UNIVERSIDAD DE FUERZAS ARMADAS "ESPE"

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA

TEMA:

DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE UNA RED DEFINIDA POR SOFTWARE (SDN) EN LA UNIVERSIDAD DE LAS FUERZAS ARMADAS "ESPE"

AUTORES:
PABLO FERNANDO ALBÁN RUIZ
DARÍO XAVIER BRITO LÓPEZ

DIRECTOR:
ING. RODOLFO GORDILLO
CODIRECTOR:
ING. CARLOS ROMERO

SANGOLQUÍ, FEBRERO 2015

SUMARIO:

CAPITULO 1: INTRODUCCIÓN
CAPITULO 2: MARCO TEÓRICO
CAPITULO 3: SIMULACIÓN DE LA RED
CAPITULO 4: IMPLEMENTACIÓN DE LA RED
CAPITULO 5: EVALUACIÓN DEL PROTOTIPO
CAPITULO 6: CONCLUSIONES Y RECOMENDACIONES

INTRODUCCIÓN

PRESENTACIÓN:

- Hoy en día las redes de computadoras son de suma importancia en nuestra vida laboral, estudiantil, social, etc. Ya que son un conjunto de equipos de comunicación conectados por medio de cables, señales, ondas o cualquier otro método de transporte de datos, que comparten información, recursos, servicios, etc.

- Uno de los inconvenientes de las redes actuales es que no poseen flexibilidad concerniente a la programabilidad de sus elementos, para que la calidad de servicio sea más dinámica, personalizada y adaptable a las necesidades de la organización en particular.

Objetivo General:

- Diseñar e implementar un prototipo de una red definida por software (SDN) con solución basada en hardware para la Universidad de Fuerzas Armadas ESPE.

Objetivos Específicos:

- Proporcionar una introducción a las redes definidas por software (SDN) y el protocolo Openflow, sus características generales, su arquitectura, ventajas y desventajas.
- Diseñar, abstraer y programar los elementos de la Red mediante la simulación en el programa Mininet. Seleccionar diferentes controladores dependiendo de la operatividad, funcionalidad y facilidad de uso de los mismos.
- Implementar, controlar y gestionar de forma centralizada los dispositivos que comprenden la topología planteada en el diseño de la Red Definida por Software (SDN).
- Realizar la evaluación de la funcionalidad y flexibilidad de la Red Definida por Software (SDN) de la ESPE.

Justificación del proyecto

La ESPE en la actualidad no posee flexibilidad en la administración de su red, por lo tanto se ha considerado la propuesta de implementar un prototipo de redes definidas por software (SDN) y evaluar diferentes controladores, debido a que:

- Las SDN migran el "plano de control" de cada dispositivo a un controlador central que gestiona todos los recursos, permitiendo cambiar el comportamiento de la red dinámicamente mediante software.

- Con la implementación de las SDN se tendrá la posibilidad de automatizar una red y de garantizar confidencialidad de la información, aplicable en los campos sociales, militares, académicos e industriales.

- Con la utilización de esta nueva tecnología, no se encuentra el usuario atado a la adquisición de ciertos equipos por su marca, ya que las SDN son configurables en todos los equipos que soporten el protocolo OpenFlow.

Importancia del proyecto

Es necesario garantizar la flexibilidad de la red con un sistema de control mediante la programabilidad. El prototipo de SDN planteado para la ESPE pondrá a disposición de docentes y estudiantes una plataforma para impulsar el desarrollo de investigaciones en esta nueva tecnología, donde se podrá cambiar el comportamiento de la red dinámicamente, obteniendo una red segura y adaptable. Además se podrá evitar los sistemas de redes cerrados aprovechando la utilización de software libre.

Alcance del proyecto

- Levantamiento de la información de la arquitectura de las SDN y el protocolo OpenFlow, software Mininet y controladores SDN.
- Instalación del entorno de simulación Mininet, estudio y simulación de varias opciones de controladores. Crear la infraestructura de la topología a simular y evaluar los diferentes controladores dependiendo de las aplicaciones requeridas.
- Instalación de la topología de red, conforme el estudio del proyecto del Consorcio Ecuatoriano Para el Desarrollo de Internet Avanzado (CEDIA), con equipos reales que soporten el protocolo OpenFlow, en las instalaciones de la ESPE.
- Evaluación del funcionamiento de la red mediante pruebas y análisis de resultados tanto en la simulación como en la implementación de la red.

MARCO TEÓRICO

Características generales de las SDN

- Las SDN se ajustan y responden de forma dinámica bajo políticas actualizadas, reducir el trabajo manual y los costos.
- Las redes han estado limitadas por la forma en que el software las ha configurado, suministrado y gestionado.
- Estudio de: protocolo OpenFlow, entorno de simulación Mininet y los diferentes controladores.

Definición

- Las SDN son aquellas que brindan la posibilidad de llevar a cabo un control mucho más directo de su comportamiento y posibilitan la interactuación directa con la red como si fuera un todo. El plano de control y el plano de datos.
- Redes más maleables y escalables que se adaptan a cambios.
- Las SDN permiten mejoras en la seguridad y monitorización.

Arquitectura SDN

- Componentes: Infraestructura de Red, Controlador y Aplicaciones.
- Se hace relación a los dispositivos de red: routers y switches, físicos (hardware) y/o virtuales (software).
- El Controlador es el software instalado en un servidor que controla de una forma lógica y centralizada a la red.
- Las Aplicaciones de voz, video, servicios de red pueden comunicarse con el controlador usando una API abierta.
- OpenFlow es el protocolo empleado para comunicación entre el controlador (plano de control) y el switch (plano de datos).

