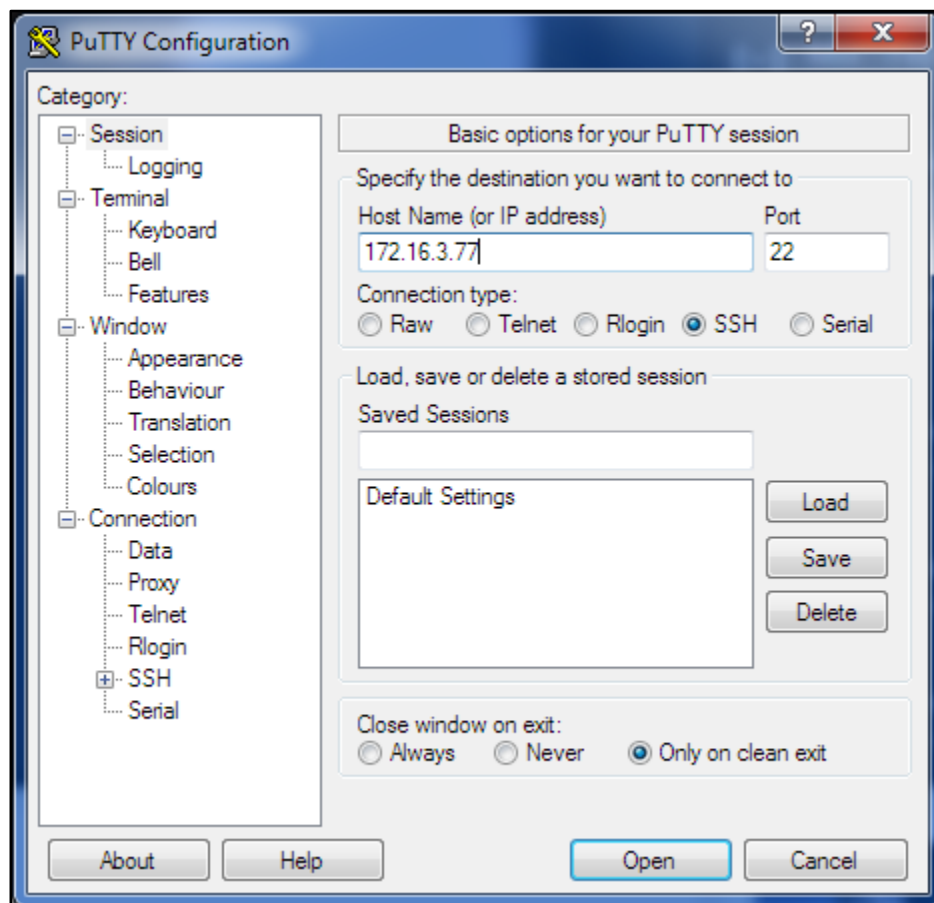


Figura G-1. Ventana de Putty

Al momento de que pide el usuario se ingresa: sdnfica

Y la contraseña es: sdnfica



## ANEXO H

### ACCESO AL SERVIDOR SDN

INTERNATIONAL  
STANDARD

ISO/IEC/  
IEEE  
29148

First edition  
2011-12-01

---

**Systems and software engineering —  
Life cycle processes — Requirements  
engineering**

*Ingénierie des systèmes et du logiciel — Processus du cycle de vie —  
Ingénierie des exigences*



Reference number  
ISO/IEC/IEEE 29148:2011 (E)

© ISO/IEC 2011  
© IEEE 2011

Authorized licensed use limited to: UNIVERSIDAD TECNICA DEL NORTE. Downloaded on February 04, 2016 at 10:57:31 UTC from IEEE Xplore. Restrictions apply.