

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación



**DISEÑO DE UN BIGSWITCH EN
DISPOSITIVOS BANANA PI PARA
LABORATORIOS DOCENTES**

TRABAJO FIN DE MÁSTER

Cristóbal Joshue Domínguez Véliz

2018

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

**Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos**

TRABAJO FIN DE MÁSTER

**DISEÑO DE UN BIGSWITCH EN
DISPOSITIVOS BANANA PI PARA
LABORATORIOS DOCENTES**

Autor

Cristóbal Joshue Domínguez Véliz

Director

Ángel Fernández del Campo

Departamento de Ingeniería de Sistemas Telemáticos

2018

Resumen

Dentro de la industria de las redes se ha tenido una idea de sistemas de hardware y software integrados verticalmente y basados solamente en arquitecturas propietarias, las que vienen siendo prácticamente iguales desde hace décadas o al menos hasta el momento en que apareció el concepto y funcionamiento de las redes definidas por Software. Con su llegada, se introdujo la función de poder tener por un lado la configuración de un dispositivo de red y por otro, el medio por el que se transmite y se puede limitar el tráfico de datos. Quedó atrás la fragilidad operativa provocada por las arquitecturas de red heredadas, también la generación de tiempos de aprovisionamientos muy largos, produciéndose costos operativos sumamente altos.

Con una red definida por software (SDN) se tiene un enfoque moderno para redes que hace uso de abstracción de software basada en procesos entre el plano del control y el plano de datos para de esa manera eliminar la naturaleza compleja y estática de aquellas arquitecturas heredadas de redes distribuidas.

El borde o la frontera de la red es el medio por el cual el operador se conecta con sus clientes. Actualmente, están surgiendo métodos que permiten una mejor experiencia hacia el usuario final ofrecida por el operador y la habilitación de plataformas con servicios ágiles. Producido a través de implantaciones modernas de centrales digitales de conmutación de frontera, en donde existe un acoplamiento de tecnologías como SDN, NFV y Cloud que permiten formar centros de datos ágiles para la frontera de la red. Se tiene al proyecto CORD (<https://www.opennetworking.org/projects/cord/>) como ejemplo de la creación de un datacenter frontera operacional con capacidades de servicio incorporadas, el cual ha sido elaborado en base a principios de diseño cloud-native.

Lo que se desea es diseñar un entorno de laboratorio docente donde se puedan demostrar las funcionalidades que posee un BigSwitch dentro de una red, por medio de un conjunto de conmutadores físicos, basado en redes definidas por software, teniendo un propósito didáctico dentro de las instalaciones de los laboratorios de la Universidad.

Para tal efecto se tendrá en cuenta el uso de dispositivos BananaPi BPI-R2 que estarán trabajando como Switches y también los dispositivos BPI-M3 que actuarán como plataformas donde realizar las funciones de controladores SDN y donde se ejecutarán entornos mininet de emulación de parte de las redes SDN a diseñar.

Abstract

Within the network industry, there has been an idea of vertically integrated hardware and software systems based only on proprietary architectures, which have been practically the same for decades or at least until the moment when the concept and functioning of Software-Defined Networking. With its arrival, the function of being able to have on the one side the configuration of a network device and on the other, the means by which it is transmitted and data traffic can be limited, was introduced. It was behind the operational fragility caused by network architectures inheritance, also the gestation of very long procurement times, resulting in extremely high operating costs.

With a Software-defined Networking (SDN) there is a modern approach for networks that makes use of standards-based software abstraction between the plane of the network control and the plane of data forwarding in order to eliminate the complex and static nature of those architectures inherited from distributed networks.

The edge or the border of the network is the means by which the operator connects with its clients. Currently, methods are emerging that allow a better experience towards the end user offered by the operator and enabling platforms with agile services. This has been through modern deployments of digital front-end switching plants, where there is a coupling of technologies such as SDN, NFV and Cloud that allow to form agile data centers for the network edge. The CORD project (<https://www.opennetworking.org/projects/cord/>) is an example of the creation of a complete operational frontier datacenter with built-in service capabilities, which has been developed on basic hardware.

What is desired is to design a teaching laboratory environment to demonstrate the features that a BigSwitch has by means of a set of virtual switches, based on software defined networks, having a didactic purpose within the facilities of the University laboratories.

For this purpose, the use of BananaPi BPI-R2 devices that will be working as Switches will be taken into account, as well as the BPI-M3 devices that will act as platforms to perform the functions of SDN controllers and where emulation mininet environments will be executed on behalf of the SDN networks to design.

Índice general

Resumen	i
Abstract.....	iii
Índice general.....	v
Índice de figuras.....	ix
Siglas	xi
1 Generalidades	15
1.1 Justificación	15
1.2 Objetivos	16
1.2.1 Objetivos Generales.....	16
1.2.2 Objetivos específicos de las prácticas a desarrollar	16
1.3 Alcances y limitaciones.....	16
1.3.1 Alcances	17
1.3.2 Limitaciones	17
2 Estado del arte	18
2.1 Redes Definidas por Software	18
2.1.1 Plano de datos en SDN	21
2.1.2 Plano de control en SDN	22
2.1.2.1 Controlador OpenDaylight	24
2.1.3 Operación en SDN.....	25
2.2 DSA-Linux.....	25
2.2.1 Protocolos de etiquetado Switch	27
2.2.2 Dispositivos de red Master	27
2.2.3 Networking Stack hooks	27
2.2.4 Dispositivos de red Slave	28
2.2.5 Estructuras de datos en DSA	29

2.2.6	Limitaciones del diseño DSA.....	30
2.2.7	Errores comunes utilizando configuraciones DSA.....	30
2.2.8	Switchdev	31
2.3	Arquitectura de Conmutación BigSwitch Leaf-Spine	31
2.3.1	Topología Leaf-Spine.....	34
2.4	Tecnologías de Hardware	35
2.4.1	Banana Pi	35
2.4.1.1	Banana Pi BPI-R2.....	36
2.4.1.2	Banana Pi BPI-M3.....	37
2.5	Tecnologías de Software.....	39
2.5.1	Ubuntu 16.04 Kernel 4.14	39
2.5.2	OpenvSwitch.....	39
3	Diseño del BigSwitch para laboratorio docente.....	41
3.1	Preparación y configuración de los conmutadores	42
3.1.1	Preparación del Host y descarga del repositorio	42
3.1.2	Inclusión de módulos OpenvSwitch, compilación del bootloader y kernel 4.14	43
3.1.3	Creación y configuración de la imagen.....	45
3.1.4	Instalación del Sistema Operativo	50
3.1.5	Instalación y configuración de OpenvSwitch.....	51
3.2	Preparación del controlador	53
3.2.1	Instalación del Software OpenDaylight	53
3.2.2	Instalación de módulos OpenDaylight	53
3.3	Dificultades con el diseño	54
3.3.1	¿Qué kernel utilizar?	54
3.3.2	Instalación de OpenvSwitch	54
3.4	Portar OpenvSwitch a un nuevo hardware o software	55
4	Prácticas de Laboratorio	58
4.1	Materiales a utilizar.....	58
4.1.1	Banana Pi BPI-R2.....	58
4.1.2	Banana Pi BPI-M3.....	59

4.1.3	Cable de red	59
4.1.4	Tarjeta Micro SD	60
4.1.5	Dispositivos adicionales	61
4.1.6	Puesto de trabajo	61
4.2	Práctica 1: BigSwitch Leaf-Spine y pruebas de algoritmos.....	62
4.3	Práctica 2: Comparación de algoritmos de asignación de recursos en un BigSwitch basado en SDN.....	69
4.4	Práctica 3: Topología dependiente del tráfico	75
5	Conclusiones y Trabajos Futuros	79
5.1	Conclusiones	79
5.2	Propuestas de trabajos futuros	79
6	Bibliografía	81

Índice de figuras

Figura 1 Planos de control y datos [1]	20
Figura 2 Esquema de un controlador SDN [4]	23
Figura 3 Esquema del controlador OpenDaylight - Hydrogen [8].....	24
Figura 4 Representación gráfica de DSA visto desde un dispositivo de red [11]	29
Figura 5 Topología en árbol de tres niveles [13].....	32
Figura 6 Topología Leaf-Spine	34
Figura 7 Logo de Banana Pi	36
Figura 8 Banana Pi BPI-R2 [14].....	37
Figura 9 Banana Pi BPI-M3 [15]	38
Figura 10 Esquema del BigSwitch.....	41
Figura 11 Menú configuración del kernel.....	43
Figura 12 Módulos OpenvSwitch incluidos	44
Figura 13 Vista de la configuración Ethernet con OpenvSwitch a nivel de hardware y software	52
Figura 14 Arquitectura OpenvSwitch [20].....	55
Figura 15 Adaptación de <i>dpif provider</i> a la arquitectura OpenvSwitch [20]	57
Figura 16 Dispositivo Banana Pi BPI-R2.....	58
Figura 17 Fuente de Voltaje	59
Figura 18 Cable de red.....	60
Figura 19 Tarjeta Micro SD 16GB.....	60
Figura 20 Esquema del escenario1.....	62
Figura 21 Esquema del escenario 2.....	69
Figura 22 Esquema del escenario 3.....	75

Siglas

API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
BPI	Banana Pi
CLI	Command Line Interface
CORD	Central Office Re-architected as a Datacenter
CPU	Central Processing Unit
DSA	Distributed Switch Architecture
FDB	Forwarding Database
FTP	File Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
IT	Information Technology
JVM	Java Virtual Machine
LACP	Link Aggregate Control Protocol
LAN	Local Area Network
MDIO	Management Data Input/Output
MMIO	Memory-mapped Input/Output

ODL	OpenDaylight
OVS	Open vSwitch
PHY	Physical Layer
REST	Representational State Transfer
SBC	Single Board Computer
SDN	Software Defined Network
SPB	Shortest Path Bridging
SSH	Secure Shell
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TRILL	Transparent Interconnection of Lots of Links
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network

Introducción

En general, las tecnologías de la información han venido irrumpiendo en la sociedad una vez iniciada la segunda mitad del siglo XX. Aquello que nos ha influenciado a encaminarnos por esta tendencia es el hecho de cada vez estar más conectados y a su vez querer tener mayor acceso a la información o al conocimiento. Los métodos de innovación y el crecimiento tecnológico que han surgido son lo que han permitido que se abarquen la mayoría de las ciencias.

Para que se dé lugar a un entorno donde predomina la transferencia de información o de datos, es necesario que se establezcan una serie de reglas las cuales asegurarán una comunicación exitosa. Para tal efecto, se han establecido los llamados protocolos de enrutamiento y conmutación en un principio, pero a medida que se expande la red de Internet también se amplía el campo de las tecnologías utilizadas.

La integración del uso de redes definidas por software (SDN) en las arquitecturas de redes cada vez se vuelve más común, ya sea en ámbitos corporativos como educativos. De la misma forma se muestra como uno de los recursos a utilizar al momento de querer simplificar la gestión o administración de una red que se considere de gran tamaño.

Desde este punto de vista y haciendo uso de los dispositivos convencionales, refiriéndose a una red en la que se utilicen enrutadores y conmutadores, se realiza una separación de lo que se denomina el plano de datos y el plano de control. El plano de control es aquella función que identifica a un enrutador, es el que se encarga de decidir cuál es el camino que debe tomar un paquete para poder llegar a su destino. El campo de acción o la función principal que cumple SDN es reemplazar aquel trabajo que realiza el enrutador y realizarlo a través de un sistema centralizado que tenga conocimiento global de la red.

Este trabajo se encuentra dividido en cinco capítulos, de los que a continuación se brinda un breve resumen. En el capítulo uno se indica la motivación y justificación del por qué se ha decidido la elaboración de este proyecto, tomando en cuenta hasta qué punto se decidió llegar y los límites que da el entorno. El capítulo 2 consta de los conceptos necesarios que permiten tener un conocimiento base y de esta manera entender todo aquello que se ha elaborado.

El capítulo tres menciona detalladamente aquellos procedimientos que se han elaborado para obtener el diseño de un BigSwitch para un laboratorio docente. En el capítulo cuatro se encuentran las prácticas de laboratorio planteadas, acorde a las limitaciones de los dispositivos. Para finalizar, en el capítulo cinco se han descrito las conclusiones que se han obtenido una vez que se ha desarrollado el trabajo y se mencionan aquellas líneas futuras que se puede ramificar este proyecto.

Capítulo 1

1 Generalidades

En este capítulo se describe el motivo que impulsó para realizar este trabajo, destacando de igual manera los objetivos a corto plazo. Así también se detallan los alcances y limitaciones que se tuvieron durante el período de elaboración.

1.1 Justificación

La arquitectura de un BigSwitch está empezando a ser utilizada para realizar implantaciones modernas de centrales digitales de conmutación de frontera, que desempeñan su labor conjuntamente con la tecnología SDN (Redes definidas por Software). Actualmente, se está involucrando SDN dentro de proyectos de establecimiento de modelos de arquitecturas IoT junto con la tecnología Blockchain, y paralelamente se promueven tecnologías de software utilizadas principalmente en la nube.

El conocimiento de las últimas novedades tecnológicas, tanto de software como de hardware, es una valiosa herramienta para los estudiantes porque les permite estar informados sobre lo que se está llevando a cabo con proyectos actuales y para líneas de trabajo futuras poder tener variedad de opciones al momento de elegir entre una tecnología u otra.

A través del uso de dispositivos BananaPi, se desea dar a conocer características y diversas aplicaciones de un BigSwitch. Aunque poseen características similares a las de un Raspberry Pi, y además sea compatible con la mayoría de sus add-On y periféricos, resulta ser más capaz y rápido gracias a la memoria instalada y al procesador con doble núcleo que posee.

1.2 Objetivos

1.2.1 Objetivos Generales

- Analizar la arquitectura de un BigSwitch para conocer bajo qué circunstancias se lo puede implantar.
- Diseñar un ejemplo de conmutador BigSwitch, basado en SDN, a través de dispositivos BananaPi-M3 y BananaPi-R2.
- Comprender el funcionamiento y cómo se encuentra implantado el kernel utilizado para ejecutar los servicios en los dispositivos BananaPi.
- Analizar cómo es el funcionamiento y bajo qué condiciones operan los drivers de Arquitectura distribuida de conmutación.
- Elaborar prácticas de laboratorio que sirvan para comprensión del material teórico impartido en clases dadas en la escuela, ya sea que se trate de SDN, OpenFlow y/o arquitecturas de conmutadores de grandes prestaciones.

1.2.2 Objetivos específicos de las prácticas a desarrollar

- Comprender el uso de este tipo de Hardware para elaboración de futuros proyectos.
- Elaborar un diseño de la arquitectura de un BigSwitch que permita comprender su funcionamiento y ventajas que posee.
- Conocer el procedimiento de operación que se desarrolla dentro de la arquitectura de conmutación Leaf-Spine.

1.3 Alcances y limitaciones

En el mundo real, gracias a los sistemas de información y en especial a los sistemas de redes podemos encontrar un sin número de aplicaciones y funciones que dependerán del entorno en el que se estén desarrollando. En este caso se trata de un

diseño de un BigSwitch, dado que se trata de un proyecto con fines académicos a continuación se presentan alcances y limitaciones del mismo:

1.3.1 Alcances

- El BigSwitch podrá ser diseñado a través de los dispositivos BananaPi-M3 y BananaPi-R2

1.3.2 Limitaciones

- El diseño del BigSwitch será realizado en un ambiente de laboratorio docente.

Capítulo 2

2 Estado del arte

En el contexto de este capítulo se menciona la descripción de aquellos conceptos, técnicas y tecnologías utilizadas durante el desarrollo de este trabajo.

Dentro de la primera sección se abarca el concepto de las Redes Definidas por Software, que está siendo de gran aporte para aquellas arquitecturas implementadas en los Centros de Datos. Se aprovecha su flexibilidad al momento de su operación y también representa la economía de software con respecto al hardware con el que puedan contar. A continuación, se tiene un apartado que cita las características con las que cuenta una arquitectura distribuida de conmutadores, que permite conocer de qué manera opera el driver DSA del que está haciendo uso el dispositivo Banana Pi BPI-R2 con el que se efectúa este trabajo. Por último, se tiene un resumen de la arquitectura de conmutación de un Big Switch, en donde se introduce lo que se conoce como la topología Leaf-Spine.

2.1 Redes Definidas por Software

Considerando la mayor capacidad de los esquemas de transmisión y un rendimiento óptimo de los dispositivos de red, las arquitecturas de red con las que se ha venido contando son cada vez menos adecuadas, dada la complejidad y carga de red. Aquello con lo que se contaba habitualmente en las redes de comunicaciones, que era precisamente la construcción de redes basados en dispositivos totalmente autónomos, como lo son los conmutadores, enrutadores, entre otros. En estos dispositivos pueden establecer comunicaciones gobernadas por una serie de protocolos, una vez que han recibido su configuración inicial. A lo que se ha llegado es a que el funcionamiento de aquellos dispositivos se pueda separar en dos planos diferentes, los cuales son el plano de control y el de datos. [1]

El plano de datos, o también llamado el plano de forwarding, es aquel que se encarga de redirigir el tráfico entrante hacia el puerto de salida al que corresponda aquella comunicación. El plano de control es el que tiene la capacidad de decidir a través de qué puerto enviar el tráfico entrante.

El uso de las SDN ayuda a cumplir con los principales requerimientos que se necesitan para establecer un enfoque de red moderno, aquellos requerimientos son los siguientes [2]:

- Adaptabilidad: Para que las redes se ajusten y respondan dinámicamente, en función de aquellas necesidades que posea la aplicación, la política comercial y las condiciones de la red.
- Automatización: Los cambios de políticas que se efectúen deben ser propagados automáticamente para que no se produzcan errores dados por un trabajo realizado manualmente.
- Mantenibilidad: Se refiere a la introducción de nuevas características y capacidades, la cual debe ser transparente y además que exista una interrupción mínima de las operaciones.
- Gestión de modelos: El software de gestión de modelos que se utilice debe permitir la gestión de la red a nivel de modelo, enés de implementar cambios conceptuales mediante la reconfiguración de los elementos de red de manera individual.
- Movilidad: La funcionalidad de control debe adaptarse a la movilidad, aquí se incluyen a los dispositivos móviles de los usuarios y también los servidores virtuales.
- Seguridad integrada: Las aplicaciones de red deben integrar seguridad como un servicio central en lugar de una solución complementaria.
- Escalabilidad bajo pedido: Las implementaciones deben tener la capacidad de escalables hacia arriba o hacia debajo de la red y sus servicios para dar soporte a las solicitudes bajo demanda.

El concepto que se abarca sobre las SDN es el de permitir que los desarrolladores y los administradores de redes puedan obtener el mismo control sobre los equipos de red, así como lo han tenido con los servidores. Al realizar la división de la función de conmutación entre el plano de datos y un plano de control, este control se efectúa en dispositivos completamente diferentes como se puede ver en la *Figura 1* [1].

La función del plano de datos es simplemente la de reenviar paquetes, a diferencia del plano de control que es aquel quien proporciona la “inteligencia” para diseñar rutas, establecer prioridades y parámetros de política de enrutamiento que les permita cumplir con los requisitos de calidad de servicio y calidad de experiencia, y de esta manera poder hacer frente a los patrones de tráfico cambiantes.