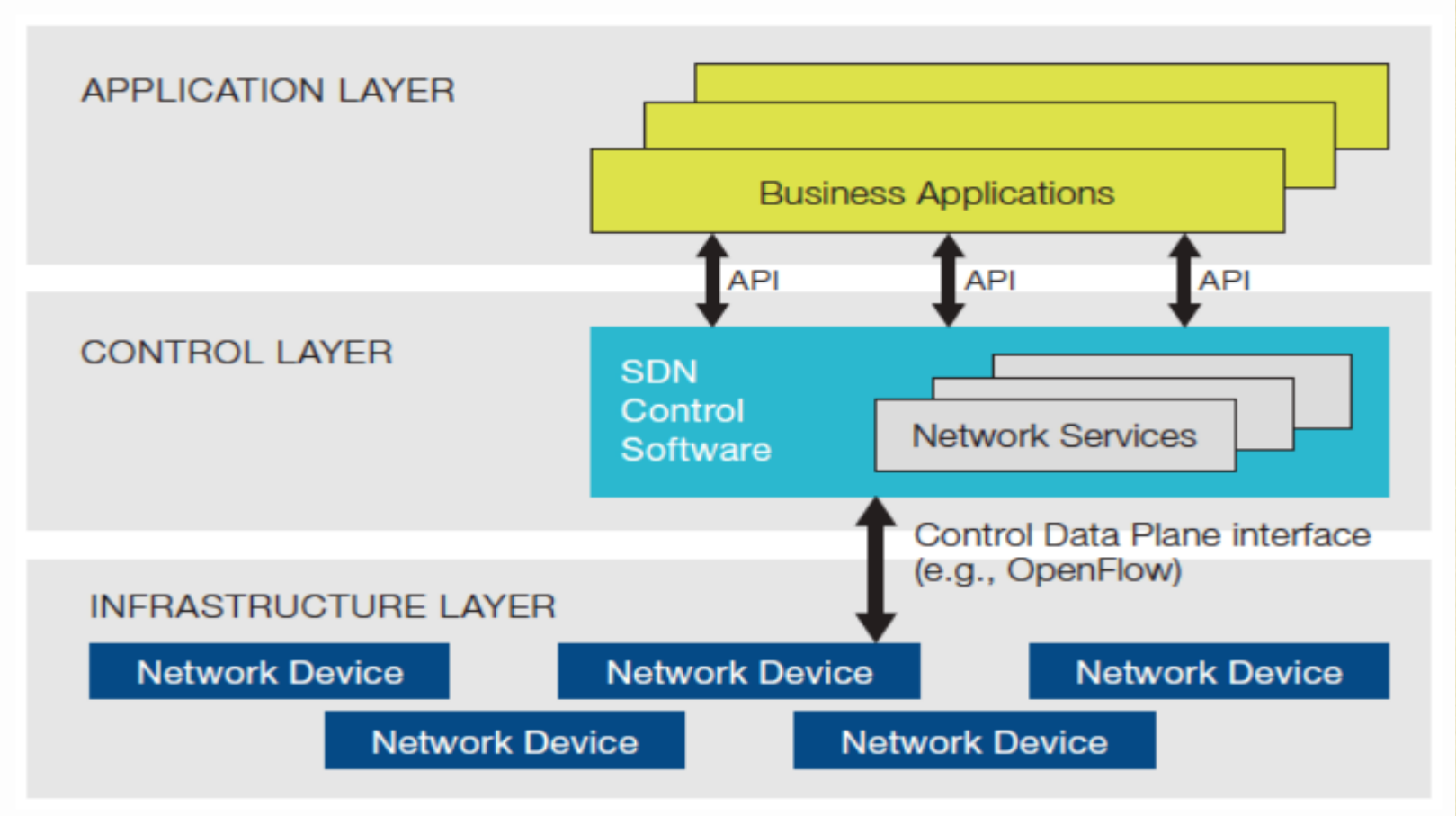


Figura 1. Arquitectura SDN (Principia technologica, 2014)

Ventajas de la arquitectura SDN

- Dividen el software en cuatro niveles: administración, servicios, control y reenvío.
- Concentran la información de los niveles de administración, servicios y control abreviando el diseño de red.
- Implementación flexible, adaptable y escalable.
- Implantan un escenario para las aplicaciones, los servicios y la integración de la red en los sistemas de administración.
- Admiten una estandarización de los protocolos, elección entre proveedores, reducción de costos y diversa asistencia.
- Seguridad.

Servidor Controlador SDN

- Es la parte central de la arquitectura SDN, mantiene toda la inteligencia de la red. Se encarga de definir reglas para administrar el flujo de datos en la red.

- El controlador ejecuta un software que le permite llevar a cabo sus tareas. NOX, POX, Beacon, Floodlight u otro.

Arquitectura de red tradicional vs arquitectura SDN

- En las arquitecturas tradicionales gran parte del tráfico de red viaja hacia los clientes, donde reside parte de la lógica de las aplicaciones, por lo que estas redes no están preparadas para ese esquema de tráfico, porque son arquitecturas jerárquicas y a pesar de ser estables también son altamente estáticas, lo que impide flexibilizar la red. Mayor tráfico implica flexibilizar la red para estas nuevas necesidades.

El protocolo OpenFlow

El emergente estándar abierto de OpenFlow, definido por la Open Networking Foundation (ONF) desde 2011 y es el protocolo que permite acceder directamente y manipular el plano de redireccionamiento de dispositivos de red como switches y enrutadores, ya sean físicos o virtuales. La interfaz de OpenFlow ofrece acceso y comunicación entre las capas de control e infraestructura de la arquitectura de SDN, tanto física como virtual. Al centralizar el control de los dispositivos de capas de infraestructuras, OpenFlow simplifica la administración de las redes y la capacidad de programación que promete SDN (Azodolmolky, 2013).

Características:

- Simplifica la administración de redes y la programación de dispositivos de redes.
- Permite cambios dinámicos en el flujo de tráfico.
- Permite que la red tenga mayor capacidad de respuesta para las necesidades empresariales.

Conmutador OpenFlow

Un conmutador OpenFlow trabaja con lo que se denominan tablas de flujos y aprovecha que la mayoría de los switches y routers Ethernet modernos contienen dichas tablas de flujos. A pesar de que cada flow-table es diferente para cada fabricante, se han identificado un conjunto de funciones comunes o genéricas entre ellos.

El camino de datos de un conmutador OpenFlow consiste en una flow- table y una acción asociada a cada entrada. El conjunto de acciones soportadas es extensible, pero existe un conjunto de requerimientos mínimo para todos los switches. Para obtener un alto rendimiento y un bajo coste.

Conmutador OpenFlow

Una flow-table se compone de distintas entradas. Cada entrada en la flow-table contiene los siguientes campos:

- Match Fields
- Priority
- Counters
- Instructions
- Timeout
- Cookie

Conmutador OpenFlow

- Cada flujo de entrada (flow-entry) es analizado en la tabla:
- 1. En primer lugar se busca el paquete entrante coincidente con la mayor prioridad.
- 2. Una vez detectado se aplican las distintas instrucciones.
- 3. Por último, se envía el dato coincidente a la siguiente tabla (flow-table) junto con el conjunto de acciones.

Mensajes OpenFlow

- Los mensajes OpenFlow son los que permiten transmitir acciones e información desde el servidor controlador hacia el o los dispositivos de red hacia el controlador. Se dividen en: mensajes del controlador al conmutador, mensajes asincrónicos y mensajes simétricos (Azodolmolky, 2013).

Mensajes del controlador al conmutador

Mensajes remitidos por el controlador y no precisamente requieren de una respuesta por parte del conmutador .