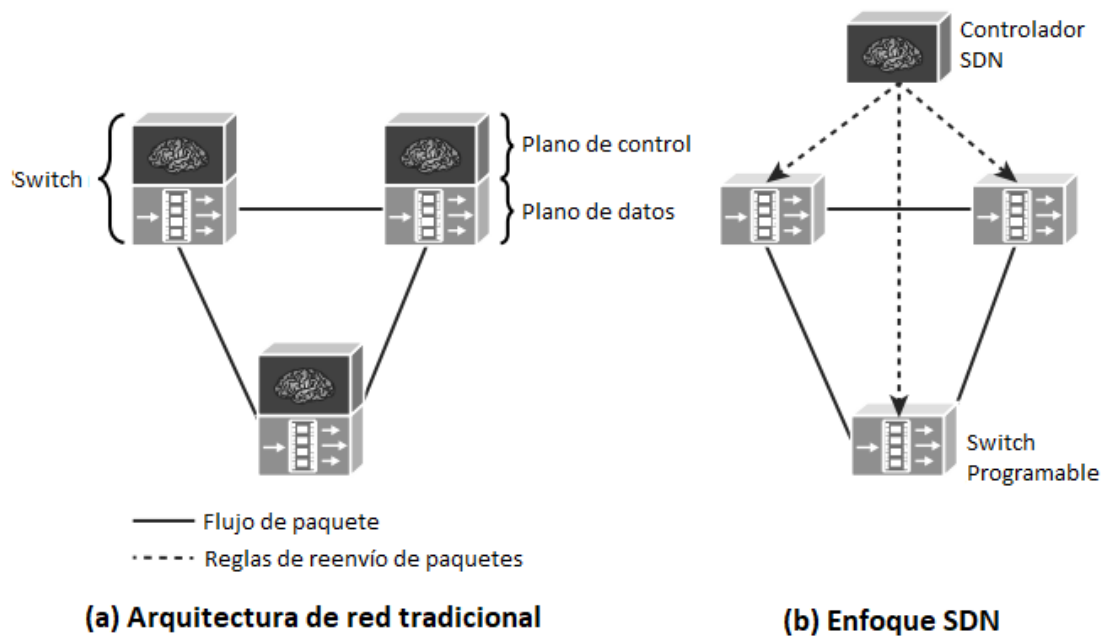


Figura 1 Planos de control y datos [1]

El plano de datos consiste conmutadores físicos y conmutadores virtuales. En cualquiera de los dos casos se tiene la misma funcionalidad de reenviar paquetes. La implementación interna de búferes, parámetros de prioridad y otras estructuras de datos que se relacionan con el reenvío pueden depender del proveedor. Sin embargo, cada conmutador debe implementar un modelo de reenvío de paquetes que sea uniforme y esté abierto a los controladores SDN.

Entre el plano de control y el plano de datos se tiene una interfaz “southbound” soportada por diferentes protocolos entre los que destaca OpenFlow, el cual se define como un protocolo. En los últimos años se lo ha utilizado para diferentes aspectos, como por ejemplo para transmisión de video y audio de alta definición, así como también ha servido para suites de productividad basadas en la nube. Esto implica que las redes SDN se hayan empleado en soluciones de gran ancho de banda, además de dinámicas, que requieren grandes cantidades de recursos que las redes tradicionales, y estáticas no pueden ofrecer de manera confiable.

Los beneficios que ofrece SDN en centros de datos son los siguientes [1] [3]:

- Manejo de grandes volúmenes de datos: Dentro de las organizaciones se acostumbra a procesar grandes conjuntos de datos mediante el procesamiento paralelo, al realizar esto se requiere que exista un amplio ancho de banda. La forma que aporta SDN es administrando de manera más efectiva el rendimiento y la conectividad.

- Soporte de tráfico basado en la nube: Se ha vuelto una tendencia el aumento de la nube dentro del sector de las IT y telecomunicaciones. La nube se basa en la idea de la capacidad bajo demanda y el autoservicio, lo cual SDN puede ofrecer de forma dinámica en función de la demanda y la disponibilidad de recursos que existan en el centro de datos.
- Administración del tráfico a muchas direcciones IP y máquinas virtuales: Con el uso de SDN se permite agregar tablas de enrutamiento dinámico para que la priorización de enrutamiento se pueda obtener en base a la retroalimentación de la red en tiempo real, teniendo como resultado que el control de las máquinas virtuales sea más fácil de administrar.
- Transformar a una infraestructura escalable y ágil: Cuando se utiliza SDN, se hace más sencillo agregar nuevos dispositivos, esto haría que se reduzca el riesgo de que exista alguna interrupción en el servicio. Es así como se puede obtener que con SDN se presente un mejor ajuste al procesamiento paralelo y al diseño en general de las redes virtualizadas.
- Administración de políticas y seguridad: Otro de sus beneficios es que se la puede usar para propagar de manera más eficiente y efectiva las políticas de seguridad a través de toda la red, incluyendo aquellos dispositivos utilizados como Firewall y otros demás elementos esenciales.

2.1.1 Plano de datos en SDN

Una vez que se ha logrado eliminar el plano de control de los dispositivos, se llega a obtener la simplificación en gran medida su funcionamiento, mejorando la capacidad de rendimiento. Dentro de sus funciones solo quedan cuatro diferentes opciones para tratar un paquete de tráfico incidente que se detallan a continuación [1]:

- FORWARD: Se enfoca en el reenvío básico de paquetes. El procedimiento que realiza primeramente es consultar una tabla para luego poder seleccionar el puerto de salida.
- DROP: En este caso se realiza un descarte del paquete. Esta acción puede ocurrir porque el dispositivo ha sido configurado específicamente para que lo haga de esta manera o por alguna situación de buffer overflow.
- CONSUME: Se realiza un reenvío de paquetes al controlador.
- REPLICATE: Esta opción permite realizar el reenvío por varios puertos. Por lo general se utiliza en escenarios multicast.

La tecnología más común para llevar a cabo todas estas funciones, por lo menos dentro de entornos OpenSource, se denomina OpenFlow.

2.1.2 Plano de control en SDN

Una vez que se habla de SDN, se debe conocer que la inteligencia de los dispositivos de red es trasladada hacia un único controlador. Dicho controlador no es más que una aplicación software encargada de poder administrar toda la red traduciendo políticas definidas en un alto nivel a unas simples instrucciones. Una vez que estas instrucciones son descargadas a los dispositivos, se les pueda hacer posible tomar las decisiones de forwarding adecuadas.

El controlador SDN debe tener una visión global de la red por completo y debe manejarla como si se tratara de un sistema distribuido, formado por diferentes dispositivos que poseen un estado propio, está encargado de solucionar los problemas de operación de red. La estructura básica de un controlador SDN se muestra en la *Figura 2*.

El núcleo del controlador SDN debe realizar las funciones de descubrimiento y administración de dispositivos y topología, además de un continuo seguimiento de estadísticas. Para realizar este trabajo, un controlador SDN debe incluir al menos los siguientes cuatro módulos [1]:

- End-user Device Discovery
- Network Device Discovery
- Network Device Topology Management
- Flow Management

Dentro de estos cuatro módulos se mantienen además bases de datos con información suficiente para describir, al menos, el estado presente de la red.

Para que se permita la comunicación del controlador con los demás dispositivos de la red y para hacer posible que se realice el control automatizado de ésta, el controlador SDN debe incluir dos interfaces: la Northbound API y la Southbound API.

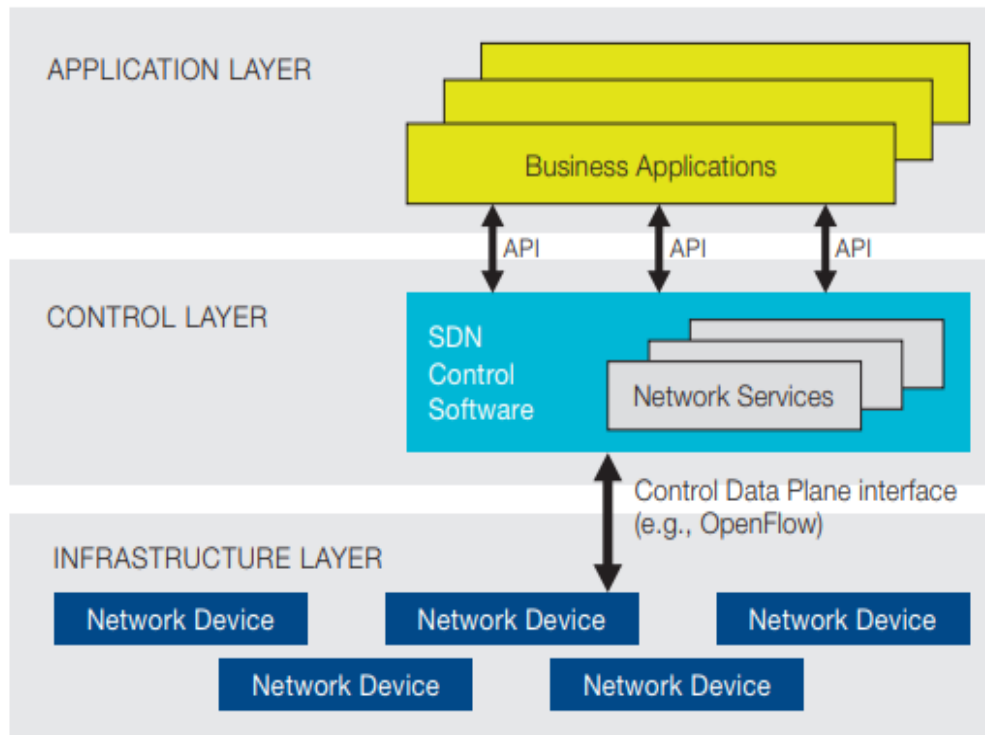


Figura 2 Esquema de un controlador SDN [4]

La Southbound API es la interfaz que permite a los dispositivos de la red comunicarse con el controlador SDN. Mientras que la implementación más común de esta interfaz es realizada con el protocolo OpenFlow, es preciso mencionar que no es el único enfoque para llevar a cabo esta tarea, existiendo otras opciones como por ejemplo la línea de comando de Cisco (CLI) o el protocolo NETCONF. [5]

La Northbound API proporciona métodos para que aplicaciones SDN se conecten con el controlador SDN con el fin de utilizar algoritmos que dentro de sus funciones automaticen el funcionamiento de la red. [6]

Hoy en día se pueden encontrar numerosas implementaciones de controladores SDN, entre alguna de ellas comerciales. Dentro del mundo del OpenSource también se han hecho varias implementaciones de controladores SDN, como NOX, en el cual se ha utilizado C como lenguaje de programación, u OpenDaylight, que utiliza Java.

2.1.2.1 Controlador OpenDaylight

El controlador OpenDaylight (ODL) se trata de un proyecto OpenSource que es desarrollado con acuerdos entre múltiples firmas de renombre en el mundo de la informática y las redes de comunicaciones, como por ejemplo Intel, Cisco, RedHat, Ericsson, etc), con el objetivo de mejorar el estado de las SDN ofreciendo un producto que es dirigido por la comunidad a la vez que es apoyado por el mundo empresarial.

El proyecto está desarrollado con el lenguaje Java, lo que le permite ser ejecutado en prácticamente cualquier sistema operativo compatible con la máquina virtual de Java (JVM). Esto hace posible que pueda ser desplegado en cualquier entorno de ejecución de red actual. [7]

En su Southbound API el controlador ODL da soporte que se efectúa por defecto a la tecnología OpenFlow, sin embargo, esta configuración puede ser modificada en cualquier momento para ofrecer soporte también para otros protocolos, tal es el caso de OVSDB. En la Northbound API ofrece una API REST para la comunicación con aplicaciones SDN. En la *Figura 3* se muestra la estructura que posee el controlador ODL de la versión Hydrogen. En la actualidad, se utiliza la versión Oxygen. [7]

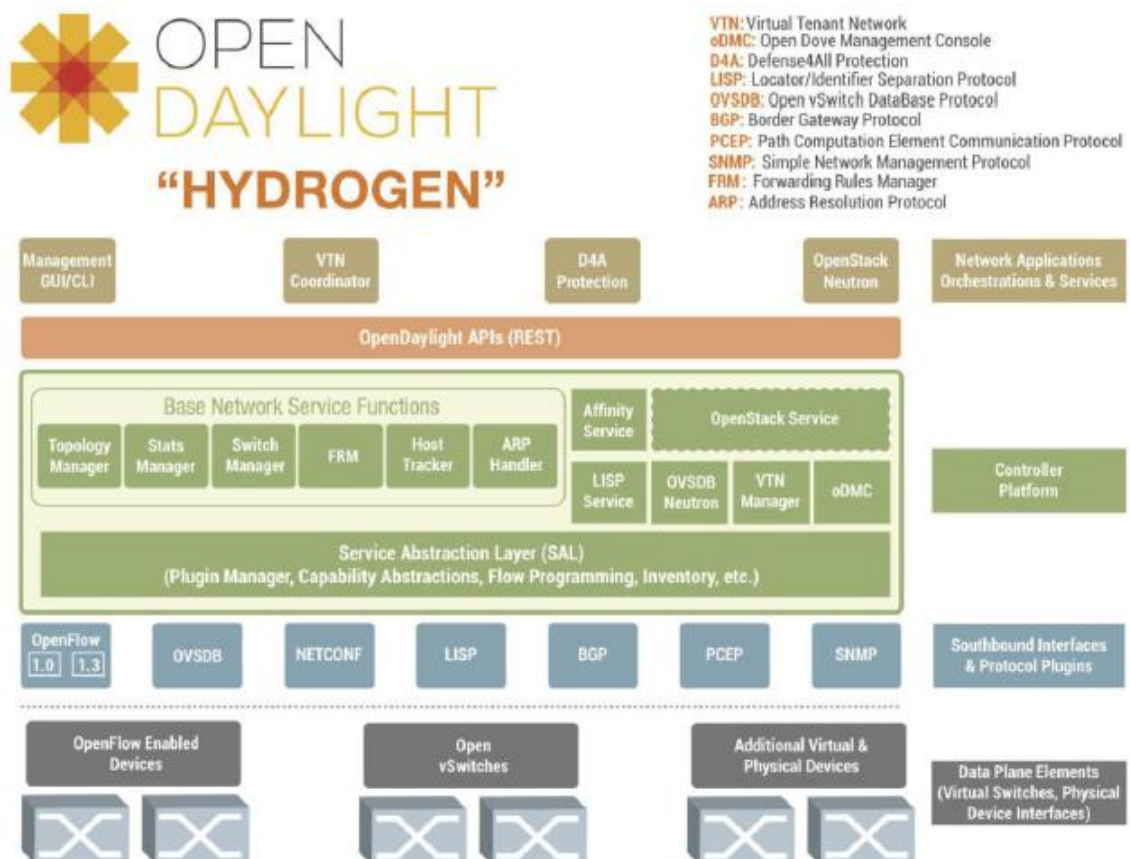


Figura 3 Esquema del controlador OpenDaylight - Hydrogen [8]

2.1.3 Operación en SDN

Cuando se trabaja con SDN, la información que se mueve por la red deja de ser tratada como unos paquetes aislados para pasar a verse como flujos de datos. Al describir un flujo de datos se puede decir que es un conjunto de paquetes que fluyen de manera unidireccional y sus correspondientes endpoints.

Una vez que un flujo es definido y pasado al controlador SDN, este se encarga de traducir y entregar las instrucciones que cada uno de los dispositivos de red han de seguir para manejar dicho flujo. Aquellas instrucciones son almacenadas en tablas (Flow Tables) para cada dispositivo de red.

Las aplicaciones conectadas al controlador son las encargadas de crear flujos o modificar las que ya existen. Los flujos pueden ser creados de manera proactiva, con intención de dar soporte a alguna tarea concreta, o de manera reactiva, que se da como respuesta a un paquete redirigido al controlador. [9]

2.2 DSA-Linux

DSA por sus siglas en inglés "*Distributed Switch Architecture*". DSA es un subsistema que en un principio fue diseñado principalmente para trabajar con conmutadores Marvell Ethernet (MV88E6xxx, que está dentro de la gama de productos LinkStreet) que utiliza Linux, pero desde ese momento ha evolucionado para ser compatible con otros proveedores también. [10] [11]

La idea con la que se empezó a elaborar este diseño era de poder utilizar herramientas Linux no modificadas como bridge, iproute2, ifconfig para que puedan trabajar de forma transparente, ya sea que se haya configurado o consultado un puerto físico de un dispositivo de red o que se trate de cualquier otro dispositivo de red más complejo como podría ser un conmutados. [11]

Un conmutador Ethernet generalmente consta de múltiples puertos en su panel frontal y con una o más CPUs y puertos de administración. El subsistema que actualmente maneja el DSA se basa en la presencia de un puerto de gestión conectado a un controlador Ethernet que es capaz de recibir tramas Ethernet desde el conmutador. Esta es una configuración muy común para todo tipo de conmutadores Ethernet que se encuentran en productos de hogares y oficinas: enrutadores o puertas de enlace.

Se le ha atribuido el término de distribuido porque el subsistema se ha diseñado con la capacidad de configurar y administrar switches en cascada utilizando enlaces Ethernet ascendentes y descendentes entre switches. Estos puertos en particular se denominan puertos “dsa”. Una colección de conmutadores múltiples conectados unos con otros se denomina “árbol de conmutación”. [11]

Para cada puerto del panel frontal, DSA creará dispositivos de red específicos que se utilizan como puntos finales de control y flujo de datos para su respectivo uso por la configuración de red en Linux. Estas interfaces de red especializadas se denominan interfaces de red esclavas.

El caso ideal para usar DSA es cuando un conmutador Ethernet admite un “switch tag”, la cual es una función de hardware que hace que el conmutador inserte una etiqueta específica para cada trama Ethernet que recibe hacia/desde puertos específicos para ayudar a la interfaz de gestión a resolver:

- ¿De qué puerto viene esta trama?
- ¿Cuál fue la razón por la cual esta trama fue enviada?
- ¿Cómo enviar tráfico originado por la CPU con destino a un puerto específico?

El sistema admite conmutadores que no son capaces de insertar etiquetas, pero en este caso las características estarían ligeramente limitadas (la existencia de la separación del tráfico se basa en las IDs de VLAN basadas en el puerto). [11]

Se debe tener en cuenta que DSA no crea actualmente interfaces de red para los puertos CPU y DSA porque:

- El puerto “cpu” es el lado del conmutador Ethernet del controlador de gestión, y como tal, entonces crearía una duplicación de la función, ya que estaría obteniendo dos interfaces para el mismo conducto: master netdev y “cpu” netdev.
- Los puertos “dsa” son solo conductos entre dos o más conmutadores, y como tales tampoco pueden utilizarse realmente como interfaces de red adecuadas.

2.2.1 Protocolos de etiquetado Switch

Actualmente, DSA admite cinco diferentes protocolos de etiquetado, así como también soporta el modo sin etiquetar. Los diferentes protocolos están implementados en [11]:

- `net/dsa/tag_trailer.c`: Marvell's trailer tag mode (heredado)
- `net/dsa/tag_tdsa.c`: Etiqueta DSA original de Marvell
- `net/dsa/tag_edsa.c`: Etiqueta DSA mejorada de Marvell
- `net/dsa/tag_brcm.c`: Etiqueta de Broadcom 4 bytes
- `net/dsa/tag_qca.c`: Etiqueta de Qualcomm 2 bytes

El formato exacto del protocolo de etiquetado es específico del vendedor, pero en líneas generales, todos ellos contienen algo que [11]:

- Identifica de qué puerto proviene la trama Ethernet/ debe enviarse a
- Provee un motivo del por qué esta trama fue reenviada a la interfaz de administración.

2.2.2 Dispositivos de red Master

Dispositivos de red Master son controladores de dispositivos de red Linux regulares no modificados para la interfaz de administración Ethernet/CPU. Sus correspondientes drivers pueden necesitar ocasionalmente saber si DSA está habilitado, pero gracias a las pruebas que se han realizado se ha demostrado que el subsistema DSA funciona con drivers estándar de la industria, como lo pueden ser: `e1000e`, `mv643xx_eth`, etc. sin tener que introducir modificaciones a estos drivers. [11]

Dichos dispositivos de red también se pueden denominar a menudo dispositivos de red de conducto ya que actúan como un conducto entre el procesador host y el conmutador Ethernet hardware.

2.2.3 Networking Stack hooks

Cuando se usa un netdev maestro con DSA, se coloca una especie de un pequeño gancho en la pila de protocolos de la red para que el subsistema DSA procese el protocolo de etiquetado específico del conmutador Ethernet. DSA logra este objetivo

registrando un tipo Ethernet específico en la pila de red, también conocido como tipo `ptype` o `packet_type`. Una secuencia típica de la recepción de trama Ethernet podría ser algo similar a lo siguiente: [11]

Dispositivo de red Master (por ejemplo, del driver `e1000e`)

Recibir disparos de interrupción:

- Se invoca a la función de recepción
- Se realiza el procesamiento básico de paquetes: obtención de longitud, estado, entre otros.
- El paquete está preparado para ser procesado por la capa Ethernet llamando a `eth_type_trans`.

2.2.4 Dispositivos de red Slave

Los dispositivos de red slave creados por DSA se apilan sobre su dispositivo de red principal, cada una de estas interfaces de red será responsable de ser un punto final de control y de flujo de datos para cada puerto del conmutador en el panel frontal. Estas interfaces se encuentran especializadas para:

- Insertar/eliminar el protocolo de etiqueta de conmutador (en caso de que exista) al enviar tráfico a/desde puertos de switch específicos.
- Consultar al conmutador para las operaciones de `ethtool`: estadísticas, estados de enlace, Wake-on-LAN
- Gestión PHY externa/interna: enlace, autonegociación, etc.

Todos aquellos dispositivos de red slave tienen punteros de función `net_device_ops` y `ethtool_ops` personalizados que permiten a DSA introducir un nivel de estratificación entre la pila de red/`ethtool` y la implementación del controlador de conmutador. [11]

Una vez que se realiza la transmisión de las tramas desde estos dispositivos de red slaves, DSA buscará qué protocolo de etiquetado de conmutadores está actualmente registrado con estos dispositivos de red e invocará una rutina de transmisión específica que se encarga de agregar la etiqueta de conmutador relevante en las tramas Ethernet.

Estas tramas se colocan en la cola para su transmisión utilizando la función de red principal `ndo_start_xmit()` del dispositivo, ya que contienen la etiqueta del conmutador apropiada, es entonces que el conmutador Ethernet podrá procesar estas tramas

entrantes desde la interfaz de administración y acto seguido entregará estas tramas al puerto del conmutador físico.

En resumen, en la *Figura 4* se muestra básicamente cómo se ve DSA desde la perspectiva de un dispositivo de red.

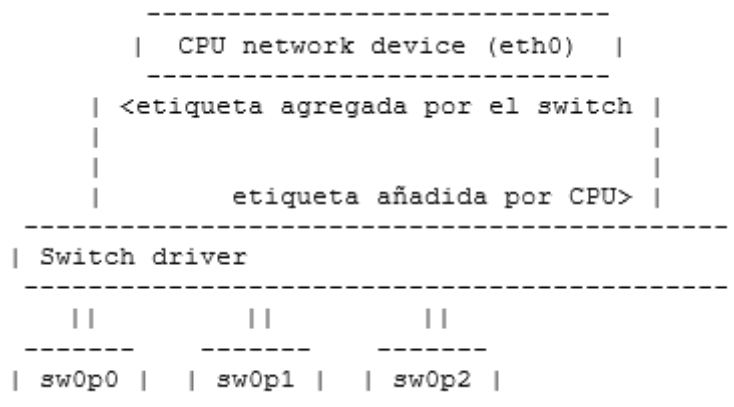


Figura 4 Representación gráfica de DSA visto desde un dispositivo de red [11]

2.2.5 Estructuras de datos en DSA

Las estructuras de datos DSA se definen dentro de `include/net/dsa.h`, así como también en `net/dsa/dsa_priv.h`, en donde se las tiene declaradas de la siguiente manera: [11]

- `dsa_chip_data`: Aquí es donde se tiene la configuración de datos de plataforma para un dispositivo de conmutación dado, esta estructura describe el dispositivo principal de un dispositivo conmutador, su dirección, así como también varias propiedades de sus puertos: nombres/etiquetas, y finalmente una indicación de tabla de enrutamiento (una vez que se conectan conmutadores)
- `dsa_platform_data`: En esta estructura de datos se tienen datos de configuración del dispositivo de plataforma que pueden hacer referencia a una colección de estructura `dsa_chip_data` si se conectan en cascada múltiples conmutadores, el dispositivo de red master al que está conectado este árbol de conmutación debe referenciarse
- `dsa_switch_tree`: Esta estructura se encuentra asignada al dispositivo de red master bajo `dsa_ptr`, esta estructura hace referencia a una estructura `dsa_platform_data`, así como al protocolo de etiquetado soportado por el árbol de conmutadores, y qué hooks de función de recepción / transmisión

deben invocarse, la información sobre el conmutador directamente conectado es también provisto por el puerto de CPU. Finalmente, se hace referencia a una colección de `dsa_switch` para abordar interruptores individuales en el árbol.

- `dsa_switch`: Esta estructura describe a un dispositivo de conmutación en el árbol, haciendo referencia a un `dsa_switch_tree` como un backpointer, dispositivos de red slave, dispositivo de red master y una referencia al respaldo `dsa_switch_ops`.
- `dsa_switch_ops`: Esta estructura hace referencia a los punteros de funciones, en donde a continuación, en el siguiente apartado se tiene una descripción completa.

2.2.6 Limitaciones del diseño DSA

DSA se implementa como un controlador de dispositivo de plataforma DSA, lo que favorece bastante porque registrará todo el árbol de conmutación DSA conectado a un dispositivo de red maestro en una sola vez, facilitando la creación del dispositivo y simplificando un poco el modelo de controlador de dispositivo, esto viene con una cantidad de limitaciones: [11]

- La compilación de DSA y sus controladores de conmutación como módulos no se encuentra disponible, se deben obtener los módulos directamente.
- La compatibilidad que posee el controlador del dispositivo no refleja necesariamente el bus/dispositivo original desde el que se puede crear el conmutador.
- No es posible admitir switches que no sean MDIO ni MMIO (plataforma).

También se tienen limitaciones en el número de dispositivos y puertos. Actualmente, DSA limita el número máximo de conmutadores dentro de un árbol a 5 (`DSA_MAX_SWITCHES`) y la cantidad de puertos por switch a 12 (`DSA_MAX_PORTS`). Estos límites podrían ampliarse para admitir configuraciones más grandes en caso de que surgiera esta necesidad.

2.2.7 Errores comunes utilizando configuraciones DSA

Una vez que un dispositivo de red maestro se encuentra configurado para usar DSA (`dev -> dsa_ptr` pasa a ser no nulo), y el conmutado ubicado detrás de él espera un

protocolo de etiquetado, esta interfaz de red solo puede usarse exclusivamente como una interfaz de conducto. Al hacer el envío de paquetes directamente a través de esta interfaz (por ejemplo, abrir un socket utilizando esta interfaz) no nos hará pasar por la función de transmisión de protocolo de etiquetado de conmutación, por lo que el interruptor de Ethernet en el otro extremo, esperando una etiqueta, es muy común que descarte el frame. [11]

Los dispositivos de red slave verifican que el dispositivo de red master esté up, antes de permitir que traiga administrativamente estos dispositivos de red slaves. Un error de configuración común es olvidar traer primero el dispositivo de red master.

2.2.8 Switchdev

DSA utiliza directamente Switchdev cuando interactúa con la capa de bridge, y más específicamente con su porción de filtrado de VLAN cuando configura VLANs sobre los dispositivos de red slaves por puerto. Dado que DSA trata principalmente con conmutadores conectados a MDIO, aunque no exclusivamente, las fases de preparación / aborto / confirmación de SWITCHDEV a menudo se simplifican en una fase de preparación que verifica si la operación es compatible con el controlador de conmutación DSA y una fase de confirmación que aplica los cambios.

Hoy en día, los únicos objetos switchdev admitidos por DSA son los objetos FDB y VLAN. [11]

2.3 Arquitectura de Conmutación BigSwitch Leaf-Spine

La búsqueda del diseño más óptimo para las redes de comunicaciones se ha basado, desde la aparición de estas, de un campo en constante evolución debido al constante aumento de su cobertura y a sus cambiantes requisitos.

En el desarrollo de Internet, la tarea fundamental de las grandes redes de comunicaciones se ha centrado en facilitar la conexión entre dispositivos situados en segmentos diferentes de la red que, por lo general, se encuentran alejados físicamente. Un requisito fundamental, por lo tanto, se ha tratado de disponer de la capacidad de encaminar tráfico eficientemente en una línea Norte-Sur, dejando en un segundo plano las comunicaciones dentro de un mismo segmento de la red (comunicación Este-Oeste).

El otro requisito importante se trata de la capacidad de expansión de la red. Las redes debían, y deben, ser escalables con una sencillez razonable.

Por lo general, las redes centradas en resolver este problema han sido diseñadas en varios niveles, lo que ha dado lugar a la topología de árbol jerárquico. En esta arquitectura, los dispositivos son colocados en diferentes capas (tres al menos) interconectadas de manera horizontal. [12]

- El nivel inferior está denominado como nivel de acceso y comprende de conmutadores que se encuentran directamente conectados con los dispositivos finales (hosts).
- El nivel intermedio tiene interconexiones con redundancia a los otros dos niveles y hace la función de capa de agregación. Normalmente está compuesto de switches de capa 3.
- El nivel superior se trata del núcleo de la red. Se compone de routers o switches de nivel 3 y se puede conectar con núcleos de otros segmentos de red.

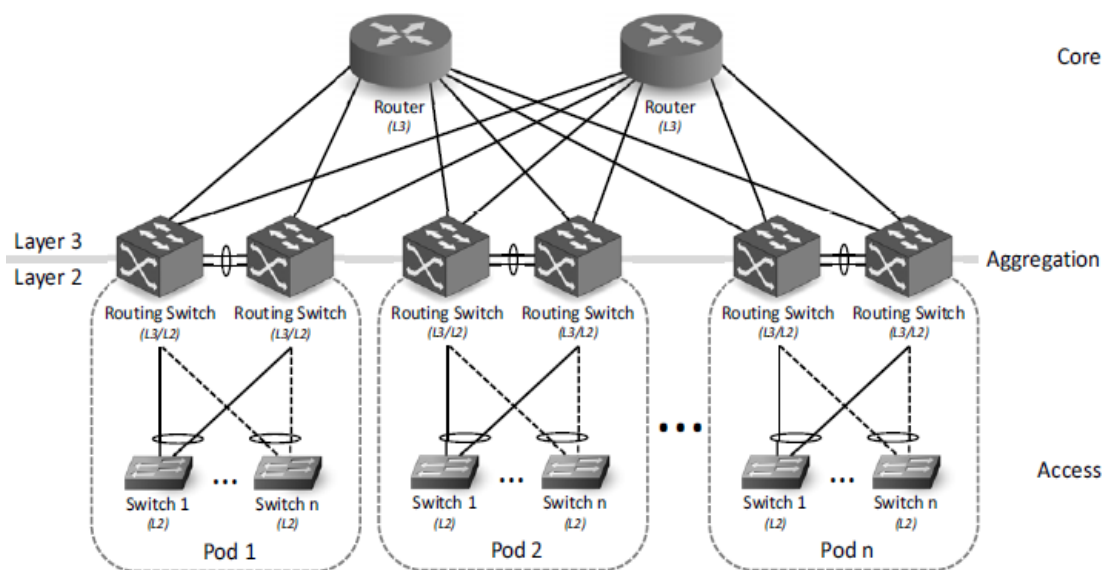


Figura 5 Topología en árbol de tres niveles [13]

Dentro de la capa del núcleo (capa 3) se tiene que los enrutadores son los que proveen de separación entre los pods. En la capa de agregación (capa 2/3) se tiene que los conmutadores pueden servir de frontera entre los pods.

Se tiene un gran inconveniente con este planteamiento, el cual se trata de la existencia o aparición de bucles en la red, haciendo que resulte imprescindible el uso de algún protocolo como STP que los evite. El problema con el uso de este protocolo

radica en que las conexiones redundantes no podrán ser utilizadas normalmente, de modo que no se utilizan eficientemente todos los recursos que se tienen disponibles.

Pero también es cierto que esta topología de red ha sido la solución preferida, para cumplir los aspectos que se han citado, durante un largo período de tiempo. Sin embargo, en la actualidad existen diferentes entornos, como los centros de datos o los centros de conmutación de las ISPs, en los que el primero de estos requisitos ha cambiado. La red no solo debe ser capaz de mover tráfico de manera eficiente en la línea Norte-Sur, sino que también es necesario poder hacerlo en caminos horizontales (Este-Oeste). [13]

Dentro de los principales factores que han hecho crecer la necesidad de cursar tráfico Este-Oeste se tiene a la popularidad que ha adquirido recientemente la computación en la nube. Bajo estas circunstancias, cientos de máquinas dentro de un mismo centro de datos necesitan mantener un intercambio constante de información, hasta el punto en el que se estima que más de 70% del volumen total de tráfico se trata de comunicaciones Este-Oeste. Otro factor importante es la creciente oferta de contenidos de las compañías ISP, lo que hace necesario que aparezcan los centros de conmutación con capacidades extraordinarias.

Efectuar una ejecución de mover tráfico Este-Oeste en una red diseñada con una topología que en un principio fue diseñada para el tráfico Norte-Sur puede causar un nivel de sobreescripción inaceptable en las conexiones entre los diferentes niveles de la misma. De este modo, el tráfico Este-Oeste se encuentra con un importante problema de cuellos de botella, que debe ser solucionado.

El hecho de agregar nuevos enlaces redundantes y sustituir el protocolo STP por algún otro más eficiente, con el fin de aumentar el ancho de banda entre los diferentes niveles, no soluciona el problema por completo sino solamente en cierta medida. Dadas todas estas circunstancias, la aparición de un nuevo enfoque para el diseño de este tipo de redes era algo previsible. [13]

El importante uso que ha tenido y tiene la arquitectura de tres niveles es lo que ha llevado a que se la haya empleado a lo largo de todos estos años. Para que sea reemplazada por otra arquitectura, debe ser por una realmente productiva como base a sus prestaciones y que mejore las ventajas actuales que son: [12]

- Rendimiento: Gracias a la agregación de enlaces entre los niveles, un núcleo de alto rendimiento permite que se curse casi en su totalidad la velocidad que permite el cable.
- Redundancia: A nivel del núcleo y de la capa de distribución se tiene la redundancia adecuada para poder asegurar la disponibilidad de la ruta.

- Seguridad: Tanto la seguridad que se configura en los puertos a nivel de la capa de acceso y las políticas que se incluyen en la capa de distribución permiten que la red tenga un nivel de seguridad mayor.
- Escalabilidad: La topología de árbol jerárquico puede expandirse con facilidad.
- Facilidad de mantenimiento.
- Facilidad de administración.

2.3.1 Topología Leaf-Spine

La topología Leaf-Spine surge tras modificar la anterior topología en árbol, añadiendo más dispositivos redundantes a el nivel intermedio. El problema que se consigue resolver de este modo es el de encaminamiento de altos volúmenes de tráfico horizontal (Este-Oeste). En la *Figura 6* se tiene un ejemplo de cómo sería la topología Leaf-Spine en el que se cuente con tres switches Leafs y dos switches Spine.

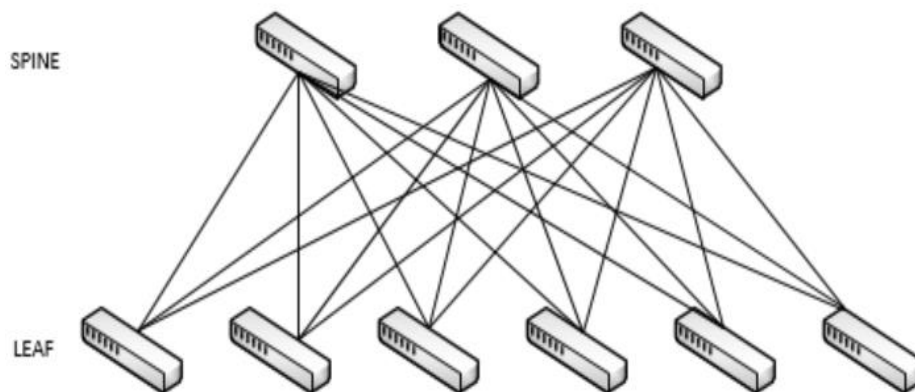


Figura 6 Topología Leaf-Spine

Bajo ese nuevo modelo se rediseñan los niveles de acceso y agregación, que ahora se llamarían Leaf y Spine respectivamente, de manera que cualquier par de switches de acceso se encuentren separados por un único salto. Además, se deja de utilizar STP, siendo este sustituido por protocolos más eficientes como Transparent Interconnection of Lots of Links (TRILL) o Shortest Path Bridging (SPB). Estos protocolos se encargan

de prevenir bucles de tráfico en la red, pero no realizan ningún bloqueo a los enlaces, lo que lo diferencia de STP. [13]

Gracias a estas dos características es que se ha llegado a provocar que la topología Leaf-Spine sea el estándar actual de mercado para aquel entorno que necesite cursar tráfico Este-Oeste. Sin embargo, también existen desventajas a la hora de construir una red de estas características, especialmente cuando se desea incrementar su tamaño.

Si se desea escalar una red que utilice este tipo de topología, se debe incrementar la cantidad de dispositivos en la capa Spine, esto se obtiene al añadir nuevos switches junto con su correspondiente cableado. Si bien esto es sencillo en la teoría, en la práctica puede ser notablemente complicado, además de económicamente inasequible. Asimismo, se puede llegar a encontrar un momento en el que no se disponen de puertos adicionales. En este caso el escalado de la red solamente es posible aumentando la proporción de sobreescripción entre los niveles Leaf y Spine, que no debe ser excesiva, ya que se tendría el mismo problema que se estaba intentando resolver.

Un BigSwitch es el resultado de la interconexión de un conjunto de conmutadores para de esta manera obtener un mayor número de puertos tanto de entrada como de salida. Por lo general, se encuentra orientado a arquitecturas donde se necesita un alto rendimiento, como en Centros de Datos o Centrales de conmutación. Para poder formar un BigSwitch, lo que se necesita es basarse en la arquitectura Leaf Spine.

2.4 Tecnologías de Hardware

En esta sección se presentan los diferentes componentes físicos que se emplean para la elaboración del diseño de un BigSwitch.