- Features
- Configuration
- Modify-State
- Read-State
- Packet-Out
- Barriel

Mensajes asincrónicos

Estos mensajes son generados por los conmutadores cuando reciben un paquete y ha ocurrido un cambio en su estado o ha ocurrido un:

- Packet_In
- Flow-Removed
- Port-Status
- Error

Mensajes simétricos

Estos mensajes se envían en cualquier dirección sin una solicitud previa. En esta clasificación se encuentran los siguientes mensajes:

- Hello
- Echo
- Experimenter

Python

- Python es uno de los idiomas más fáciles de computadora para entender, aprender y utilizar, debido a su la sintaxis legible, similitud con otros lenguajes orientados a objetos, y muchas bibliotecas útiles (Doxygen, 2015).

Mininet

- Existen varios programas de simulación, se puede crear varias topologías de red con equipos virtuales (Switches, hosts y controladores SDN).
- Otros simuladores de red: OpenFlow VMS o Netkit pero a diferencia de Mininet, estos no son tan robustos. Mininet dispone además de rapidez, se consigue crear topologías y ponerlas a trabajar en tan solo unos segundos. Se escriben o reutiliza códigos en Python.

Controladores SDN

- Son el motor de todas las políticas o acciones que se debe tomar en una red SDN. Ofrece servicios que pueden realizar un plano de control distribuido que se han creado con código abierto.
- La diferencia entre cada controlador está en el lenguaje de programación y la plataforma en la que éste trabaje, pero la funcionalidad es la misma ya que todos tienen que transmitir y recibir paquetes OpenFlow para comunicarse con los dispositivos SDN.

POX (Python)

- POX es la evolución del controlador NOX clásico, particularmente éste controlador permite un rápido desarrollo y creación de prototipos de software de control de red usando el lenguaje de programación Python.
- Para un correcto funcionamiento POX requiere de la versión 2.6 o 2.7 de Python y puede ser ejecutado en prácticamente cualquier sistema operativo que tengan o soporten dichas versiones de Python.
- POX actualmente trabaja con switches que soportan la versión 1.0 del protocolo OpenFlow.

Pyretic (Python)

- Pyretic permite a los administradores de red realizar aplicaciones modulares y robustas, reutilizando módulos y código.
- Pyretic también soporta la ejecución en paralelo pero ésta tiene que ser secuencial.
- Se está añadiendo a Pyretic características como el soporte para calidad de servicio y se están desarrollando nuevas aplicaciones, incluyendo servidores RADIUS y DHCP, incluso se está mejorando la eficacia de su compilador y la verificación de la generación de código correcto.

PyResonance (Python)

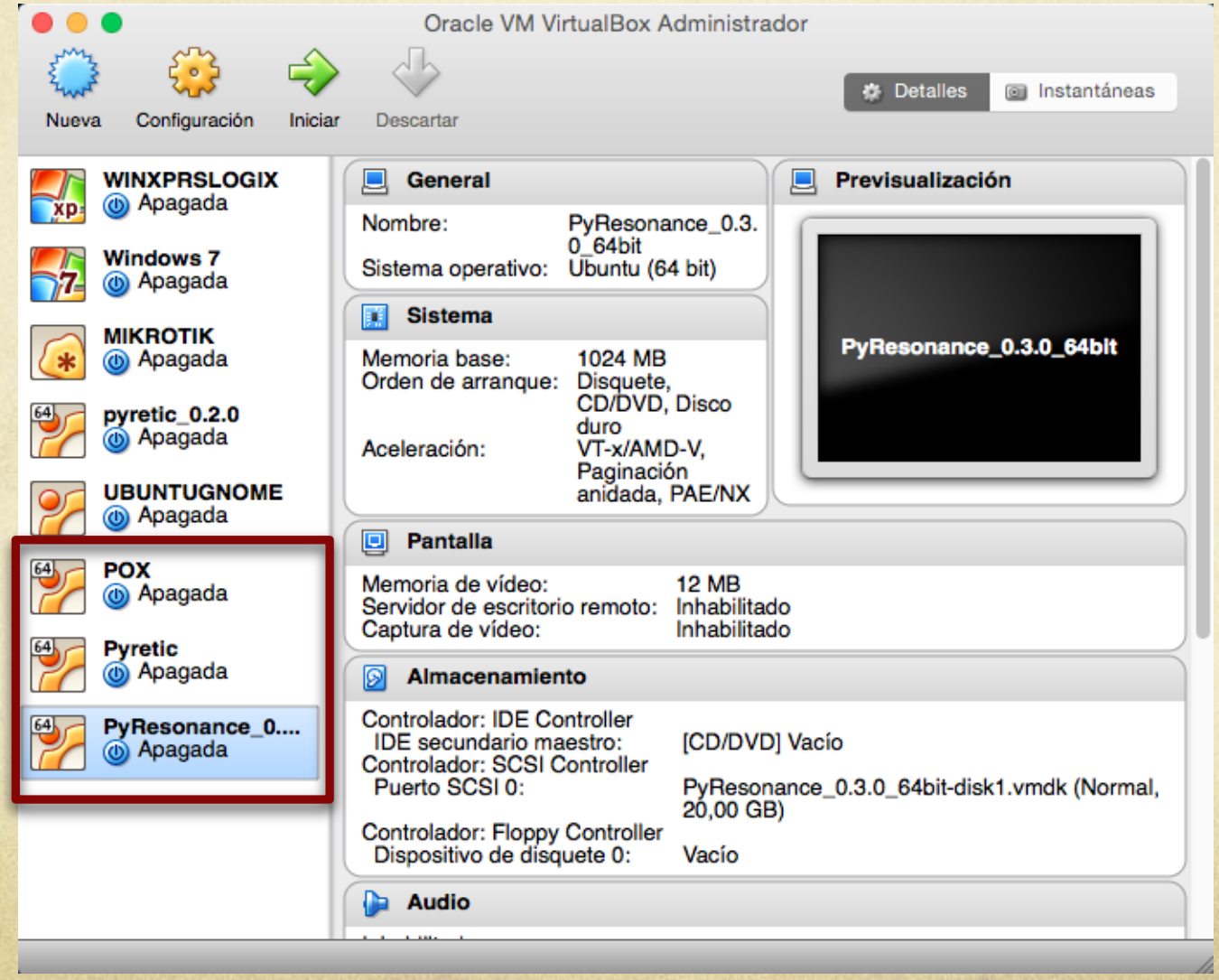
- PyResonance es una plataforma Resonance implementada con Pyretic.
- Resonance es una plataforma de control SDN que se implementó inicialmente con NOX, y que trabaja en un plano de gestión donde los administradores de red implementan las reglas de control como una máquina de estados finita.
- PyResonance está construido en la cima de Pyretic, en un entorno atractivo y atrayente para la implementación de los controladores basados en resonancia.