2.4.1 Banana Pi

Es toda una línea de computadoras, o también llamadas microcomputadoras, que cuentan con una placa del tamaño de una tarjeta de crédito. La idea fundamental para la creación de esta gama de dispositivos es la de aprovechar el bajo costo y de esta manera fomentar el desarrollo interno de software y hardware, además del aprendizaje de software escolar. Posee un diseño similar al de su predecesor Raspberry Pi, lanzado al mercado en el 2013. La compañía que produce estos dispositivos es Shenzhen SINOVOIP Co., Ltd.



Figura 7 Logo de Banana Pi

Pese a que son dispositivos bastante pequeños, en comparación con los ordenadores, se pueden desenvolver sin ningún inconveniente con software Linux. A nivel de hardware, se tienen las siguientes características principales:

- Procesador ARM7 Dual Core 1 Ghz
- GPU ARM MALI 4000
- 1GB de RAM DDR3
- Conector SATA para disco duro
- Conexión infrarroja

A continuación, se mencionan características de los dispositivos utilizados para llevar a cabo este trabajo. Entre los cuales se cuenta con los siguientes:

- Banana Pi BPI-R2
- Banana Pi BPI-M3

2.4.1.1 Banana Pi BPI-R2

Este modelo es una placa de desarrollo basada en enrutadores. En ella se pueden ejecutar varios sistemas operativos de código abierto, entre los que se encuentran: OpenWrt, Android y Bananian. Puede utilizarse para obtener un alto rendimiento inalámbrico, entretenimiento en el hogar, domótica, entre otros usos.

Además, el BPI-R2 incluye un ARM Cortex-A7 MPCore de cuatro núcleos, que puede estar llegando a operar sin inconvenientes hasta los 1.3GHz. Entre sus principales características de hardware se tienen las siguientes: [14]

- MediaTek MT7623N, Quad-core ARM Cortex-A7

- GPU Mali 450 MP4
- 2G DDR3 SDRAM
- Mini interfaz PCIE
- Soporte de hasta 2 interfaces SATA
- Ranura Micro SD admite una expansión de hasta los 256GB
- Flash 8G eMMC (opcional 16/32/64G)
- Soporte de interfaz de pantalla MIPI
- 2 puertos USB 3.0
- 1 puerto OTG USB 2.0
- Reproducción de vídeo de alta definición 1080p
- Puerto HDMI y salida de audio multicanal
- WIFI y Bluetooth 4.1 con 802.11B/G/N integrado

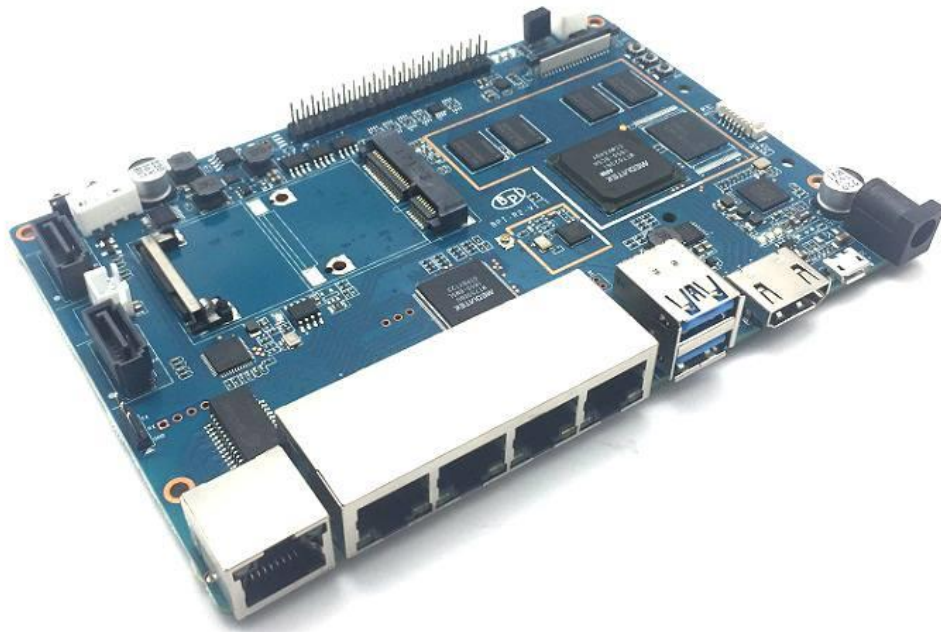


Figura 8 Banana Pi BPI-R2 [14]

2.4.1.2 Banana Pi BPI-M3

Este modelo de Banana Pi es una computadora de placa única (SBC) a la que se le ha cargado un procesador Octa-core y 2 GB de memoria RAM. Además, cuenta con una conexión Gigabit Ethernet, 2 USB, SATA, Bluetooth y HDMI. Esta placa también se puede ejecutar en una variedad de sistemas operativos en los que se incluyen a

Android, Lubuntu, Ubuntu, Debian y Raspbian. Entre sus principales características de hardware se tienen las siguientes: [15]

- Octa-core 1.8GHz
- 2 GB de memoria LPDDR3
- 8 GB de almacenamiento eMMC
- WiFi y Bluetooth integrados.



Figura 9 Banana Pi BPI-M3 [15]

2.5 Tecnologías de Software

En esta sección se presentan los diferentes componentes de sistema que se emplean para la elaboración del diseño de BigSwitch.

2.5.1 Ubuntu 16.04 Kernel 4.14

Para poder hacer uso del BananaPi BPI-R2 se ha hecho uso de la versión de Ubuntu 16.04 con kernel 4.14. Es preciso mencionar que el kernel implanta la tecnología DSA que abstrae, como interfaces Linux, los puertos finales gestionados por una arquitectura de switches-ASIC organizada en árbol.

En trabajos anteriores se ha optado por un equipo similar (BPI-R1) pero este no contaba con el hardware ASIC necesario para implantar la arquitectura DSA. Para hacer la diferenciación de paquetes era necesario realizar un encapsulado a nivel de capa 2, en donde se agregaban los diferentes segmentos de red en VLANs distintas y de esa manera poder etiquetar aquellos paquetes que necesitaran ir de una red a otra.

2.5.2 OpenvSwitch

OpenvSwitch es un proyecto de código abierto de los más utilizados que permite a los hipervisores poder virtualizar la capa de red. Esto puede ayudar en gran manera dentro de un entorno en el que se tienen una gran cantidad de máquinas virtuales que se están ejecutando en uno o más nodos físicos. La acción de las máquinas virtuales es conectarse a puertos virtuales a través de puentes virtuales, todo esto dentro de la capa de red virtualizada. [16]

Está diseñado para admitir la distribución en múltiples servidores físicos que son similares al conmutador virtual distribuido vNetwork de VMware o al Nexus 1000V de Cisco. Se considera como un conmutador virtual multicapa de calidad de producción. Otro enfoque que se tuvo en su diseño fue de permitir la automatización masiva de redes por medio de la extensión programática, al mismo tiempo que se admiten las interfaces y protocolos de administración estándar.

Para llevar el control del curso de los paquetes se ha utilizado el software OpenvSwitch en su más reciente versión (v2.10.1) para evitar posibles bugs que se puedan generar. [17]

Dentro de sus principales características, se tienen las siguientes: [18]

- Visibilidad en la comunicación entre máquinas virtuales mediante NetFlow, sFlow (R), IPFIX, SPAN, RSPAN y GRE-tunnelled mirror
- LACP (IEEE 802.1AX-2008)
- Modelo de VLAN 802.1Q estándar con enlace
- STP (IEEE 802.1D-1998) y RSTP (IEEE 802.1D-2004)
- Protocolos de tunelización múltiple (GRE, VXLAN, STT y Geneve, con soporte IPsec)
- Opciones de motor de reenvío de kernel y espacio de usuario

Se ha incluido este software en el desarrollo del trabajo debido a su capacidad de comportarse como un controlador OpenFlow en una red SDN, permitiendo de esta manera esta que sea introducido en las prácticas de laboratorio planteadas. Este software puede ejecutarse en un equipo SBC, el cual es la base del trabajo, ayudando a simular que fuese un switch real y estableciendo la conexión con el controlador ODL.

Capítulo 3

3 Diseño del BigSwitch para laboratorio docente

Según con los objetivos planteados, en este capítulo se encuentra la información referente al diseño del BigSwitch. Se ha llevado a cabo mediante el uso de dispositivos hardware (Banana Pi R2), además de tecnología SDN.

La configuración que se establece en cada uno de los dispositivos que se utilizan como conmutadores es la misma para todos, no se tiene una configuración específica para una parte de la red. Todos los conmutadores cumplen una única función, lo que permite unificar su configuración para de esta manera poder replicarla una vez que se haya terminado con uno de ellos. Aquel que podrá controlar lo que ocurra dentro de la red es el controlador que se encontrará alojado en un dispositivo BananaPi BPI-M3.

La información que se detalla en esta sección tiene que ver con el proceso que se ha llevado a cabo para crear una imagen en la que se contenga la construcción y compilación del kernel 4.14 con los módulos necesarios, instalación del sistema operativo, instalación y configuración del software OpenvSwitch, entre otros detalles que permiten el correcto funcionamiento del sistema.

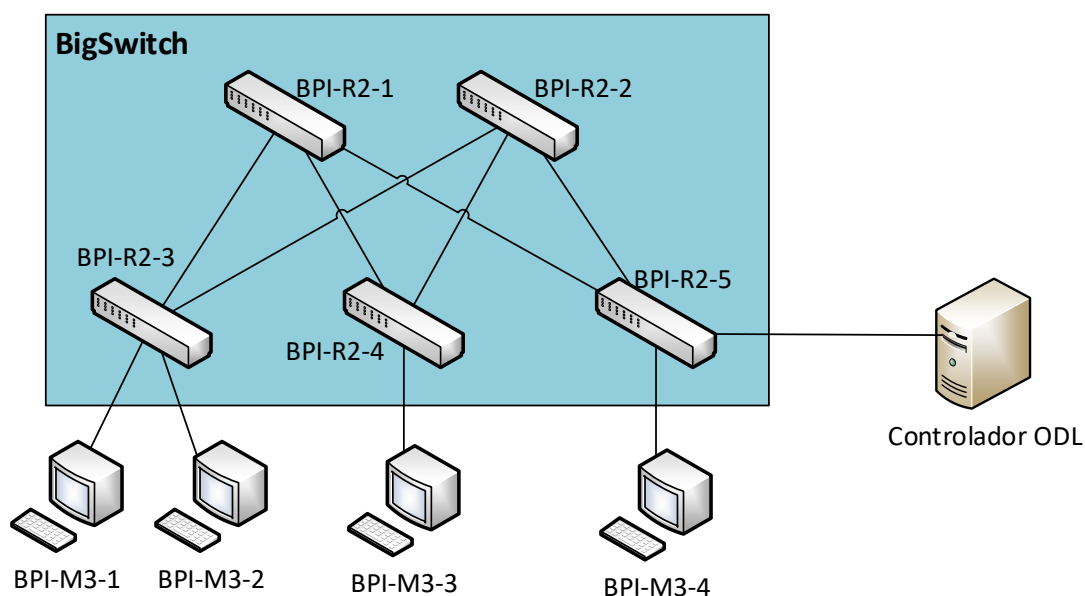


Figura 10 Esquema del BigSwitch

3.1 Preparación y configuración de los conmutadores

Para que se pueda disponer de un BigSwitch utilizando los dispositivos BananaPi BPI-R2, es necesario proceder cautelosamente con cada uno de los puntos que se detallan en esta sección. En caso de realizar algún tipo de prueba de lo que se ha elaborado en este trabajo, se recomienda que se lean adecuadamente y se efectúen correctamente las instrucciones indicadas, dado que se podrían dar fallos en el sistema debido a una ligera equivocación.

Dentro de las primeras instrucciones encontraremos que no será necesario trabajar directamente con el hardware BananaPi BPI-R2. Se debe hacer uso de un host, el cual tenga instalado el sistema operativo Ubuntu 16.04, ya que posee un conjunto de dependencias que facilitan los procedimientos, incluyendo un entorno de compilación cruzada para la CPU ARM que incluyen los equipos BananaPi.

3.1.1 Preparación del Host y descarga del repositorio

El host que se utilizará puede ser tanto una máquina física como una máquina virtual, pero en el entorno de preparación del laboratorio se hará uso de una máquina física. En cualquiera de los dos casos debe estar instalado de preferencia el Sistema Operativo Ubuntu 16.04. A modo de preparación de este host, lo que se busca es poseer todas las dependencias necesarias para la correcta ejecución de cada uno de los comandos a utilizar.

Una vez que se tiene el host se empezaría a ingresar en una terminal los siguientes comandos: [19]

Se empieza actualizando el sistema con el que estaremos trabajando.

```
host@ubuntu:~$ sudo apt-get update
host@ubuntu:~$ sudo apt-get upgrade
```

A continuación, se procede con la descarga de las herramientas que se van a necesitar para el desarrollo del trabajo.

```
host@ubuntu:~$ sudo apt-get install git build-essential automake
autoconf gcc-arm-linux-gnueabi u-boot-tools libc6-armhf-cross bc
libc6-dev libncurses5-dev libssl-dev bison flex pv
```

Revisar cuál es la versión de `gcc` que se tiene instalada, si existe una más nueva que la versión 5, entonces instalar esta versión.

```

host@ubuntu:~$ sudo apt-get install gcc-5-arm-linux-gnueabi
host@ubuntu:~$ sudo update-alternatives --install /usr/bin/arm-
linux-gnueabi-gcc arm-linux-gnueabi-gcc /usr/bin/arm-linux-
gnueabi-gcc-7 50
host@ubuntu:~$ sudo update-alternatives --install /usr/bin/arm-
linux-gnueabi-gcc arm-linux-gnueabi-gcc /usr/bin/arm-linux-
gnueabi-gcc-5 100
host@ubuntu:~$ sudo update-alternatives --config arm-linux-
gnueabi-gcc

```

Por último, clonar el repositorio oficial BananaPi que contiene el Kernel 4.14 en el host.

```

host@ubuntu:~$ sudo git clone https://github.com/BPI-SINOVOIP/BPI-
R2-bsp-4.14.git bsp
host@ubuntu:~$ cd bsp

```

3.1.2 Inclusión de módulos OpenvSwitch, compilación del bootloader y kernel 4.14

Ahora se tienen los archivos y herramientas necesarias para realizar la compilación del bootloader y del kernel 4.14. Previo a efectuar la compilación del kernel, es primordial agregar los módulos que utiliza OpenvSwitch para su correcto funcionamiento. Para cumplir este paso, ingresamos a la configuración del kernel mediante el comando

```

host@ubuntu:~$ ./build.sh

```

Y se selecciona la opción 4 “kernel configure”. Entonces nos aparecerá una pantalla de configuración como la que se muestra en la Figura 11.

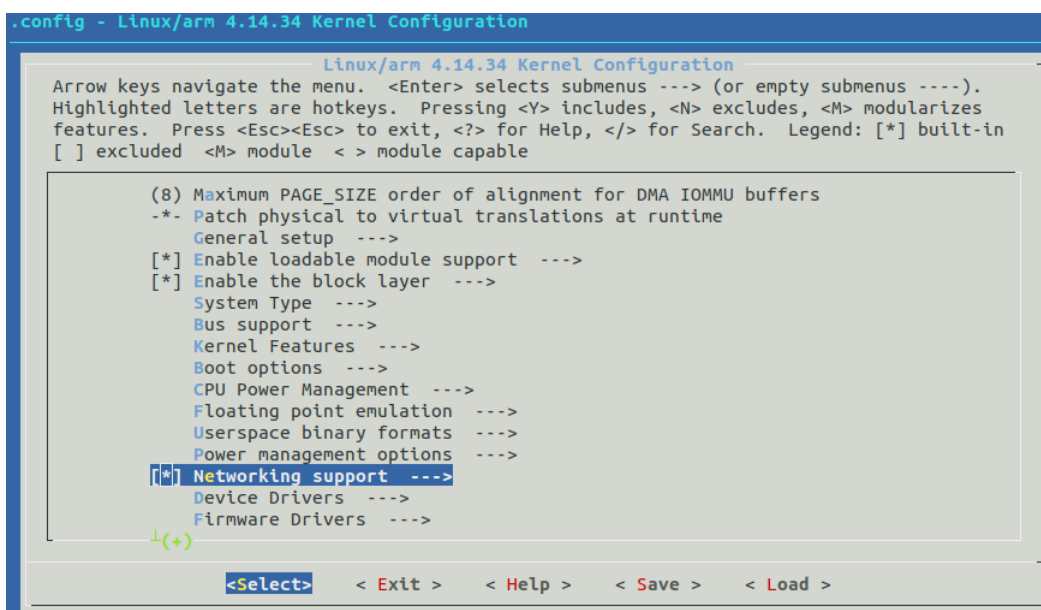


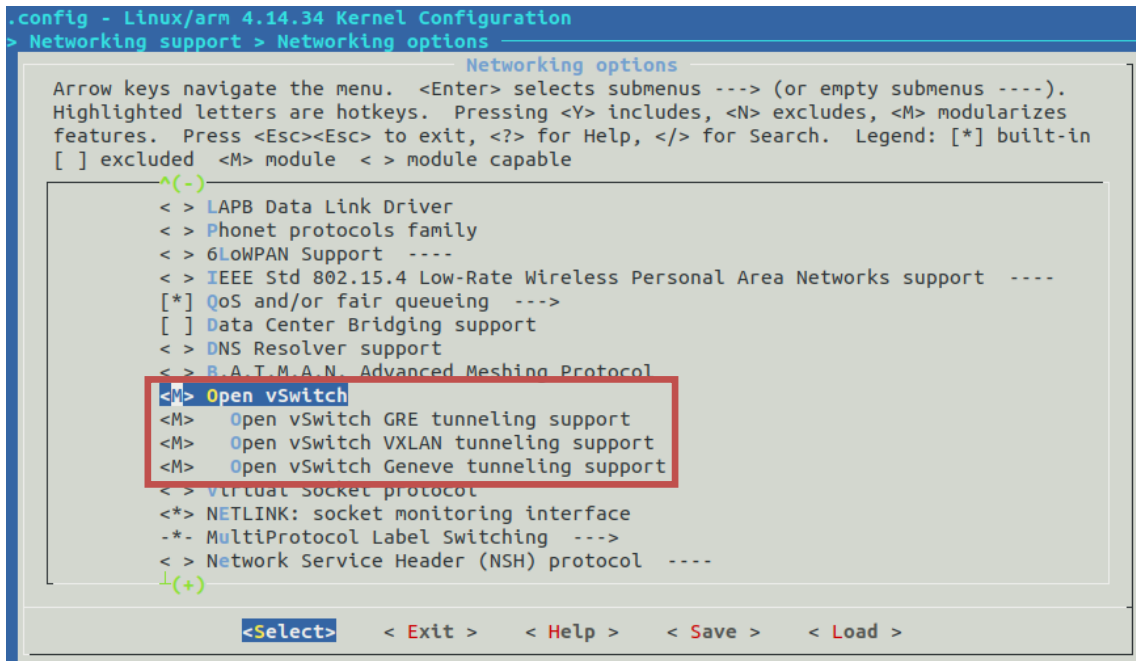
Figura 11 Menú configuración del kernel

Es este uno de los puntos más importantes para que nuestro entorno se configure de manera correcta. Este menú posee todas las opciones que se pueden agregar al entorno de operación del kernel, existen unas opciones activas por defecto, las cuales son necesarias. Nuestra labor es agregar aquellos módulos que vamos a utilizar.