SIMULACIÓN DE LA RED (Mininet)

Entorno de simulación (hardware y software)

- VirtualBox con la versión 4.3.20
- Máquinas virtuales que contienen Linux, y que a su vez tienen instaladas la versión 2.1.0 de la herramienta de simulación de Mininet.
- Cliente SSH. Putty (Windows)
- X Server. Xming (Windows) y Xquartz (OS X)

Entorno de simulación (VirtualBox)



Topologías usadas en la simulación

- **Single o única**. Consiste de un único conmutador conectado a un número determinado de hosts.

$sudo mn --topo single, N

- **Linear o lineal**. Esta topología consta de un determinado número de conmutadores interconectados de forma lineal, donde cada uno de ellos tiene un host conectado.

$sudo mn --topo linear, N

- **Tree o árbol**. Topología tipo árbol donde hay dos parámetros para tomar en cuenta, la profundidad (depth) y la distribución (fanout).

$sudo mn --topo tree, depth=N, fanout=M

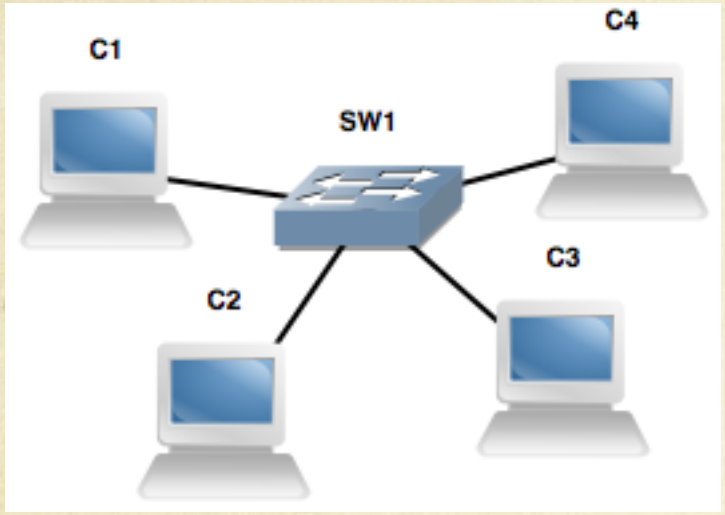
Topologías usadas en la simulación

Custom o Personalizada. Para topologías personalizadas es necesario:

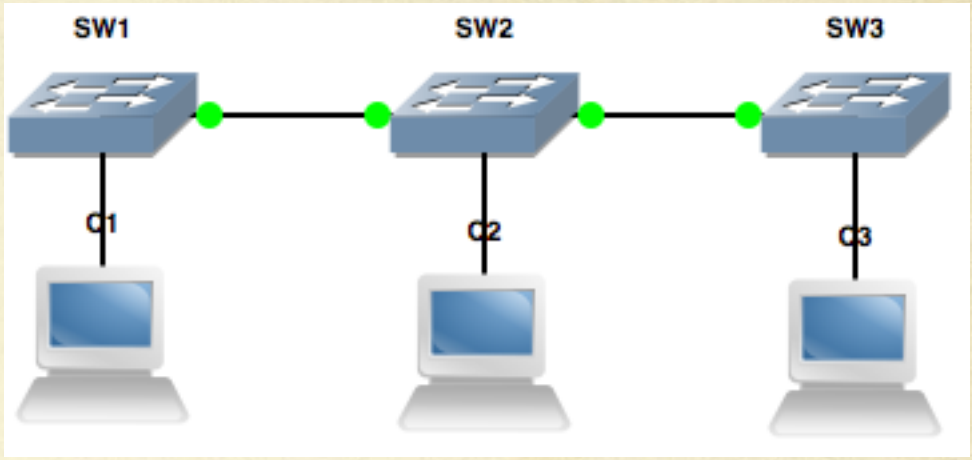
- Escribir un script en python.
- Conocer la utilización de algunas clases, métodos, funciones y variables importantes para realizar el script.
- Colocar al script en una carpeta dentro de mininet
- Finalmente ejecutar el siguiente comando:

$ sudo python nombre_de_archivo.py

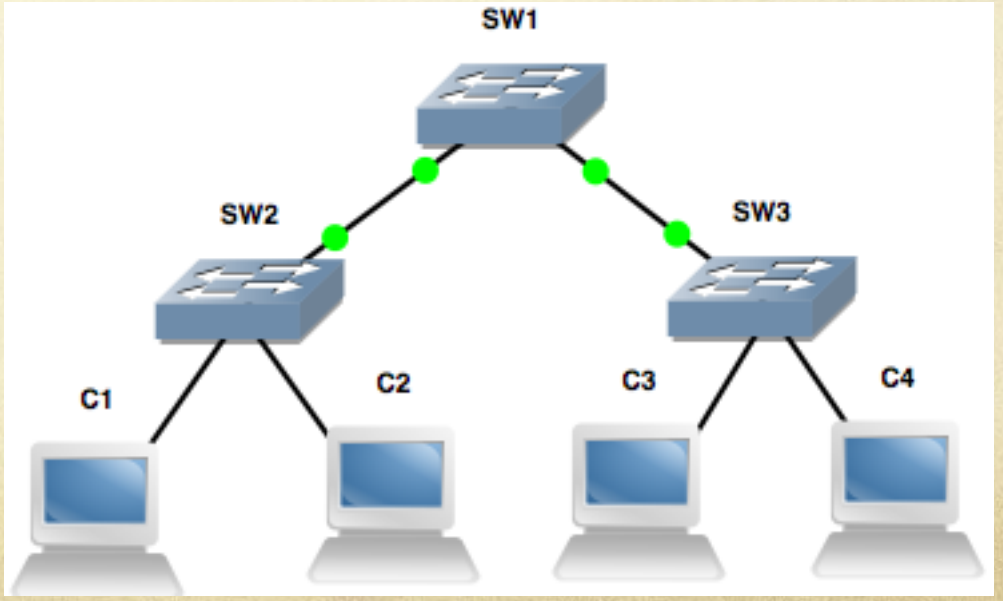
Topologia Single
$sudo mn --topo single,4