Para habilitar el módulo de OpenvSwitch y todos aquellos que este requiere, es necesario dirigirse al directorio *Networking Support > Networking Options > OpenvSwitch* del menú de configuración. Cabe señalar que OpenvSwitch podrá ser incluido de forma modularizada y no compilado. Entonces se desplegarán más opciones de OpenvSwitch que también deberán ser incluidas, estas opciones son:

- OpenvSwitch GRE tunneling support
- OpenvSwitch VXLAN tunneling support
- OpenvSwitch Geneve tunneling support

Se puede dar por terminada esta parte de la configuración una vez que las opciones de OpenvSwitch se muestren como en parte resaltada de la *Figura 12*.



```
.config - Linux/arm 4.14.34 Kernel Configuration
> Networking support > Networking options
Networking options
Arrow keys navigate the menu. <Enter> selects submenus --- (or empty submenus ---).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in
[ ] excluded <M> module < > module capable
^(-)
< > LAPB Data Link Driver
< > Phonet protocols family
< > 6LoWPAN Support ----
< > IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support ----
[*] QoS and/or fair queueing ---
[ ] Data Center Bridging support
< > DNS Resolver support
< > B.A.T.M.A.N. Advanced Meshing Protocol
<M> Open vSwitch
<M> Open vSwitch GRE tunneling support
<M> Open vSwitch VXLAN tunneling support
<M> Open vSwitch Geneve tunneling support
< > Virtual socket protocol
<*> NETLINK: socket monitoring interface
-* MultiProtocol Label Switching ---
< > Network Service Header (NSH) protocol ----
+(-)
<Select> < Exit > < Help > < Save > < Load >
```

Figura 12 Módulos OpenvSwitch incluidos

Ahora se procede con la compilación del kernel y el bootloader con el mismo comando anterior pero esta vez seleccionando la opción 1, que indica *“Build all, uboot and kernel and pack to Download images.”*. Este proceso tomará unos minutos, dependiendo de las prestaciones que se tengan en el host.

3.1.3 Creación y configuración de la imagen

Para poder dar uso al kernel que se ha compilado en el apartado anterior, se ha considerado la creación de una imagen en la que se incluya tanto el sistema operativo Linux como el kernel en mención. Esto se lo puede conseguir mediante las siguientes instrucciones. Primero se empezará con la instalación de las herramientas a utilizar.

```
host@ubuntu:~$ sudo apt-get install qemu-user-static debootstrap  
binfmt-support parted zip
```

Se crea una imagen que en un principio estaría vacía y se le irá agregando todo lo necesario.

```
host@ubuntu:~$ sudo dd if=/dev/zero bs=1M count=7296 | pv | dd  
of=bpir2.img
```

Ahora se carga la imagen como un disco virtual nuevo, para ello se utiliza el comando *losetup*, el cual debe incluir el disco virtual a utilizar y el nombre de la imagen que se ha creado en el paso anterior. Primero se verifican cuáles son los discos virtuales (loop) que se están utilizando y se trabajará con uno que tenga una numeración mayor al último. En este caso se incluirá el loop8.

```
host@ubuntu:~$ sudo fdisk -l  
host@ubuntu:~$ sudo losetup /dev/loop8 bpir2.img
```

Con la imagen cargada, ahora se hacen las respectivas particiones y el formateo de las mismas. Las particiones deben ser creadas como primarias, una con formato fat32 y la otra con formato ext2.

```
host@ubuntu:~$ sudo parted -s /dev/loop8 mklabel msdos  
host@ubuntu:~$ sudo losetup /dev/loop8 bpir2.img  
host@ubuntu:~$ sudo parted -s /dev/loop8 unit MiB mkpart primary  
ext2 -- 356MiB 7295MiB  
host@ubuntu:~$ sudo partprobe /dev/loop8  
host@ubuntu:~$ sudo mkfs.vfat /dev/loop8p1 -I -n BPI-BOOT  
host@ubuntu:~$ sudo mkfs.ext4 -O ^has_journal -E stride=2,stripe-  
width=1024 -b 4096 /dev/loop8p2 -L BPI-ROOT  
host@ubuntu:~$ sudo sync  
host@ubuntu:~$ sudo parted -s /dev/loop8 print
```

El último comando es el que permite ver las 2 particiones que se han creado con todo su detalle en memoria. Parted es un comando que en este caso se lo utiliza para visualizar el inicio en memoria y su longitud en MB y no en MiB.

Ahora es momento de empezar a llenar el sistema de archivos root de Linux. Se va a proceder a realizar una instalación de un Debian GNU/Linux, teniendo como origen otro sistema Linux existente. La herramienta que a considerar en los próximos pasos es *debootstrap*, la cual es utilizada por el instalador de Debian y también es la manera oficial de instalar un sistema base Debian. Primero se crean los directorios necesarios y se los monta como se indica a continuación.

```
host@ubuntu:~$ sudo mkdir /mnt/rootfs
host@ubuntu:~$ sudo mount /dev/loop8p2 /mnt/rootfs
host@ubuntu:~$ sudo mkdir /mnt/rootfs/boot
host@ubuntu:~$ sudo mount /dev/loop8p1 /mnt/rootfs/boot
```

Luego para poder usar el comando *debootstrap*, es necesario que saber cuál es la distribución que se desea descargar, indicar el procesador con el que se cuenta en el dispositivo donde será usada la imagen y la carpeta o directorio que se quiere realizar la operación.

Qemu es la herramienta que se utiliza para emular un Sistema Operativo dentro de otro, sin la necesidad crear una nueva partición en el disco, tan solo se necesita el directorio.

```
host@ubuntu:~$ sudo debootstrap --arch=armhf --foreign xenial
/mnt/rootfs
host@ubuntu:~$ sudo cp /usr/bin/qemu-arm-static
/mnt/rootfs/usr/bin/
```

Chroot es otro de los comandos más importantes que van a ser ejecutados en esta sección. Su función primordial es la de permitir configurar un directorio dado como *root* del sistema de ficheros, a partir de ahí todo fichero o directorio fuera del *chroot* quedaría totalmente inaccesible. Solo puede ser invocado por el usuario root del sistema, dado que se tiene un gran alcance en los comandos que se ingresen.

Aquel directorio que se lo ejecuta como *chroot*, se lo denomina *jaula chroot*. De esta

```
host@ubuntu:~$ sudo chroot /mnt/rootfs
I have no name!@ubuntu:/# export LANG=C
I have no name!@ubuntu:/# /debootstrap/debootstrap --second-stage
```

manera se procederá a ejecutar esta función para el directorio */mnt/rootfs*.

Desde este momento, todos los comandos que se ejecuten y los procesos que se inicien en este terminal servirán para modificar el sistema operativo dentro de la imagen creada. Por lo cual se continúa con la creación del *sources.list*, en donde se detallan o enlistan los repositorios disponibles de aquellos paquetes de software los cuales pueden ser instalados, actualizados, buscados, removidos, entre otras opciones.

```

I have no name!@ubuntu:/# echo "" > /etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial main restricted">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial main restricted">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial universe">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial universe">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial multiverse">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial multiverse">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial-updates main restricted">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial-updates main restricted">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial-updates universe">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial-updates universe">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial-security main restricted">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial-security main restricted">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb http://ports.ubuntu.com/ubuntu-ports/
xenial-security multiverse">>/etc/apt/sources.list
I have no name!@ubuntu:/# echo "deb-src http://ports.ubuntu.com/ubuntu-
ports/ xenial-security multiverse">>/etc/apt/sources.list
I have no name!@ubuntu:/# apt-get update
I have no name!@ubuntu:/# apt-get upgrade
I have no name!@ubuntu:/# apt-get dist-upgrade

```

Así también, se agregarán unos paquetes que se han considerado como esenciales para las actividades previstas, pero se pueden agregar los que necesite.

```

I have no name!@ubuntu:/# apt-get install sudo nano openssh-server
locales ntpdate http pv

```

Es importante también establecer un hostname al equipo como se detalla en las siguientes líneas.

```

I have no name!@ubuntu:/# echo "bpi-r2" >/etc/hostname
I have no name!@ubuntu:/# nano /etc/hosts
-> add "bpi-r2" to line with 127.0.0.1

```

Además, se puede agregar la opción de establecer una contraseña root en caso de querer habilitar un login root.

```

I have no name!@ubuntu:/# passwd

```

En este punto se pueden crear uno o más usuarios. Tome en cuenta que aún estamos utilizando el comando *chroot*, es decir, que estos comandos no modificarían a su sistema operativo local.

```
I have no name!@ubuntu:/# useradd -m -G users,sudo,ssh -s /bin/bash
bpi
I have no name!@ubuntu:/# passwd bpi
```

Se configuran también los *locales*, agregar aquellos que considere necesarios.

```
I have no name!@ubuntu:/# locale-gen en_US
I have no name!@ubuntu:/# locale-gen en_US.UTF-8
I have no name!@ubuntu:/# dpkg-reconfigure locales
I have no name!@ubuntu:/# dpkg-reconfigure tzdata
I have no name!@ubuntu:/# update-locale LANG=en_US.UTF-8
```

Adicionalmente, configurar el *fstab*

```
I have no name!@ubuntu:/# nano /etc/fstab
-> proc /proc proc defaults 0 0
    LABEL=BPI-BOOT /boot vfat errors=remount-ro 0 1
    LABEL=BPI-ROOT /ext4 defaults noatime 0 0
```

Se agrega también la configuración para sus interfaces o parte de ellas, luego se tendrá que modificar para cada dispositivo en el que se instale esta imagen.

```
I have no name!@ubuntu:/# nano /etc/network/interfaces
-> auto eth0
    iface eth0 inet manual
        pre-up ip link set $IFACE up
        post-down ip link set $IFACE down
auto eth1
    iface eth1 inet manual
        pre-up ip link set $IFACE up
        post-down ip link set $IFACE down
auto lan0
    iface lan0 inet static
        address 192.168.0.10
        netmask 255.255.255.0
        pre-up ip link set $IFACE up
        post-down ip link set $IFACE down
```

En este tipo de entornos es necesario que se encuentre habilitado el acceso vía ssh, por lo que también se incluye la configuración a continuación.

NOTA: Esta configuración de red no será la definitiva para la construcción del “bigswitch”

```
I have no name!@ubuntu:/# mkdir /root/.ssh
I have no name!@ubuntu:/# chmod 0700 /root/.ssh
I have no name!@ubuntu:/# touch /root/.ssh/authorized_keys
I have no name!@ubuntu:/# chmod 0600 /root/.ssh/authorized_keys
I have no name!@ubuntu:/# nano /etc/ssh/sshd_config
-> PubkeyAuthentication yes
    PasswordAuthentication no
```

En este punto termina la configuración de la imagen, se sale del modo *chroot* y se elimina el *qemu*.

```
I have no name!@ubuntu:/# exit
host@ubuntu:~$ sudo rm /mnt/rootfs/usr/bin/qemu-arm-static
```

Para tener todos los archivos necesarios que integran la imagen, faltarían agregar ciertos archivos en la partición boot, los archivos de kernel y el bootloader al sistema de ficheros del root.

```
host@ubuntu:~$ sudo tar -x -f bsp14/SD/BPI-BOOT-bpi-r2.tgz --keep-
directory-symlink -C /mnt/rootfs/boot
host@ubuntu:~$ sudo tar -x -f bsp14/SD/4.14.34-BPI-R2-Kernel.tgz --
keep-directory-symlink -C /mnt/rootfs
host@ubuntu:~$ sudo tar -x -f bsp14/SD/4.14.34-BPI-R2-Kernel-
net.tgz --keep-directory-symlink -C /mnt/rootfs
host@ubuntu:~$ sudo tar -x -f bsp14/SD/BOOTLOADER-bpi-r2.tgz --
keep-directory-symlink -C /mnt/rootfs
```

Luego se realiza un *blacklisting* del módulo de botón de encendido. El *blacklisting* es un mecanismo que se utiliza para evitar que un módulo del Kernel se cargue. Esto tiene como objetivo principal evitar que dos módulos del Kernel intenten controlar una misma parte del hardware, si llegara a ocurrir provocaría un conflicto.

```
host@ubuntu:~$ sudo echo "blacklist mtk_pmic_keys" >
/mnt/rootfs/etc/modules-load.d/mtk_pmic_keys.conf
```

Se desmontan las particiones que se han utilizado.

```
host@ubuntu:~$ sudo umount /mnt/rootfs/boot
host@ubuntu:~$ sudo umount /mnt/rootfs
```

A través del repositorio de archivos de BPI en Github se obtienen los *header blocks*.

```
host@ubuntu:~$ sudo gunzip -c BPI-R2-HEAD440-0k.img.gz | dd
of=/dev/loop8 bs=1024 seek=0
host@ubuntu:~$ sudo gunzip -c BPI-R2-HEAD1-512b.img.gz | dd
of=/dev/loop8 bs=512 seek=1
```

Se continúa con la escritura del preloader y el u-boot bootloader.

```
host@ubuntu:~$ sudo dd if=bsp14/mt-pack/mtk/bpi-  
r2/bin/preloader_iotg7623Np1_sd_1600M.bin of=/dev/loop8 bs=1024  
seek=2  
host@ubuntu:~$ sudo dd if=bsp14/u-boot-mt/u-boot.bin of=/dev/loop8  
bs=1024 seek=320  
host@ubuntu:~$ sudo sync
```

Se elimina el loop que se agregó al principio, para esto utilizamos el siguiente comando.

```
host@ubuntu:~$ sudo losetup -d /dev/loop8
```

Ya se tiene una imagen lista para cargar a los dispositivos BananaPi BPI-R2 y que se encuentra totalmente operativa. Adicionalmente, se puede crear un archivo zip de la imagen que se tiene ahora.

```
host@ubuntu:~$ sudo zip bpir2.img.zip bpir2.img
```

3.1.4 Instalación del Sistema Operativo

Una vez que tiene la imagen lista (con el sistema operativo), el próximo paso a seguir es su debida instalación. Este procedimiento consta de copiar la imagen creada a una tarjeta SD previamente formateada. Este proceso se lo realiza con el comando *dd* de la siguiente manera.

```
host@ubuntu:~$ dd if=bpir2.img of=/dev/mmcblk0
```

El proceso se tardará unos minutos en ejecutarse. Al finalizar ya se puede extraer la tarjeta SD e instalarla en el dispositivo BananaPi BPI-R2. Para comprobar que se han cargado los archivos correctamente, se enciende el dispositivo manteniendo pulsado el botón de encendido durante unos 5 segundos.

El sistema operativo que se ha escogido para la operación de los conmutadores es el Ubuntu 16.04. Se lo puede comprobar con el comando *uname -r*. La modificación del fichero de configuración de red */etc/network/interfaces* es importante efectuar en cada uno de los dispositivos. Vendrá cargada por defecto la siguiente configuración, la cual es general.

```
# nano /etc/network/interfaces
auto eth0
iface eth0 inet manual
    pre-up ip link set $IFACE up
    post-down ip link set $IFACE down
auto eth1
iface eth1 inet manual
    pre-up ip link set $IFACE up
    post-down ip link set $IFACE down
auto lan0
iface lan0 inet static
    address 192.168.0.10
    netmask 255.255.255.0
    pre-up ip link set $IFACE up
    post-down ip link set $IFACE down
```

Dependiendo de las necesidades, se cambiaría la IP de las Lans para cada dispositivo. Después de esto, se levantan aquellas interfaces que se estén utilizando.

```
pi@switch:~$ sudo ip link set lan0 up
pi@switch:~$ sudo ip link set lan1 up
pi@switch:~$ sudo ip link set lan2 up
pi@switch:~$ sudo ip link set lan3 up
```

Ahora ya se tiene a disposición el dispositivo BananaPi con sus cinco puertos Ethernet, esperando por el software que permita conmutar el tráfico que curse por cada uno de ellos.

3.1.5 Instalación y configuración de OpenvSwitch

Dado que el sistema operativo con el que cuenta el dispositivo BananaPi es una distribución que está basada en Debian, entonces dispone del comando *apt-get* que permite la instalación de paquetes en el sistema. Conociendo esto, y además de que se encuentran disponibles los *.deb* para OpenvSwitch, su instalación resulta realmente sencilla.

```
pi@switch:~$ sudo apt-get update
pi@switch:~$ sudo apt-get upgrade
pi@switch:~$ sudo apt-get install openvswitch*
```

Se coloca un asterisco junto a OpenvSwitch porque son varios *.deb* que se incluirán en la instalación. Al finalizar la ejecución del comando ya se tendrá el software instalado y listo para ser usado. Ahora se da paso a la creación y configuración del switch virtual.

```

pi@switch:~$ sudo ovs-vsctl add-br br0
pi@switch:~$ sudo ovs-vsctl set-controller br0 tcp:<ip_controlador>:6653
pi@switch:~$ sudo ovs-vsctl set manager tcp:<ip_controlador>:6640
pi@switch:~$ sudo ovs-vsctl add-port br0 wan
pi@switch:~$ sudo ovs-vsctl add-port br0 lan0
pi@switch:~$ sudo ovs-vsctl add-port br0 lan1
pi@switch:~$ sudo ovs-vsctl add-port br0 lan2
pi@switch:~$ sudo ovs-vsctl add-port br0 lan3

```

Toda la configuración que se ha añadido permite que el switch virtual tenga similares funcionalidades que un conmutador tradicional. Con la configuración actual el conmutador podrá actuar de manera autónoma hasta el momento en que se conecte el controlador a la red. Una vez que se conecta el controlador, el tráfico que pase por la red será manejado por el mismo.

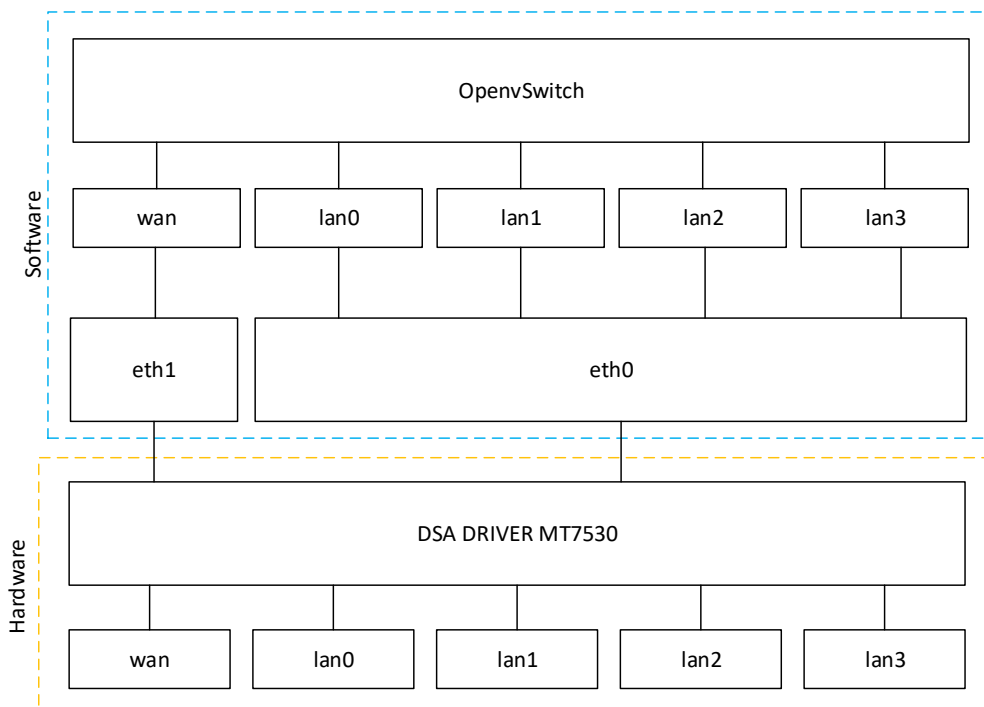


Figura 13 Vista de la configuración Ethernet con OpenvSwitch a nivel de hardware y software

Con respecto a la anterior versión a la que se ha usado en este trabajo, el dispositivo BananaPi BPI-R1, se ha dado un gran paso al poder incluir en su arquitectura un procesador desarrollado por Mediatek (MT7530). Esta inclusión ha representado que se cuente con una arquitectura DSA en el dispositivo, y de esta manera obtener el total control de los puertos. No hace falta incluir configuraciones que involucren el uso de *swconfig*, en la Figura 13 se puede ver una representación de cómo se maneja compone esta arquitectura.