Topologia Linear
$sudo mn --topo linear,3



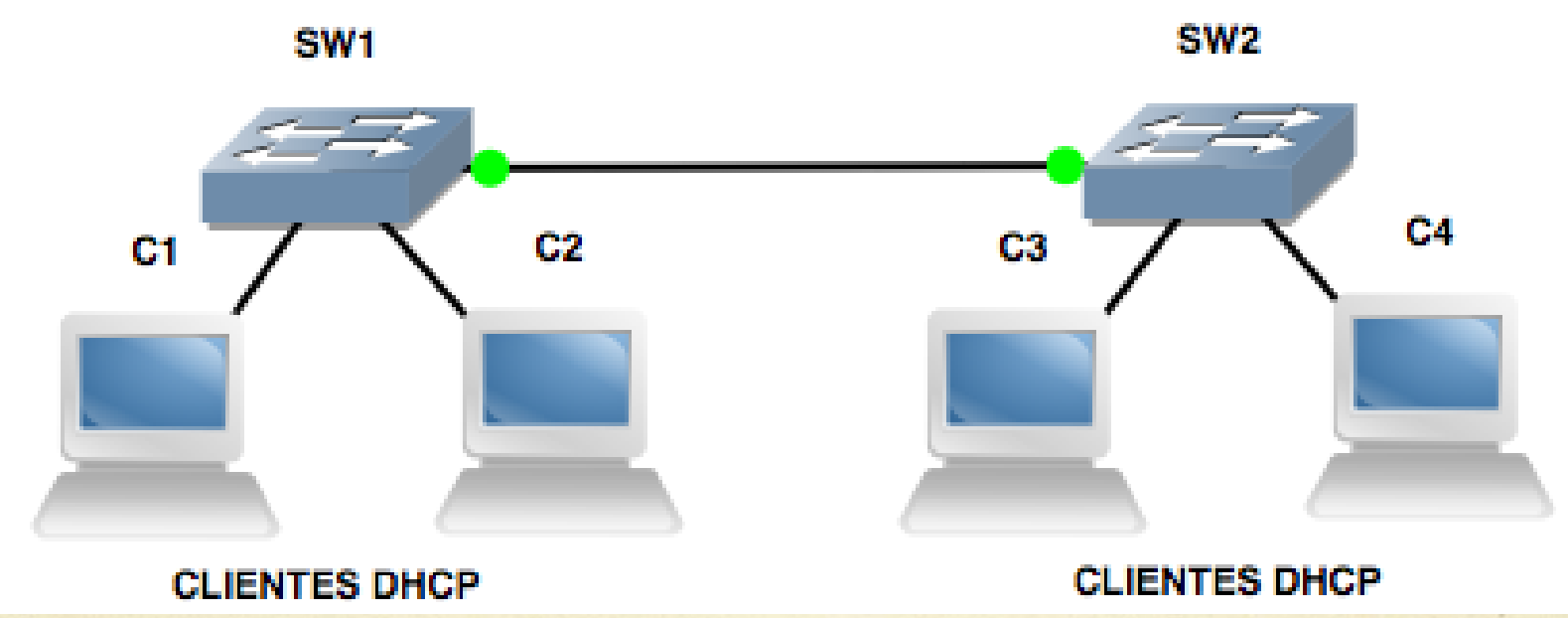
Topologia Tree
$sudo mn --topo tree, depth=2, fanout=2



Importantes clases y métodos

- **MiniNet ():** clase principal para crear y administrar una red.
- **AddSwitch ():** añade un conmutador en la topología.
- **AddHost ():** añade un host a la topología y devuelve el nombre de host.
- **AddLink ():** añade un enlace entre dispositivos.
- **start ():** inicia la red.
- **pingAll ():** prueba la conectividad al tratar de tener ping entre todos los nodos.
- **stop ():** detiene la red.
- **net.hosts:** todos los anfitriones en una red.

Topología personalizada



$sudo phyton topologiaSDN.py

Topología personalizada (*topologiaSDN.py*)

```
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import RemoteController
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

def topologia_SDN():

 net = Mininet()

 # Creacion del controlador
 net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6633)

 # Creacion de hosts
 h1 = net.addHost('h1')
 h2 = net.addHost('h2')
 h3 = net.addHost('h3')
 h4 = net.addHost('h4')

 # Creacion de Switches
 s1 = net.addSwitch('s1')
 s2 = net.addSwitch('s2')
```

Topología personalizada (*topologiaSDN.py*)

```
# Creacion de enlaces
net.addLink(h1, s1)
net.addLink(h2, s1)
net.addLink(h3, s2)
net.addLink(h4, s2)
net.addLink(s1, s2)

# Inicio de funcionamiento
net.start()
# Configuracion de clientes dhcp
h1 = net.get('h1')
h1.cmd('dhclient -r')
h1.cmd('dhclient h1-eth0')
h2 = net.get('h2')
h2.cmd('dhclient -r')
h2.cmd('dhclient h2-eth0')
h3 = net.get('h3')
h3.cmd('dhclient -r')
h3.cmd('dhclient h3-eth0')
h4 = net.get('h4')
h4.cmd('dhclient -r')
h4.cmd('dhclient h4-eth0')
```

Topología personalizada (*topologiaSDN.py*)

```
print "Registro de las conexiones de los hosts"
dumpNodeConnections(net.hosts)
CLI(net)
net.stop()
```

Comandos Mininet

- Para salir de mininet: **mininet> exit**
- Para abrir un terminal para el nodo**: mininet> xterm [nodo]**
- Para crear o eliminar un o link entre dos nodos:

  mininet> link [node1][node2][up or down]
- Para prueba de conectividad entre dos nodos:

  mininet> pingall
- Ayuda, muestra lista de comandos de mininet: **mininet> help**
- Para limpiar la topología: **$ sudo mn –c**

Comandos Mininet

Cuando vamos a ejecutar un comando en mininet debemos saber la sintaxis del mismo por ejemplo si queremos realizar ping de un nodo a otro:

- Debemos colocar el nodo origen
- Luego el comando ping
- Finalmente el nodo destino

Por ejemplo:

mininet> h2 ping h3

- Es posible utilizar el nombre del nodo para sustituir la IP.

IMPLEMENTACIÓN DE LA RED SDN

Topología

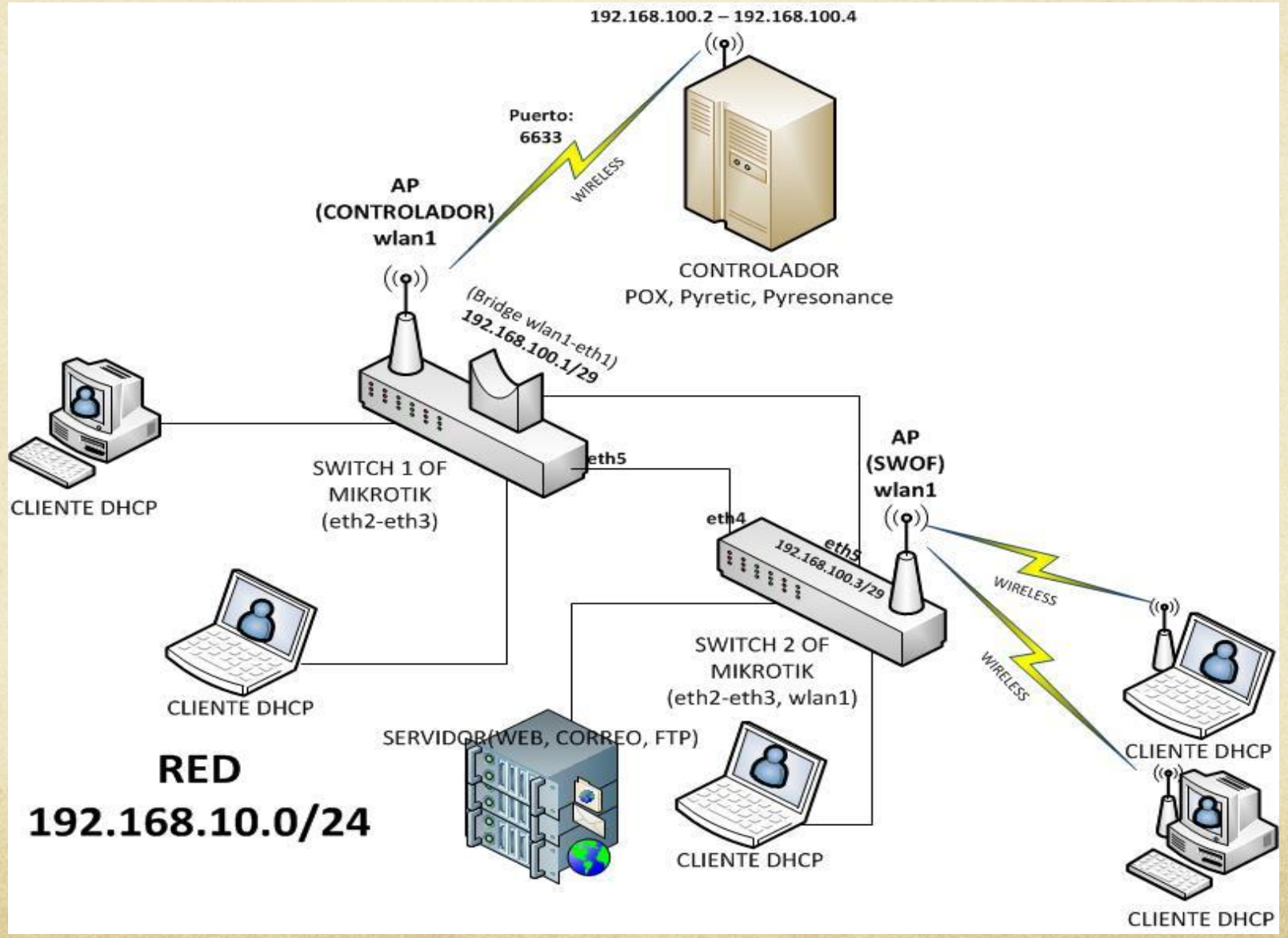

192.168.100.2 – 192.168.100.4
Puerto:
6633
WIRELESS
AP
(CONTROLADOR)
wlan1
CONTROLADOR
POX, Pyretic, Pyresonance
(Bridge wlan1-eth1)
192.168.100.1/29
CLIENTE DHCP
SWITCH 1 OF
MIKROTIK
(eth2-eth3)
eth5
AP
(SWOF)
wlan1
eth4
eth5
192.168.100.3/29
WIRELESS
WIRELESS
CLIENTE DHCP
CLIENTE DHCP
SWITCH 2 OF
MIKROTIK
(eth2-eth3, wlan1)
SERVIDOR(WEB, CORREO, FTP)
CLIENTE DHCP
CLIENTE DHCP
RED
192.168.10.0/24

Tabla de direccionamiento

Dispositivo	Interfaz	Dirección IP	Máscara	Red
SW1	WLAN1	192.168.100.1	255.255.255.248	BRIDGE
	ETH1			
	ETH2	DHCP	DHCP	ETHERNET LAN
	ETH3			
	ETH5			
	ETH4	-	-	WINBOX
SW2	ETH5	192.168.100.3	255.255.255.248	LAN
	ETH2	DHCP	DHCP	ETHERNET LAN
	ETH3			
	ETH4			
	WLAN1	DHCP	DHCP	WLAN
	ETH1	-	-	WINBOX
CONTROLADOR POX	ETH1	192.168.100.2	255.255.255.248	ETHERNET
CONTROLADOR PYRETIC	ETH1	192.168.100.4	255.255.255.248	ETHERNET
CONTROLADOR PYRESONANCE	ETH1	192.168.100.5	255.255.255.248	ETHERNET

Plataforma de hardware

- El servidor controlador se implementó en un computador MacBookPro.
- Se utilizó dos router Mikrotik RB951Ui-2HnD.
- Los host probados tienen un sistema operativo Windows 7 y estos a su vez poseen máquinas virtuales que emulan más hosts que se conectan a la red.

Software del servidor controlador

- Existen varios controladores SDN que se diferencian comunmente por el lenguaje de programación con el que trabajan, en este proyecto se analizarán tres tipos de controladores que trabajan con python y que se instalarán en máquinas virutales con Linux como sistema operativo, los controladores son: POX, Pyretic y PyResonance (Kinetic, en su ultima versión)

Maquinas Virtuales

- La máquina virtual que se utilizó es la misma tanto para POX como para Pyretic y podemos bajarla de:

  http://www.frenetic-lang.org/pyretic/

- En esta máquina virtual vienen ya preinstalados los dos controladores, y viene lista para ejecutarla en cualquier motor de máquinas virtuales, en este caso VirtualBox.

- Para el caso de Pyresonance o Kinetic, podemos bajar la máquina virtual con el controlador instalado de:

  http://resonance.noise.gatech.edu/

Controlador POX

- Es una plataforma de software escrita en python, que tiene módulos o componentes escritos en el mismo lenguaje, para ejecutar POX basta con tener instalado python e iniciar el componente principal pox.py, el cuál se encargará de reconocer el nombre de todos los módulos a ejecutar y realiza la llamada de cada uno de ellos, para que el controlador POX finalmente se ejecute correctamente.