3.2 Preparación del controlador

Para poder manejar conceptos de SDN, es necesario que la red contenga un servidor en el que se incluya un controlador. Este servidor debe ser un dispositivo que cuente con el sistema operativo Ubuntu y poseer la suficiente memoria que permita la gestión de la conmutación del tráfico.

En este servidor, que puede ser un ordenador o una BananaPi BPI-M3, se va a instalar el software OpenDaylight y también se incluirán módulos que permitan el funcionamiento básico de la red.

3.2.1 Instalación del Software OpenDaylight

Para obtener el software OpenDaylight, fue a través del sitio web oficial y proceder con la descarga. Lo puede realizar a través del siguiente enlace <https://docs.opendaylight.org/en/latest/downloads.html> que cuenta con una versión estable del controlador. Entonces para realizar la instalación es necesario seguir ciertos pasos:

```
server@ubuntu:~$ mkdir ODL
server@ubuntu:~$ unzip Descargas/distribution-karaf-0.5.3-Boron-SR3.zip
server@ubuntu:~$ mv Descargas/distribution-karaf-0.5.3-Boron-SR3 ODL
server@ubuntu:~$ ./ODL/distribution-karaf-0.5.3-Boron-SR3/bin/karaf
```

3.2.2 Instalación de módulos OpenDaylight

Para realizar la instalación de los módulos es necesario ingresar a la consola del software, esto se lo consigue primero ubicándose en el directorio en donde se ha instalado el software y utilizar el comando

```
server@ubuntu:~$ ./karaf
```

Se tiene instalado el software pero se necesita también que este se pueda comunicar con los conmutadores virtuales que se hayan incluido en la red, para lo cual se van a utilizar dos interfaces en la Southbound API. Una de esas interfaces es la que se encarga de controlar los flujos OpenFlow que incluirán en cada uno de los conmutadores y la otra interfaz su función será de actuar como el manager de OpenvSwitch. De manera predeterminada se activa el primer módulo mencionado, mientras que el segundo de ellos, el *odl-ovsdb*, debe ser instalado. A continuación, se indican aquellos comandos necesarios para agregar los módulos más útiles.

```
karaf> feature:install odl-ovsdb-southbound-impl-ui
karaf> feature:install odl-l2switch-all
karaf> feature:install odl-dlux-all
karaf> feature:list -i
```


3.3 Dificultades con el diseño

Para llegar a obtener el resultado final de este trabajo, se presentaron inconvenientes o dificultades en diferentes campos, los cuales se fueron solucionando a medida que se iba consultando más material disponible. Las pruebas y errores que surgieron en cada idea planteada, permitieron comprender de mejor manera la arquitectura con la que se contaba e ir esclareciendo poco a poco las dudas que se generaban. Aquellos problemas que no se pudieron resolver, se encuentran planteados como propuesta de trabajos futuros.

3.3.1 ¿Qué kernel utilizar?

Esta fue una de las primeras interrogantes que surgieron. Se presentó como interrogante porque existe una imagen en la página web oficial de los dispositivos BananaPi (SINOVOIP) en la que se utiliza un Sistema Operativo Ubuntu 17.04 con un kernel 4.4.70. Bajo estas circunstancias, el dispositivo funciona correctamente, pero esto no era suficiente ya que se necesitaba hacer uso del driver DSA y no se encuentra considerado en esta versión de kernel.

El driver DSA que ha sido descrito en un apartado del capítulo 2, es necesario para poder tener el control de cada uno de los puertos Ethernet por separado, al mostrarlos como interfaces de red Linux independientes. El dispositivo posee integrado un procesador Mediatek MT7530, el cual evita que la conmutación de los paquetes sea ejecutada por la CPU y más bien se encarga de esta función exclusivamente.

La integración del driver al sistema operativo Ubuntu viene a partir del kernel 4.14, es decir que la versión oficial que brinda el fabricante no era de mucha ayuda, sabiendo que en este caso no aporta para las necesidades solicitadas. Por lo tanto, se tuvieron que buscar nuevas alternativas que permitan suplir los requerimientos. Entre los cuales se obtuvo una imagen compatible con BananaPi BPI-R2, con sistema operativo Ubuntu 18.04 y además incluía el kernel 4.14. Fácilmente se pudo pensar que se había obtenido una imagen que cumplía los requisitos necesarios para montar la arquitectura del BigSwitch.

Dada la cantidad de información con la que se contaba con respecto al kernel 4.14 y la integración del driver SDA, se decidió optar por esta alternativa que fue con la que se terminó elaborando este material. Además, desde esta versión se han añadido varias mejoras que incluyen la compatibilidad con componentes de hardware para BananaPi y mejoras en su rendimiento en general.

3.3.2 Instalación de OpenvSwitch

Aquí se cuenta parte de la labor que se tuvo que realizar para la instalación del software, específicamente OpenvSwitch. En un principio, se realizaron una larga serie de intentos que resultarían fallidos, tanto haciendo la instalación a través de los

paquetes *.deb*, como compilando el software hasta la debida generación de los archivos *.ko*.

Cuando se intentaba instalar OpenvSwitch desde los paquetes *.deb*, es decir, utilizando el comando *apt-get install*, se reflejaban errores durante el proceso. Estos errores consistían en la no presencia de archivos headers, lo que se intentó solucionar mediante la compilación de los archivos fuente de OpenvSwitch.

Al utilizar los archivos fuentes de OpenvSwitch y compilarlos, se evidenció que en el directorio */lib/modules/4.14.48-main* no se contaba con los ficheros de aquella versión del kernel. Por lo tanto, se realizó una clonación del fichero GitHub donde se encontraba alojado y se procedió a realizar la compilación del programa. Para culminar con la instalación hace falta portar, la siguiente sección detalla parte de este procedimiento.

3.4 Portar OpenvSwitch a un nuevo hardware o software

Existe la posibilidad de portar fácilmente OpenvSwitch a nuevo software y plataformas hardware. Conviene que se tenga portabilidad ya que, si es en gran proporción, será menor la dependencia que tendría el software con respecto al hardware. Al utilizar el sistema operativo Linux, el propio OpenvSwitch proporciona una gran parte de la portabilidad que se necesita para este trabajo. Para poder realizar este proceso de portabilidad se necesita conocer cuál es la arquitectura sobre la que se está trabajando.

En la *Figura 14* se muestra la arquitectura de OpenvSwitch desde la perspectiva de la portabilidad.

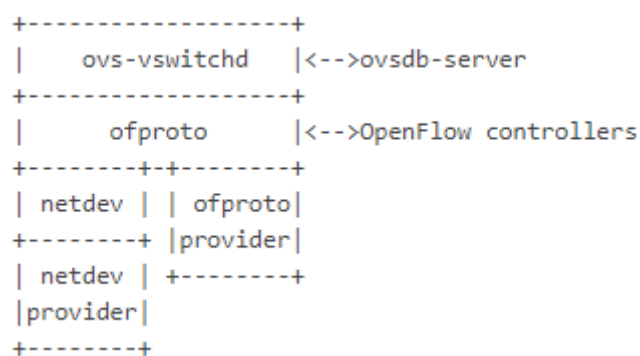


Figura 14 Arquitectura OpenvSwitch [20]

De donde se tiene que *ovs-vswitchd* es el encargado de leer la configuración de OpenvSwitch desde el *ovsdb-server* a través de un *IPC channel*, y luego envía esta información a la librería *ofproto*. *Ofproto* es una librería de OpenvSwitch que

implementa un conmutador OpenFlow, a la vez se comunica con los controladores OpenFlow por medio de la red y con el conmutador hardware o software a través de un *ofproto provider*. Por último, se tiene el *netdev* que es una librería cuya función es la de aislar la interacción con interfaces Ethernet. [20]

Es muy común que para realizar la portabilidad lo que se necesite implementar sea el *netdev provider*, o se podría dar el caso de que se requiera también la implementación del *ofproto provider*, pero esto va a depender del tipo de portabilidad que se esté elaborando y el rendimiento deseado. El *netdev provider* es el que se encarga de la inclusión entre un sistema operativo y la interfaz hardware, o la integración de OpenvSwitch con el sistema operativo. Así es como se le permite a OpenvSwitch abrir cada puerto de un switch como *netdev*.

Se cuenta con varias implementaciones de *netdev* que pueden ayudar a comprender de mejor manera esta parte de la portación. Por ejemplo, la que se ha usado en el desarrollo de este trabajo ha sido la *netdev-linux.c*, que implementa funcionalidades *netdev* para dispositivos de red Linux. Es de gran aporte porque posee todas las funciones, entre las cuales se tiene la capacidad de informar la dirección Ethernet del hardware e inspeccionar la tabla ARP.

Así también, se cuenta con otra implementación que se corresponde a *netdev-vport.c*, la cual se encarga de brindar el soporte para la manipulación y configuración de los puertos virtuales. En caso de que estas implementaciones no satisfagan las necesidades del usuario entonces se debe empezar a escribir algo de código en el que se incluye o bien un *ofproto provider* o un *dpif provider*, esto va a depender de diferentes factores [20].

Si solo se desarrolla el *ofproto provider* entonces se puede obtener una mayor ventaja al hardware y sus prestaciones. Si se desarrolla un *dpif provider* la ventaja obtenida será con respecto a las implementaciones integradas de OpenvSwitch de bonding, LACP, 802.1ag, entre otras.

OpenvSwitch tiene integrado un *ofproto provider*, llamado *ofproto-dpif*, el cual se encarga de manejar o gestionar las tablas de flujo sobre el *dpif*, considerado como un datapath interface. Muchas de las cosas que se necesitan para obtener un conmutador openflow ya las tiene implantadas y no dependen de su hardware, solo restaría por implementar la tabla de openflow. En este trabajo se ha conseguido que se realice el openflow solo de los paquetes difusivos y no los unicast porque no se encontraba bien integrado el OpenvSwitch con el DSA.

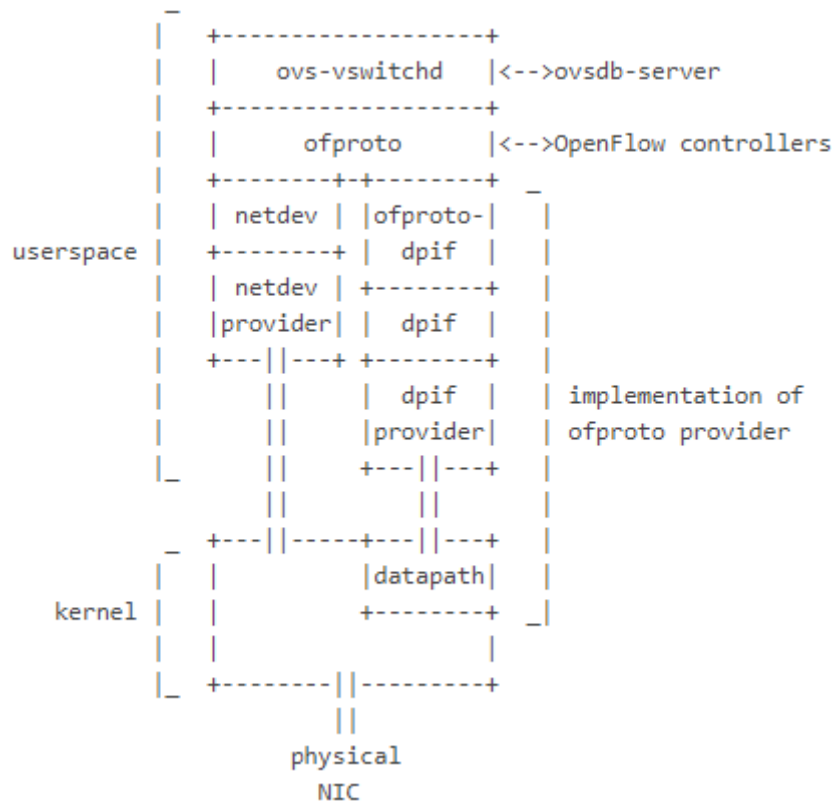


Figura 15 Adaptación de *dpif provider* a la arquitectura OpenvSwitch [20]

En esta parte de la adaptación es donde se hace uso de una estructura llamada *dpif_class*, ubicada en el directorio *lib/dpif-provider.h*, la cual indica cuáles son las interfaces necesarias para implementar un *dpif provider* para un nuevo hardware o software. Entre sus implementaciones elaboradas se tiene a *lib/dpif-netlink.c* que se encarga de interactuar con un módulo de kernel de OpenvSwitch, el cual realiza el trabajo de conmutación y pasando aquellos paquetes que no coinciden con ninguna entrada de la tabla de flujo al userspace; también se tiene a *lib/dpif-netdev.c* como la encargada de realizar todo el switching internamente.

Lo que realiza DSA es presuponer un conmutador Ethernet transparente, que consiste en aprender, olvidar e inundar, como comportamiento del ASIC hardware de conmutación. Para portar OVS es necesario escribir el módulo *dpif provider*, el cual programará adecuadamente aquel ASIC hardware de conmutación para que, junto al *datapath* del kernel Linux se pueda implantar el forwarding con la tabla de flujos simple que se requiere.

Capítulo 4

4 Prácticas de Laboratorio

Las prácticas que se detallan en este capítulo constan de actividades que servirán a los alumnos para entender los conceptos impartidos en clase sobre lo que es un BigSwitch y las diferentes propiedades que posee. Además, estas prácticas han sido diseñadas para un laboratorio de hasta un máximo de quince estaciones de trabajo, para el que se requieren específicamente los materiales mencionados en el siguiente apartado.

4.1 Materiales a utilizar

4.1.1 Banana Pi BPI-R2

Dispositivo que es utilizado como un elemento de red capa dos, es decir, que emplea la función de tramas Ethernet. Cada una de sus cinco interfaces físicas poseen una capacidad de 1Gbps. Por cada estación de trabajo se necesitan cinco BPI-R2 para lograr establecer las prácticas, lo que representaría un total de **75 BPI-R2** a utilizar. De donde también se necesitarían 75 fuentes de 12V cada una.

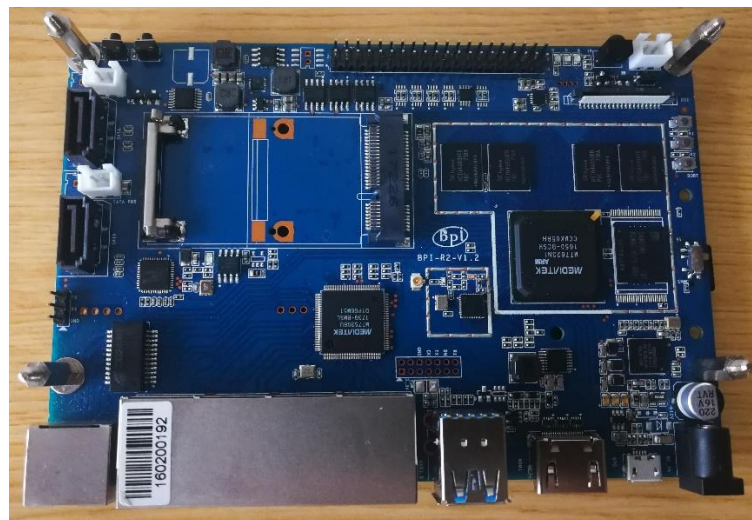


Figura 16 Dispositivo Banana Pi BPI-R2



Figura 17 Fuente de Voltaje

4.1.2 Banana Pi BPI-M3

Dispositivo que es utilizado como un host, cliente o terminal durante la ejecución de las prácticas. También se usará como soporte al controlador SDN en las prácticas que lo requieran. Su interfaz Ethernet posee una capacidad de 1Gbps. Por cada puesto de trabajo se necesitan cuatro dispositivos, entonces para todo el laboratorio se necesitarían 60 de estas placas. Así también, se deben considerar las fuentes de voltaje para los equipos, similares a las del BPI-R2, en este caso con un total de 60.

4.1.3 Cable de red

Para poder realizar la interconexión entre los dispositivos BPI-R2 y los BPI-M3 se necesitan cables UTP (Unshielded Twisted Pair) Cat 5e o Cat 6. Según el diseño de las prácticas, es necesario que como mínimo se tengan 165 de estos cables, con una longitud de 60 cm cada uno, medida considerada suficiente para la interconexión de los dispositivos. Con conectores RJ-45 en cada extremo, es decir, que se hace uso de 330 de estos tipos de conectores.

Es importante realizar una prueba a los cables para verificar que tengan conducción en cada uno de sus pines, de esta manera se descarta que exista un problema físico en la conectividad.



Figura 18 Cable de red

4.1.4 Tarjeta Micro SD

Los dispositivos Banana pi no poseen un disco duro, pero sí un espacio de almacenamiento interno; pero dado el dinamismo de las configuraciones a realizar resulta necesario integrar a su placa un dispositivo de almacenamiento externo para poder cargar el sistema que permita obtener un entorno de desarrollo apto para realizar las prácticas.

Para este caso, se ha considerado la inclusión de tarjetas Micro SD de 16 GB. Capacidad suficiente para las actividades que se necesitan elaborar. Antes de utilizarlas es recomendable realizar un formateo de las mismas. Dado que se utilizan una tarjeta micro SD por cada dispositivo Banana Pi, eso representa un total de 135 tarjetas, 75 para los BPI-R2 y 60 para los BPI-M3.



Figura 19 Tarjeta Micro SD 16GB

4.1.5 Dispositivos adicionales

Para poder trabajar con los dispositivos mencionados anteriormente también se necesita el uso de otros que se detallan a continuación, con su número total:

- 30 Monitores con entradas HDMI
- 30 Cables HDMI
- 30 Teclados
- 30 Ratones

4.1.6 Puesto de trabajo

Un puesto de trabajo está descrito como el espacio en el que se cuenta con los dispositivos e infraestructura necesarios para desarrollar las prácticas de laboratorio. Cada puesto de trabajo está diseñado para que preferiblemente trabajen dos estudiantes, pero pueden ser hasta tres. Debido a la infraestructura del laboratorio, los dispositivos BananaPi deben estar en una sección aparte en donde puedan ser conectados todos a la vez y evitar que en cada práctica se consuma tiempo realizando este trabajo.

Se hará uso de servidores alojados en dispositivos BananaPi BPI-M3, que se localizarán en una de las dos computadoras con las que se contará en cada estación de trabajo. Lo que representaría un uso de máximo 15 servidores, ejecutándose en un BPI-M3 por estación de trabajo.