Controlador POX

Los módulos son localizados en todos los directorios que constan en la carpeta en la que se ha instalado POX, por lo tanto si queremos ejecutar un módulo tenemos que especificar a qué carpeta pertenece, y en el caso de que el módulo se encuentre en el directorio raíz de POX, no será necesario especificar dicha carpeta, por ejemplo:

- El módulo ejemplo se encuentra en el directorio raíz de POX

  $ pox.py ejemplo

- El módulo l2_learning se encuentra en la carpeta forwarding

  $ pox.py forwarding.l2_learning

- Si ejecutamos los dos módulos simultaneamente

  $ pox.py ejemplo forwarding.l2_learning

Controlador POX

Dependiendo de cómo estén estructurados los módulos, pueden tener ciertas opciones o parámetros y éstos representan valores a ciertas variables que se tenga en el código

Por ejemplo si queremos ejecutar el módulo of_01 Con valores por defecto:

```
$ pox.py openflow.of_01
```

Con valores distintos, se añade los parámetros y sus valores después del módulo:

```
$ pox.py openflow.of_01 --port=6644 --address=192.168.100.1
```

Controlador POX

Para poder realizar un módulo personalizado es importante conocer algunos módulos y APIs preestablecidos, que se utilizan por ejemplo para el manejo de direcciones IPv4, IPv6 y Ethernet, para el envío y recepción de paquetes de los protocolos como DHCP, IPv4, ICMP,TCP, etc., para controlar eventos que existen en una clase o una superclase, para el manejo de los DPIDs, o bien para el manejo de los logs (mensajes de información).

Debemos conocer también funciones importantes como la función launch, la función ConnectionUp, etc.

Controlador POX

La función PacketIn que se activa cuando el controlador recibe un mensaje Openflow desde el switch y maneja ciertos eventos para indicar que un paquete ha llegado o no a un puerto de un switch. En la siguiente página se encuentra toda la información y algunos ejemplos para realizar un módulo personalizado:

https://openflow.stanford.edu/display/ONL/POX+Wiki

Módulo personalizado con POX

Si queremos realizar un módulo, que permita que un switch sea despreciado, el código será:

```
from pox.core import core
from pox.lib.util import str_to_dpid
from pox.forwarding.l2_learning import LearningSwitch

def launch (idswitch):
        idswitch = str_to_dpid(idswitch)
  def _handle_ConnectionUp (event):
        if event.dpid != idswitch:
            LearningSwitch(event.connection, True)
        if event.dpid == idswitch:
            core.getLogger().info("Switch Despreciado: %s" %
            (event.connection,))
core.openflow.addListenerByName("ConnectionUp",
_handle_ConnectionUp)
```

Controlador Pyretic

Al igual que POX este controlador está implementado en python, tiene módulos escritos en el mismo lenguaje, para ejecutarlo basta con tener instalado python e iniciar el componente principal pyretic.py, el cuál se encargará de reconocer el nombre de todos los módulos a ejecutar y realiza la llamada de cada uno de ellos, para que el controlador Pyretic finalmente se ejecute correctamente.

Si queremos ejecutar un módulo tenemos que especificar a qué carpeta de pyretic pertenece, por ejemplo:

El módulo mac_learner se encuentra en el directorio pyretic/modules, por lo tanto se utilizará el comando:

$ pyretic.py pyretic.modules.mac_learner

Controlador Pyretic

Una política en Pyretic, es una función que toma un paquete como entrada y devuelve un conjunto de paquetes. Ésta función indica que es lo que un switch debe hacer con los paquetes entrantes. En fin todas las políticas existentes en Pyretic tienen un objetivo específico, por ejemplo el manejo de un switrch o de los puertos de un switch, el manejo de direcciones IP y direcciones MAC, el control de flujo entre los switches de la red, etc.

Controlador Pyretic

Política	Sintaxis	Descripción	Ejemplo
match	match(f=v)	Si f es igual a v, se permite el intercambio de paquetes, según se establezca la condición	match(switch=4)
drop	drop	Retorna un conjunto de paquetes vacíos	drop
flood	flood()	Se permite el flujo de paquetes por cada puerto local en el switch	flood()
Composición Secuencial	A >> B	Retorna la salida de B donde la salida de A es la entrada de B.	fwd(1) >> fwd(2)
Negación	~ A	Retorna la negación lógica de una política establecida en este caso A	~match(switch=1)

Controlador Pyretic

Para realizar un módulo personalizado, se debe por lo menos conocer el funcionamiento de las políticas básicas de Pyretic. Estas políticas nos simplifican el código, dependiendo de lo que queramos realizar, por ejemplo: en el controlador POX se necesita programar muchas líneas de código, para realizar un módulo que funcione como un componente hub, a diferencia que en Pyretic para realizar el mismo componente simplemente se agrega la política flood.

Módulo personalizado con Pyretic

Si queremos realizar un módulo que permita seleccionar un switch y despreciarlo, podemos escribir el código:

```
from pyretic.lib.corelib import *

from pyretic.lib.std import *

from pyretic.modules.mac_learner import mac_learner

def main (idswitch):

        print "Switch Openflow %s Despreciado" % idswitch

return ~match(switch=double(str(idswitch))) >> mac_learner()
```

Controlador Pyresonance

Esta implementado con Pyretic, por lo tanto para ser ejecutado tenemos que tener instalado tanto el controlador Pyretic como python, y lo iniciamos con el componente pyretic.py

Los módulos son localizados de la misma forma que en POX y en Pyretic, por lo tanto tenemos que especificar a qué carpeta pertenece, por ejemplo:

El módulo ids se encuentra en el directorio pyretic/pyresonance/apps, por lo tanto para ejecutar el módulo se utiliza el comando:

$ pyretic.py pyretic.pyresonance.apps.ids

Controlador Pyresonance

- Pyresonance permite un control dinámico en la red SDN por eventos, es decir este controlador reacciona a diferentes tipos de acontecimientos que se presentan en la red, cambiando la política de red aplicada dinámicamente.
- Un módulo personalizado de Pyresonance se implementa igual que un módulo de Pyretic, donde se puede construir múltiples políticas de red y componerlas entre sí secuencial o paralelamente, expresando una política general para la red SDN.

Controlador Pyresonance

Un módulo en Pyresonance debe tener la siguiente estructura:

- La función LPEC
- Las funciones de transición que son aquellas que asignan un valor que una variable debe tomar cuando llega un evento en particular en el controlador.
- La máquina de estados finita (FSM) que es la que asocia las funciones de transición que definimos con las variables de estado correspondientes.
- Las políticas y los eventos. Pyretic automáticamente dirige cada evento de entrada a la adecuada LPEC FSM, donde estará a cargo de la función de transición que se especifica en la definición de la FSM (por ejemplo, fsm_desccription)

Módulo personalizado con Pyresonance

- Con la estructura anteriormente mencionada, podemos escribir el código para un módulo de Pyresonance, donde un switch de la red SDN pase a un estado de infectado y por lo tanto el controlador no permita el flujo de paquetes en ese switch en específico, es decir lo desprecie porque está infectado.