4.2 Práctica 1: BigSwitch Leaf-Spine y pruebas de algoritmos

Objetivos

Por medio de esta práctica se desea analizar en un BigSwitch con dispositivos reales, el funcionamiento y características de los algoritmos de asignación de recursos que se encuentran disponibles en este tipo de arquitecturas. Estos algoritmos son Spanning Tree Protocol y Shortest Path Bridging.

ES NECESARIO QUE ANTES DE EMPEZAR LAS PRÁCTICAS SE LEAN DETENIDAMENTE LOS PASOS A REALIZAR

Conocimientos necesarios:

- Protocolos STP y SPB
- Topología Leaf-Spine. Switch transparente.
- Herramientas que permitan realizar pruebas de conectividad, ping e iperf.

Descripción del escenario de la práctica

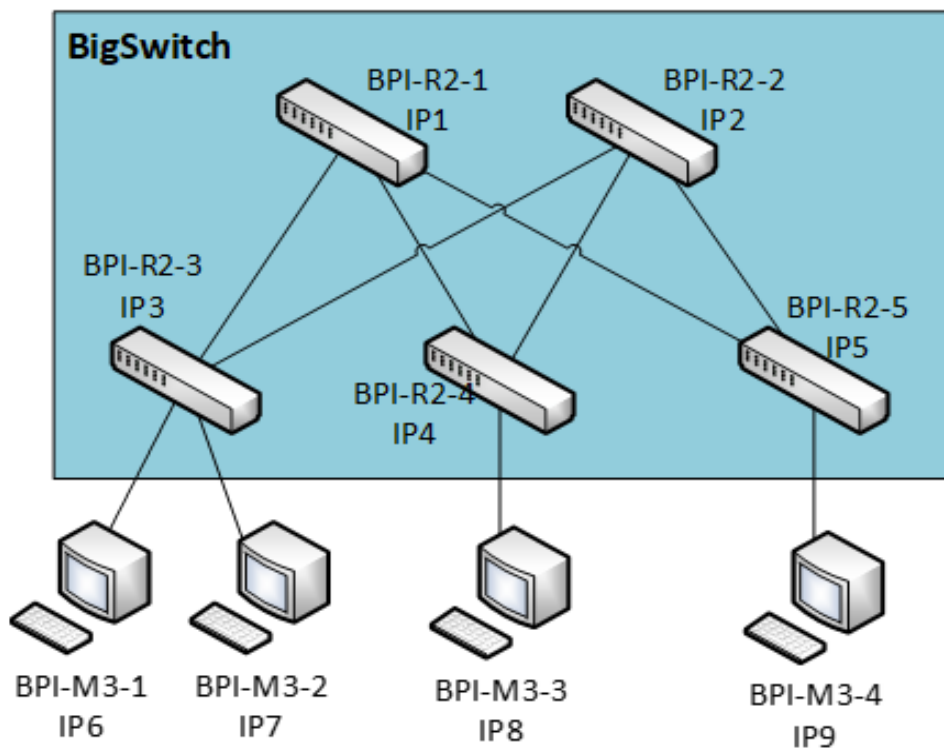


Figura 20 Esquema del escenario 1

Para la elaboración de esta práctica se tienen a disposición cinco placas BananaPi BPI-R2 por puesto de trabajo. En cada una de las placas se van a encontrar cinco puertos Giga-Ethernet, los cuales se van a poder utilizar como Switch por medio del software OpenvSwitch que ya se encuentra instalado en la placa. Las interconexiones se realizarán según lo descrito en el esquema de red que se presenta en la Figura del escenario 1.

Adicionalmente se tienen cuatro placas BananaPi BPI-M3, las cuales cuentan con un sistema operativo Ubuntu MATE (versión original del fabricante SINOVOIP) y con las herramientas de prueba, ping e iperf, en funcionamiento. La tabla que se muestra a continuación muestra las IPs asignadas, las cuales servirán para configurar los dispositivos de red.

IP1	192.168.1.11	Máscara /24
IP2	192.168.1.12	
IP3	192.168.1.13	
IP4	192.168.1.14	
IP5	192.168.1.15	
IP6	192.168.1.21	
IP7	192.168.1.22	
IP8	192.168.1.23	
IP9	192.168.1.24	

TAREA 1: Preparación del BigSwitch

Para que se puedan realizar las pruebas de los algoritmos que se permiten en una topología que involucra a un BigSwitch se deben iniciar los conmutadores y conectarlos entre sí. Esta interconexión debe seguir una topología leaf-spine, tal como se muestra en la Figura del escenario 1. A continuación, se indican los pasos a realizar para configurar uno de los conmutadores.

1. Conecte el conmutador BPI-R2 a configurar a la corriente. Si el conmutador ya está iniciado, reinícielo para empezar a trabajar (botón de “reset” de la placa).
2. Acceda, vía ssh, al conmutador, según las IPs dadas. Las credenciales de acceso son: *user: pi, password: bananapi*.
3. Antes de realizar cualquier configuración, a través de la conexión anterior, comprobar que no se conserven configuraciones previas como puede ser el caso de un conmutador virtual. Para esto utilice el comando *ovs-vsctl show*,

este comando le mostrará los conmutadores virtuales que posee el dispositivo. Si existiera algún elemento, elimínelo con el comando `ovs-vsctl del-br <nombre_bridge>`

4. Con el comando `ip link` compruebe el estado de las interfaces. Las cinco interfaces que posee deben estar "up", las cuales son: wan, lan0, lan1, lan2 y lan3. Si se encuentran en estado "down", primero ingresar el comando `sudo ip link set eth0 up`, y luego el comando `sudo ip link set <interface> up`
5. Crear un virtual switch por medio de OpenvSwitch. Debe ingresar el siguiente comando `ovs-vsctl add-br <nombre switch>` y el comando `ovs-vsctl set_fail_mode <nombre switch> autonomous`.
6. Proceda a agregar las cinco interfaces físicas al virtual switch creado. Para esto debe ejecutar los siguientes comandos:

```
$ovs-vsctl add-port <nombre switch> wan
```

```
$ovs-vsctl add-port <nombre switch> lan0
```

```
$ovs-vsctl add-port <nombre switch> lan1
```

```
$ovs-vsctl add-port <nombre switch> lan2
```

```
$ovs-vsctl add-port <nombre switch> lan3
```

7. Para comprobar que la configuración ingresada se ha almacenado correctamente, lo puede realizar mediante el comando `ovs-vsctl show`.

Una vez que se han realizado los pasos anteriores en cada uno de los dispositivos BananaPi BPI-R2, proceda a testear que estén funcionando como un conmutador convencional. Una forma de comprobar que se ha realizado correctamente la configuración es conectando dos hosts BananaPi BPI-M3 (deben tener configuradas las IPs indicadas en la figura) a cada uno de los BananaPi BPI-R2 configurados y verifique que exista conectividad por medio del comando `ping`.

NOTAS:

La configuración que se ha realizado con uno de los conmutadores banana pi BPI-R2 se debe repetir en cada uno de los otros conmutadores. Para evitar que se convierta en una labor extensa, preparar un script con todos aquellos comandos que sean necesarios. Esto servirá también para la siguiente práctica que se va a realizar.

No efectúe conexiones entre los conmutadores antes de activar algún mecanismo para la prevención de bucles.

Troubleshooting

Existe la posibilidad de que se encuentren fallos con el entorno del escenario, enfocándose principalmente con las tarjetas SD. Existe una imagen sin modificaciones que se encuentra disponible en el Moodle de la asignatura. Si llegara a suscitarse un inconveniente como lo descrito anteriormente entonces formatee la SD, proceda a descargar la imagen y cópiela a la SD utilizando el siguiente comando

```
dd if=<image file> of=/dev/mmcbk0
```

TAREA 2: Pruebas con Spanning Tree Protocol

En esta sección de la práctica, se comprobará el protocolo STP como un mecanismo de asignación de recursos utilizado dentro de la topología leaf-spine. Con este mecanismo se puede comprobar que no se usan todos los enlaces establecidos entre los Spines y los Leafs sino que lo realiza por medio de uno de ellos.

Por defecto, al realizar las configuraciones básicas del OpenvSwitch en cada conmutador BananaPi BPI-R2, también se activa Spanning Tree. Esto se puede comprobar con el comando `sudo ovs-vsctl list interface`, en el parámetro `stp_enable` que debe tener valor `true`. En caso de que figure el valor de `false` entonces se procederá a ingresar el siguiente comando `sudo ovs-vsctl set bridge <nombre_bridge> stp_enable=true`.

Para montar la topología Leaf-Spine, es necesario que se seleccionen dos conmutadores, los cuales serán utilizados como conmutadores Spine, y los otros tres conmutadores restantes serán utilizados como conmutadores Leaf cumpliendo así con la arquitectura de lo que es un BigSwitch. La conexión a realizar debe cumplir con la siguiente regla: cada conmutador Spine debe estar conectado directamente con cada uno de los conmutadores Leaf. No se pueden realizar conexiones de forma directa entre conmutadores Spine ni entre conmutadores Leaf.

Para comprobar el funcionamiento del BigSwitch, se debe conectar un host (BananaPi BPI-M3) en el conmutador Leaf 1 y otro host en el en el conmutador Leaf 2 o Leaf 3, y luego realizar un ping entre ellos. Ambos hosts deben estar configurados en la misma subred.

Después de haber verificado que funciona correctamente nuestro BigSwitch, se procederá a obtener mediciones de las prestaciones que posee nuestra red. En este caso

vamos a hacer uso de la herramienta *iperf3*. En primer lugar, se conectarán los hosts según lo descrito en el esquema del escenario 1, es decir, conectaremos dos hosts en el conmutador Leaf 1 (a los que llamaremos H1 y H2), un host en el conmutador Leaf 2 (H3) y por último un host en el conmutador Leaf 3 (H4). A continuación, se procederá a realizar la siguiente comprobación:

- Configurar en modo servidor los host H2, H3 y H4 con el comando *iperf3 --server*.
- Configurar en modo cliente el host H1 y obtenga el ancho de banda del que se dispone en cada uno de los tres pares cliente-servidor:
 - *iperf3 --client <ip_H2>*
 - *iperf3 --client <ip_H3>*
 - *iperf3 --client <ip_H4>*

Anotar los resultados de este procedimiento en la siguiente tabla.

Par cliente-servidor	Ancho de banda
H1-H2	
H1-H3	
H1-H4	

Según los datos obtenidos, ¿considera usted que un BigSwitch funciona como si fuera un único switch transparente global?

NOTA: Todos los resultados van a ser similares independiente de las rutas que siga el tráfico por los diferentes conmutadores que configuran el “BigSwitch”

A continuación, proceder a medir la capacidad total de conmutación del BigSwitch. Para lograr este resultado, se deberán llenar los enlaces internos a su máxima capacidad. Se hará uso nuevamente de la herramienta *iperf3*, pero en esta ocasión se la utilizará en modo UDP. Las acciones a realizar son las siguientes:

- Configurar en modo servidor los host H3 y H4 con el comando *iperf3 --server*.
- Configurar en modo cliente y UDP los hosts H1 y H2, en donde se creará un flujo de datos que posea la misma velocidad que la obtenida en el procedimiento anterior. Esta acción se hará en cada par cliente-servidor con los siguientes comandos en cada uno de los clientes H1 y H2:
 - *iperf3 --client <ip_H3> --udp --bandwidth 1G --interval 1*
 - *iperf3 --client <ip_H4> --udp --bandwidth 1G --interval 1*

Anotar los resultados de este procedimiento en la siguiente tabla.

Par cliente-servidor	Ancho de banda
H1-H3	
H1-H4	
H2-H3	
H2-H4	

Según los resultados que se han obtenido en este procedimiento, ¿Cuál es la capacidad de conmutación real que posee el BigSwitch utilizando STP?

NOTA: El protocolo STP no permite distribuir los tráficos simultáneos en diferentes rutas.

Considerando que los puertos físicos de los conmutadores BananaPi son Giga-Ethernet, ¿Cuál es el porcentaje de capacidad de uso en cada uno de sus puertos por los que está circulando, entrando o saliendo, el tráfico con respecto a su capacidad total disponible?

La herramienta *iperf3* posee varias opciones útiles que nos permite saber las prestaciones con las que contamos. Para ver más información sobre esta herramienta y sus opciones, lo puede realizar a través del link <https://iperf.fr/iperf-doc.php>. También puede desplegar sus opciones mediante el comando *man iperf3*.

TAREA 3: Pruebas con Shortest Path Bridging

Básicamente, se harán las mismas actividades que se realizaron en la tarea 2, con la diferencia que ahora se utilizará el protocolo SPB en lugar del protocolo STP.

Para activar este protocolo en cada uno de los conmutadores virtuales creados con OpenvSwitch, se le ha proporcionado un fichero de configuración o script. Este script posee los comandos necesarios para efectuar este protocolo SPB y se encuentra en el directorio */root/spb*.

Realice el siguiente procedimiento para activar el protocolo SPB:

- Desactivar el protocolo STP en todos los conmutadores virtuales que se han creado en cada BananaPi BPI-R2, mediante el comando *sudo ovs-vsctl set bridge <nombre_bridge> stp_enable=false*
- Ejecutar el script proporcionado */root/spb/enable_spb.sh*

Inmediatamente comprobar que el BigSwitch continúa funcionando correctamente, conectando los hosts en los respectivos conmutadores y probando la conectividad entre ellos mediante la herramienta ping al igual que se hizo en la tarea anterior.

Una vez comprobada la conectividad, repita los procedimientos ejecutados en la tarea anterior con la herramienta iperf3 tanto en modo TCP como en modo UDP. Anote los resultados. En la primera de las pruebas ¿existe alguna mejora en la velocidad de conmutación en comparación a lo realizado con el protocolo STP? ¿Cuál es el motivo para que suceda?

NOTA: No debería de haber ningún cambio ya que no hay tráficos simultáneos

Respecto a la segunda de las pruebas ¿mejora el porcentaje de capacidad de uso en cada uno de sus puertos con respecto a su capacidad total disponible?

NOTA: El protocolo SPB permite utilizar simultáneamente todos los enlaces disponibles entre los conmutadores del "bigswitch"

4.3 Práctica 2: Comparación de algoritmos de asignación de recursos en un BigSwitch basado en SDN

Objetivos

Por medio de esta práctica se desea analizar en un BigSwitch con dispositivos reales, el funcionamiento y características de los algoritmos de asignación de recursos basados en SDN que se encuentran disponibles en este tipo de arquitecturas y se han estudiado en teoría.

**ES NECESARIO QUE ANTES DE EMPEZAR LAS PRÁCTICAS SE LEAN
DETENIDAMENTE LOS PASOS A REALIZAR**

Conocimientos necesarios:

- Haber realizado la Práctica 1: BigSwitch Leaf-Spine y pruebas de algoritmos.
- Operaciones básicas con OpenvSwitch junto con tablas de flujos OpenFlow.
- Funcionamiento básico de las redes definidas por software.
- Protocolos STP y SPB
- Topología Leaf-Spine.
- Herramientas que permitan realizar pruebas de conectividad, ping e iperf.

Descripción del escenario de la práctica

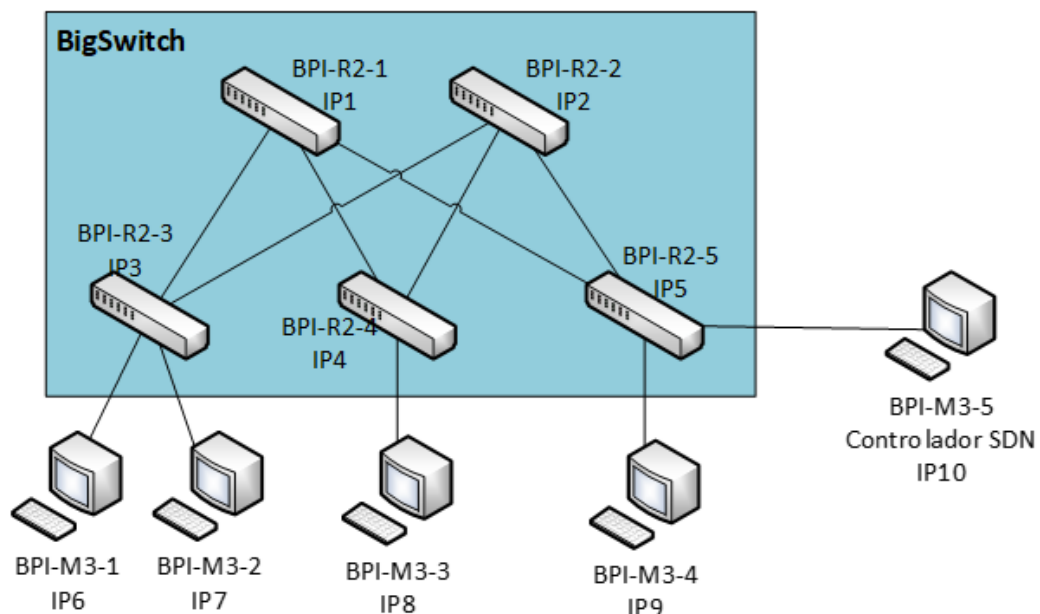


Figura 21 Esquema del escenario 2

Para la elaboración de esta práctica se tienen a disposición cinco placas BananaPi BPI-R2 por puesto de trabajo. En cada una de las placas se van a encontrar cinco puertos Giga-Ethernet, los cuales se van a poder utilizar como Switch por medio del software OpenvSwitch que ya se encuentra instalado en la placa, la cual también se la configurará como un conmutador SDN. Se procederá a realizar la construcción del BigSwitch utilizando la topología leaf-spine como se muestra en el esquema del escenario 2.

Adicionalmente se tienen cuatro placas BananaPi BPI-M3, las cuales cuentan con un sistema operativo Ubuntu MATE (versión original del fabricante SINOVOIP) y con las herramientas de test ping e iperf en funcionamiento.

Para completar el escenario, es necesario contar con un controlador SDN que se encuentre conectado al BigSwitch. Este se soporta en un dispositivo BananaPi BPI-M3 en el que se dispone del paquete software correspondiente a un controlador SDN OpenDaylight.

La tabla que se muestra a continuación posee las IPs asignadas a cada uno de los dispositivos de red.

IP1	192.168.1.11	Máscara /24
IP2	192.168.1.12	
IP3	192.168.1.13	
IP4	192.168.1.14	
IP5	192.168.1.15	
IP6	192.168.1.21	
IP7	192.168.1.22	
IP8	192.168.1.23	
IP9	192.168.1.24	
IP10	192.168.1.30	

TAREA 1: Preparación del controlador

Acceder al servidor en el que se realizará la instalación del controlador. Para obtener el paquete software a instalar en este dispositivo BananaPi BPI-M3 dirijase al directorio `~/odl` y siga los pasos a continuación.

1. Acceda vía ssh al servidor, por medio de la IP10 dada en la tabla de IPs. Las credenciales de acceso son: *user: pi, password: bananapi*.
2. Iniciar el controlador. Para esto se debe desplegar el fichero tar disponible en el directorio indicado anteriormente, cambiar al directorio que se crea como resultado del despliegue y ejecutar el comando: `./bin/karaf`.
3. Desde la línea de comandos que ofrece “karaf”, instalar el módulo “odl-dlux” mediante el comando `feature:install odl-dlux-all`
4. El módulo instalado permite al usuario tener una interfaz gráfica desde la que se puede ver la topología que se encuentra conectada al controlador. Para poder acceder a dicha interfaz, se debe abrir en el navegador de la máquina desde la que está accediendo al controlador SDN por ssh, la siguiente dirección <http://<IP10>:8181/index.html>. Las credenciales de acceso son admin/admin.
5. Mantener abierta la conexión “ssh” que está dando acceso a la consola de “karaf” que se utilizó en el paso 3.