Módulo personalizado con Pyresonance

```
from pyretic.lib.corelib import * # Importamos librerias de Pyretic
from pyretic.lib.std import *

from pyretic.pyresonance.fsm_policy import * # Importamos la Politica FSM
from pyretic.pyresonance.drivers.json_event import JSONEvent # Importamos librerias JSON
#from pyretic.pyresonance.smv.translate import *
from pyretic.modules.mac_learner import mac_learner
# Importamos la funcion mac_learner del modulo mac_learner

class seleccionar(DynamicPolicy):
      def __init__(self):

       ### SE DEFINE LA FUNCION LPEC
        def lpec(f):
          return match(switch=f['switch']) # Se escoge un switch

       ## INSTALAMOS LAS FUNCIONES DE TRANSICION
        def infected(event):
          return event

        def policy(state):
            if state['infected']:
                return drop # No se permite el flujo de paquetes
            else:
                return identity # Se permiten los paquetes originales

        ### DEFINIMOS LA FSM DONDE SE ANALIZA SI ESTA O NO INFECTADO UN SWITCH
```

Módulo personalizado con Pyresonance

```
        self.fsm_description = {
              'infected' : (bool,
                            False,
              'policy'   : ([drop,identity],
                            NextFns(state_fn=policy)) }

        ### SE DEFINE LA POLITICA DE LA FSM Y SE PASAN LOS VALORES DE LOS EVENTOS

        fsm_pol = FSMPolicy(lpec,self.fsm_description)
        json_event = JSONEvent()
        json_event.register_callback(fsm_pol.event_handler)

        super(seleccionar,self).__init__(fsm_pol)

def main():

    return seleccionar() >> mac_learner()
```

Análisis de resultados

Comparación de resultados entre la simulación e implementación de la red SDN

CONTROLADOR	RED SDN	LATENCIA MIN	LATENCIA MAX	LATENCIA PROM
POX	SIMULADA	0,043	39,724	19,8835
Pyretic		0,031	70,897	35,464
PyResonance		0,047	101,816	50,9315
POX	FÍSICA	2	510	256
Pyretic		1	138	69,5
POX y Pyretic		0	883,409	441,7045
PyResonance		2	340	171
POX y PyResonance		0	492	246

Análisis de resultados

- En la implementación al igual que en la simulación se obtiene un mejor rendimiento de la red utilizando el controlador Pyretic, ya que el nivel de procesamiento al instalar o actualizar una regla en la tabla de flujos es más rápido.
- Cuando se hace coexistir dos o más controladores notoriamente el nivel de procesamiento es mayor, pero solo cuando se instala por primera vez una regla en la tabla de flujos, posterior a esto se puede observar que el tiempo de respuesta es prácticamente instantáneo.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Al finalizar el presente Proyecto se dispone del diseño e implementación de un prototipo de una red definida por software (SDN) para la ESPE, compuesta por un servidor controlador, dos conmutadores y seis hosts.

- En el presente proyecto se proporciona una introducción a las redes definidas por software (SDN) y el protocolo Openflow, sus características generales, su arquitectura, ventajas y desventajas, sobresaliendo tendencias como la movilidad del usuario, la virtualización de servidores, y la necesidad para responder a las cambiantes condiciones en la red, que significan demandas sobre las redes.

Conclusiones

- Las SDN como tecnología nueva incorporan una etapa en el desarrollo de las redes, permitiendo mejorar significativamente tanto la capacidad de gestión, como la escalabilidad y agilidad dentro de la red al emplear controladores.

- Los elementos a tener en cuenta para la selección de controladores SDN son: soporte OpenFlow, virtualización de red, funcionalidad de la red, escalabilidad, rendimiento, programación de red, confiabilidad, seguridad de la red, monitorización centralizada y visualización, fabricantes de controladores SDN, soporte de plataformas y procesamiento.

Conclusiones

- Se diseñó y programó los elementos de la Red mediante la simulación en el programa Mininet que permite simular redes SDN en un solo host o máquina virtual como si fuera una red física.
- Se implementó, controló y gestionó de forma centralizada los dispositivos que comprenden la topología planteada en el Proyecto ya que la arquitectura de las SDN tiene un servidor o grupo de servidores controladores donde se centraliza la toma de decisiones, al contrario que en una red tradicional, lo que conlleva a políticas inconsistentes.

Conclusiones

- En el proyecto se realizó la implementación de la SDN para la ESPE, para la cual se utilizó dos conmutadores Mikrotik RB951Ui-2HnD, que fueron adaptados para que funcionen con el protocolo OpenFlow. Estos equipos de fábrica no vienen diseñados para soportar este protocolo pero si es posible actualizarlos.

- El controlador Pyretic es muy similar POX, se podría decir que es su evolución ya que tiene estructuras y métodos más complejos. En el controlador Pyretic se puede programar parámetros para tomarlos en cuenta o ignorarlos, esto hara posible un mayor control para un administrador y se evitaran inconsistencias.

- El controlador PyResonance, que tiene una gran aceptación a nivel comercial debido a su gran facilidad de uso, puede definir reglas de manera mucho más sencilla que el resto de controladores por lo que es considerado de alto nivel, es similar a los controladores POX y Pyretic pero se lo considera la evolución de ellos.

Recomendaciones

- Se recomienda tener conocimientos previos sobre el lenguaje de programación Python, ya que esto nos permite crear los métodos necesarios y utilizar de manera óptima los APIs existentes para cada controlador y asi poder generar nuevos módulos.
- Se recomienda monitorear la aplicación, se lo puede hacer agregando una base de datos en la cual se almacenen las notificaciones de eventos ocurridos en cada equipo de la red, estos datos estén disponibles y puedan ser utilizados en un análisis posterior.

Recomendaciones

- Para evitar problemas de compatibilidad es necesario verificar que el software del servidor controlador sea compatible con los conmutadores elegidos para realizar la implementación. Esto es vital ya que caso contrario la SDN no podrá ser implementada.

- Se recomienda empezar usando la herramienta Mininet para estudiar las SDN. En esta herramienta se puede simular redes SDN y es posible familiarizarse con conceptos y procedimientos propios de OpenFlow y de las SDN. Es una herramienta muy práctica, ya que incluye todo lo necesario para trabajar con SDN en un nivel básico. No posee una interfaz gráfica y se sugiere previamente tener conocimientos básicos del sistema operativo Linux y de Python para la creación de topologías o componentes personalizados.