Troubleshooting

El controlador ODL va acumulando las “features” que se instalan en el controlador ODL, sean estas correctas, o no. Para obtener un controlador en situación “inicial”; es decir, sin ninguna “feature” instalada, basta con borrar completamente la carpeta donde se ha desplegado el fichero “tar” que contiene el software del controlador ODL, con el comando `rm -r <directorio_instalación>`, y volver a desplegar dicho fichero.

TAREA 2: Preparación del BigSwitch

Se debe comenzar configurando cada uno de los dispositivos BananaPi BPI-R2 como conmutador SDN. Para realizar esta acción, se va a hacer uso del script que se solicitó crear al final de la Práctica 1 - Tarea 2 en el que habrá que cambiar la opción "autonomous" en "*ovs-vsctl set_fail_mode <nombre switch> autonomous*" por "secure"

Adicionalmente, se agregará una nueva línea de comando que permitirá añadir un manager OVS a la configuración actual. El comando que se agrega al final del script es el siguiente:

```
ovs-vsctl set manager:tcp:<IP10>:6653
```

Ahora se procede a configurar cada uno de los conmutadores SDN (BPI R2) con el script generado. Una vez que se ha realizado la configuración, en cada uno de ellos conecte dos hosts y efectúe la comprobación de que exista la conectividad necesaria. Esto sin conectar aún el cable hacia el controlador.

Después de haber comprobado que todos los conmutadores operan con normalidad individualmente, se puede montar la topología leaf-spine y por último conectar el controlador SDN al BigSwitch, siguiendo la figura del escenario 2.

Comprobar, mediante el navegador y la URL usadas anteriormente (<http://<IP10>:8181/index.html>) que se refleje la topología que se ha montado.

TAREA 3: Análisis de la topología utilizando SDN-STP

OpenDaylight es un controlador SDN que posee un gran número de funcionalidades y herramientas. En este caso se hará uso de un módulo en específico, llamado *odl-l2switch*, que contiene la función de eliminación de los bucles dentro de la red. Esta acción podría llegar a confundirse con la función de STP, pero no es así, porque no se está trabajando bajo un entorno Ethernet tradicional. Sin embargo, el resultado final es equivalente.

A continuación, se detallan las instrucciones que van a permitir instalar el módulo y probar su funcionamiento.

1. Acceder a la consola “karaf” preparada anteriormente y que se está ejecutando en el controlador OpenDaylight.
2. Instalar el módulo *odl-l2switch* mediante el comando *feature:install*.
3. No cerrar la consola de “karaf” durante la realización de la práctica.

Ahora ya se tiene un algoritmo para manejar la red, el cual es el equivalente en SDN a lo que realiza STP, lo que quiere decir que ya se habrán eliminado los bucles bloqueando diferentes enlaces internos en el conmutador Spine-Leaf (Bigswitch).

Conectar dos hosts BananaPi BPI-M3 (deben tener configuradas IPs de la misma subred) a un BananaPi BPI-R2 con cables Ethernet y verificar que exista conectividad por medio del comando ping. Ahora se puede comprobar que en la interfaz gráfica del controlador la visualización de la topología con los dos hosts agregados.

Repita todos los pasos de la TAREA 2 del escenario 1.

NOTA: Deberá obtener exactamente los mismos resultados que en el caso anterior.

TAREA 4: Análisis de la topología utilizando SDN-SPB

Básicamente, se harán las mismas actividades que se realizaron en la tarea 3, con la diferencia de que ahora se va a utilizar el algoritmo SDN-SPB en vez del SDN-STP.

El algoritmo SDN-SPB está enfocado como una aplicación SDN que ha sido diseñado por un tercero y está ubicado dentro de la máquina M3 en uso, en el directorio *~/algoritmosSDN/SPB*. Previamente examine el contenido de este directorio y siga las instrucciones que ha proporcionado el desarrollador para realizar la instalación de la aplicación en OpenDaylight. Este paquete está basado en los módulos de redes virtuales (VTN) de ODL.

Una vez que se haya completado la instalación, repita todos los pasos de la TAREA 3 del escenario 1.

NOTA: Deberá obtener exactamente los mismos resultados que en el caso anterior.

4.4 Práctica 3: Topología dependiente del tráfico

Objetivos

Por medio de esta práctica se desea analizar que bajo diferentes circunstancias a las presentadas en las prácticas anteriores se puede controlar el flujo que pasa por una red. En donde se hará uso de nuevos módulos basados en SDN que se encuentran disponibles para el tipo de arquitectura descrito posteriormente. Se modifica la topología físicamente para de esta manera obtener como resultado una topología más simple, con el objetivo de comparar las prestaciones entre ellos.

**ES NECESARIO QUE ANTES DE EMPEZAR LAS PRÁCTICAS SE LEAN
DETENIDAMENTE LOS PASOS A REALIZAR**

Conocimientos necesarios:

- Haber realizado las Prácticas 1 y 2.
- Operaciones básicas con OpenvSwitch junto con tablas de flujos OpenFlow.
- Funcionamiento básico de las redes definidas por software.
- Demostrar el control del flujo de tráfico en la red
- Herramientas que permitan realizar pruebas de conectividad, ping e iperf.

Descripción del escenario de la práctica

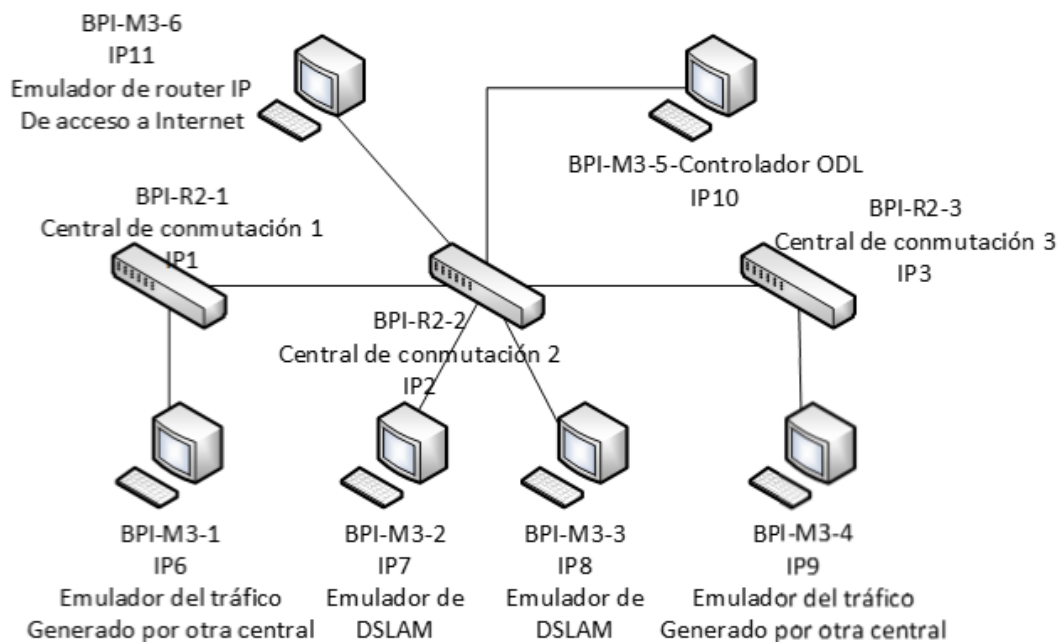


Figura 22 Esquema del escenario 3

En este caso se abarca un ejemplo más orientado a lo que se tiene en el campo real. Supongamos que se tiene a un proveedor de servicios (ISP); el cual cuenta con una cantidad numerosa de clientes y quienes se encuentran conectados mediante la tecnología ADSL. Debido a la cantidad de clientes, es necesario que se tengan mucho más que una central de conmutación, para nuestro ejercicio vamos a considerar que son tres.

Una de las centrales de conmutación se encuentra directamente conectada al enlace con el Router IP, mediante el cual se obtiene el acceso a Internet. Las demás centrales de conmutación enviarían su tráfico a través de la primera central para lograr tener acceso a Internet.

Cada uno de los clientes con su ADSL llegan a la central de conmutación gracias a la intervención de un DSLAM, el cual cuenta con una potente capacidad de procesamiento de ancho de banda. El DSLAM sirve como un multiplexor de las conexiones ADSL hacia la central de conmutación. El tráfico que cursa esta red en su mayoría es web multimedia, donde se transfiere tanto audio como video.

La arquitectura que se ha establecido para el escenario se encuentra en la Figura del escenario, la cual permite manejar este tipo de tráfico adecuadamente sin la necesidad de utilizar la topología Leaf-Spine vista en las prácticas anteriores. En este caso no se necesita hacer un balanceo de carga, como si lo hace Leaf-Spine; porque gracias a la ingeniería de tráfico, se conoce previamente cuál es el camino que cursan los paquetes.

Para la elaboración de esta práctica se tienen a disposición tres placas BananaPi BPI-R2 por puesto de trabajo. En cada una de las placas se van a encontrar cinco puertos Giga-Ethernet, los cuales se van a poder utilizar como Switch por medio del software OpenvSwitch que ya se encuentra integrado en la placa, la cual también se la configurará como un conmutador SDN.

La tabla que se muestra a continuación, posee las IPs asignadas, las cuales servirán para configurar los dispositivos de red.

IP1	192.168.1.11	Máscara /24
IP2	192.168.1.12	
IP3	192.168.1.13	
IP4	192.168.1.14	
IP5	192.168.1.15	
IP6	192.168.1.21	
IP7	192.168.1.22	
IP8	192.168.1.23	
IP9	192.168.1.24	
IP10	192.168.1.30	
IP11	192.168.1.1	

TAREA 1: Preparación del controlador

Realice las instrucciones que se encuentran en el manual de la Práctica 2 – Tarea 1 para lograr ejecutar el controlador SDN en el dispositivo BananaPi BPI-M3.

TAREA 2: Preparación de la topología

Para que se puedan realizar las pruebas con la topología presentada en la Figura del escenario, primeramente, se deben iniciar los conmutadores y conectarlos de la manera que se señala. Aquella interconexión representa parte de la topología que utiliza un proveedor de servicios. En el campo real, el medio que se utiliza para las conexiones entre los conmutadores (centrales de conmutación) deben ser por fibra óptica de 100Gbps, pero en esta práctica se tienen medios que permiten hasta 1Gbps. A continuación, se indican los pasos a realizar para configurar uno de los conmutadores.

Repita los pasos de la TAREA 2 del escenario 2 teniendo en cuenta la nueva topología descrita en la figura del escenario 3.

TAREA 3: Generación del tráfico y comprobación de conexión

El escenario de la práctica está constituido por simuladores de tráfico, centrales de conmutación y emulador de router IP que permite la salida a Internet. Donde para verificar el funcionamiento básico de la topología, se deben realizar mediciones y comprobar que el tráfico generado siga un encaminamiento específico en la red.

Repita los pasos de la TAREA 4 del escenario 2 (análisis del SPB) teniendo en cuenta la nueva topología descrita en la figura del escenario 3, además considerando la siguiente asignación de equipos de usuario (BPI-M3):

H1	H2	H3	H4
BPI-M3-1	BPI-M3-2	BPI-M3-6	BPI-M3-4

NOTA:

En este caso, para la prueba de conectividad, los resultados deben ser similares a los de las prácticas anteriores, ya que la infraestructura hardware es similar.

En el segundo caso, prueba de porcentaje de ocupación de los puertos, por la arquitectura de la red, el enlace por la BPI-M3-6 es un enlace compartido por todos los accesos a Internet.

Compruebe que del lado del emulador de router IP el tráfico cursado es la suma de los tráficos de acceso a Internet.

Capítulo 5

5 Conclusiones y Trabajos Futuros

5.1 Conclusiones

A continuación, se tiene un resumen de las conclusiones consideradas como las más significativas gracias a la elaboración de este trabajo.

- Conocer que un BigSwitch viene dado por un conjunto de conmutadores conectados entre sí para poder alcanzar entre las más altas prestaciones fue primordial para saber a qué punto se debía llegar y poder transmitir ese conocimiento a través del diseño.
- Diseñar un BigSwitch con dispositivos BananaPi BPI-R2, los cuales son de bajas prestaciones con respecto a lo que tiene en el campo real, ha llegado a ser factible y accesible. De esta manera también se han podido aprovechar los recursos para incluirlo en el estudio de las Redes Definidas por Software.
- El uso de las Redes Definidas por Software ofrece la posibilidad de optimizar el uso de los recursos y de esta manera poder ofrecer un alto rendimiento a la red.
- Manejar correctamente las versiones del software utilizado llegó a ser fundamental al momento de poner en marcha el diseño establecido, ya que existían una gran variedad de conflictos por incompatibilidades entre ellos.
- En la parte práctica se ha conseguido que se haga openflow solo de aquellos paquetes que son difusivos debido a que, en la actualidad no se encuentra correctamente integrado el OpenvSwitch con el DSA.

5.2 Propuestas de trabajos futuros

El trabajo que se ha realizado puede tener ciertas ampliaciones, las cuales ayudarán a abarcar más temas vistos en la teoría.

Se presenta la posibilidad de incluir prácticas que consten de un IDS/IPS para agregar políticas de seguridad a la red y de esta forma establecer las diferencias en cuanto a lo que ocurre mientras se está midiendo la capacidad de conmutación de los dispositivos.

Una posible variante también sería la de conseguir que se realice openflow de los paquetes que no solo sean difusivos o multicast sino que también se lo consiga con los paquetes direccionados o unicast. Consistiría en lograr integrar correctamente el OpenvSwitch con el driver DSA de la arquitectura del BananaPi BPI-R2, sin tener que esperar a las correspondientes actualizaciones de los paquetes oficiales.

Otras de las modificaciones que se pueden desarrollar es la de utilizar el programa Jperf al momento de realizar las pruebas. El cual integra las funcionalidades del *iperf2* e *iperf3* pero posee una interfaz gráfica en Java. Aquella interfaz permite modificar los parámetros que se requieran para realizar diversas pruebas desde varias perspectivas. Se presentan gráficos, que ayudan a obtener estadísticas de las prestaciones con las que se cuenta en la red.

6 Bibliografía

- [1] W. Stallings, *Foundation of Modern Networking. SDN, NFV, QoE, IoT, and Cloud*, Indianápolis, Indiana: Pearson Education, 2016.
- [2] O. D. C. Alliance, «Master USAGE MODEL: Software-Defined Networking,» 2015. [En línea]. Available: <https://joinup.ec.europa.eu/solution/open-data-center-alliance-master-usage-model-software-defined-networking-rev-10-open-data/about>. [Último acceso: 11 Junio 2018].
- [3] Y. Wen, W. Xia, C. Heng Foh, D. Niyato y H. Xie, «Nanyang Technological University,» Marzo 2015. [En línea]. Available: *A Survey on Software-Defined Networking*. [Último acceso: 11 Junio 2015].
- [4] Open Networking Foundation, «Software-Defined Networking: The New Norm for Networks,» 13 Abril 2013. [En línea]. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>. [Último acceso: 12 Junio 2018].
- [5] SDxCentral, «What are SDN Southbound APIs?,» [En línea]. Available: <https://www.sdxcentral.com/sdn/definitions/southbound-interface-api/>. [Último acceso: 14 Junio 2018].
- [6] SDxCentral, «What are SDN Northbound APIs (and SDN Rest APIs)?,» [En línea]. Available: <https://www.sdxcentral.com/sdn/definitions/northbound-interfaces-api/>. [Último acceso: 14 Junio 2018].
- [7] OpenDaylight, «OpenDaylight Controller:Overview,» [En línea]. Available: https://wiki.opendaylight.org/view/OpenDaylight_Controller:Overview. [Último acceso: 17 Junio 2018].
- [8] A. Abdul-hafiz, E. Adewale Adedokun y S. Man-Yahya, «Improved Extended Dijkstra's Algorithm for Software Defined Networks,» [En línea]. Available: https://www.researchgate.net/publication/320960597_Improved_Extended_Dijkstra's_Algorithm_for_Software_Defined_Networks.

[Último acceso: 17 Junio 2018].

- [9] SUNSET LEARNING INSTITUTE, «SOFTWARE-DEFINED NETWORKING (SDN) WHAT IS IT AND HOW DOES IT WORK,» [En línea]. Available: <https://www.sunsetlearning.com/wp-content/uploads/2014/09/SDN-Software-Defined-Networking.pdf>. [Último acceso: 15 Junio 2018].
- [10] A. Lunn, V. Didelot y F. Fainelli, «Distributed Switch Architecture, A.K.A. DSA,» [En línea]. Available: <https://netdevconf.org/2.1/papers/distributed-switch-architecture.pdf>. [Último acceso: 25 Junio 2018].
- [11] Linux Kernel Organization, «Distributed Switch Architecture,» [En línea]. Available: <https://www.kernel.org/doc/Documentation/networking/dsa/dsa.txt>. [Último acceso: 20 Junio 2018].
- [12] Academia Cisco, «Fundamentos de enrutamiento y conmutación (Routing and Switching Essentials) CCNA2 V5,» Marzo 2015. [En línea]. Available: https://julioestrepo.files.wordpress.com/2015/03/pdf_ccna2_v5.pdf. [Último acceso: 6 Junio 2018].
- [13] W. Nelson, «Introduction to Spine-Leaf Networking Designs,» 7 Noviembre 2017. [En línea]. Available: <https://lenovopress.com/lp0573-introduction-to-spine-leaf-networking-designs>. [Último acceso: Junio 2018].
- [14] Sinovoip, «Banana Pi BPI-R2,» 8 Mayo 2018. [En línea]. Available: http://wiki.banana-pi.org/index.php?title=Banana_Pi_BPI-R2&action=history. [Último acceso: 20 Junio 2018].
- [15] Snovoip, «Banana Pi BPI-M3,» 10 Mayo 2018. [En línea]. Available: http://wiki.banana-pi.org/Banana_Pi_BPI-M3#Documents. [Último acceso: 15 Junio 2018].
- [16] B. Pfaf, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon y M. Casado, «The Design and Implementation of Open vSwitch,» Mayo 2015. [En línea]. Available: <https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-pfaff.pdf>. [Último acceso: 4 Julio 2018].
- [17] The Linux Foundation, «Open vSwitch Release notes,» 19 Octubre 2018. [En

- línea]. Available: <https://www.openvswitch.org/releases/NEWS-2.10.1.txt>. [Último acceso: 14 Noviembre 2018].
- [18] The Linux Foundation, «Production Quality, Multilayer Open Virtual Switch,» 2016. [En línea]. Available: <https://www.openvswitch.org/>. [Último acceso: Julio 2018].
- [19] Sinovoip, «How to build an Ubuntu/Debian SD image from scratch,» 1 Septiembre 2018. [En línea]. Available: <http://forum.banana-pi.org/t/how-to-build-an-ubuntu-debian-sd-image-from-scratch/6805>. [Último acceso: 12 Diciembre 2018].
- [20] The Linux Foundatrion, «How to Port Open vSwitch to New Software or Hardware,» 2016. [En línea]. Available: <http://www.openvswitch.org/support/dist-docs-2.5/PORTING.md.html>. [Último acceso: Diciembre 2018].