Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

ESTUDIO DEL ESTANDAR OPENFLOW MEDIANTE CASO PRÁCTICO DE UTILIZACIÓN CON LA HERRAMIENTA VNX.

TRABAJO FIN DE MÁSTER

Vanessa Alejandra Moreno Galué

2012

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

ESTUDIO DEL ESTANDAR OPENFLOW MEDIANTE CASO PRÁCTICO DE UTILIZACIÓN CON LA HERRAMIENTA VNX.

Autor:
Vanessa Alejandra Moreno Galué

Director:
David Fernández Cambronero.

Departamento de Ingeniería de Sistemas Telemáticos

2012

Dedicatoria:

A mis Padres José Luis y Silvana por su apoyo constante e incondicional durante todos mis estudios, son la luz de mi vida.

Agradecimientos:

Me gustaría empezar este capítulo de agradecimientos con David Fernández Cambronero, director de este Trabajo, el cual depositó su confianza en mí, haciendo todo lo que estaba en su mano para que el mismo pudiera ser realizado. También debo destacar su gran valía científica y su criterio, que han permitido que el contenido y tema de este trabajo alcanzaran un considerable estatus de originalidad y calidad.

También deseo agradecer al personal e infraestructura de la Universidad Politécnica de Madrid que con sus numerosos libros, equipos y distintas herramientas de aprendizaje facilitan que todos y cada uno de sus alumnos pueda aprender y trabajar sin ningún tipo de limitaciones.

Por último a todos mis profesores del Máster de Ingeniería de Redes y Servicios Telemáticos quienes con sus diferentes backgrounds aportan una mezcla interesante de teoría y práctica que hace que el curso tome un valor agregado adicional.

Resumen:

El objetivo fundamental de este proyecto consiste en estudiar el estado de arte de las tecnologías de virtualización y la capacidad de poder llevar a cabo una implantación de la misma a pequeña escala y de esta forma dar veracidad a que este concepto tan en boga facilita de forma ágil a que los usuarios tengan gestión y administración de las redes sin necesariamente poseer un exhaustivo conocimiento de los recursos correspondientes a los escenarios virtualizados. Para ello, se ha llevado a cabo una exhaustiva búsqueda de información en la web relacionada tanto con el estado del arte de las redes actuales como del protocolo OpenFlow.

Primero que nada es importante recalcar que la virtualización es la noción que computa recursos que pueden estar disponibles con un gran nivel de rapidez y flexibilidad, sin requerir que los usuarios tengan conocimiento detallado de recursos físico-disponibles. La importancia de la virtualización radica en compartir los recursos de una manera aislada en ambientes contenidos en la red (por ejemplo: la tecnología cloud), lo que permite maximizar la utilización de los recursos. Con respecto a dicha maximización se debe buscar que cada recurso se comporte como una serie de pequeños recursos, y que al compartir los mismos sea posible alcanzar el máximo porcentaje de su utilización.

En el estudio de la tecnología del protocolo OpenFlow, en primer lugar, se ha buscado información sobre sus características, funcionalidad, arquitectura, plataformas de desarrollo, etc. que ha permitido conocer en detalle todo lo relacionado con dicha tecnología. Posteriormente, se ha analizado el rol que tiene activamente la misma y toda la vanguardia tecnológica que involucra su aplicación no solo como una opción de fácil implementación para la pequeña y mediana empresa, sino como modelo ejemplar de los nuevos esquemas tecnológicos.

Además, dado el gran interés investigador que existe en la actualidad sobre el protocolo en cuestión, se ha estudiado un escenario planteado con un grupo de pruebas adjuntas, de manera que sea posible apreciar la aplicabilidad del mismo a través de herramientas de libre instalación como son openvswitch y VNX.

El presente trabajo está estructurado por una breve introducción que condensa la información más relevante del protocolo OpenFlow, desde sus inicios, predecesores, situación actual, hasta llegar a un tutorial que expone sus funcionalidades básicas y que posteriormente será adaptado por el autor con las herramientas ya mencionadas anteriormente; todo esto en un entorno domestico, lo cual conllevará a una serie de conclusiones y observaciones técnicas.

Una vez realizado este nexo de unión entre el estado de arte de la tecnología y la aplicación del protocolo, se da un paso más allá, elaborando un análisis donde se

compararán ambos casos prácticos, el planteado por el tutorial de la herramienta y el realizado dentro de este trabajo para verificar fortalezas y debilidades y poder emitir una opinión de utilidad y veracidad de cada uno de ellos.

Abstract:

The objective of this project is to study the state of the art of virtualization technologies and the ability to carry out a deployment of this technology on a small scale and thus give credence that this so popular concept, makes network management and control an easy process for users without necessarily having a thorough knowledge of the resources for virtualized scenarios. For this purpose, It has been conducted an exhaustive search for information on the web related to both the current state of the art about existing networks protocols and the specifically the OpenFlow protocol.

First, it is important to establish that virtualization is the notion in which computing resources are available with a high level of speed and flexibility, without requiring users to have detailed knowledge of the physical resources available. The importance of virtualization is to share in isolated environments contained in a network (eg. cloud computing technology), this way allowing maximum use of the resources.

In the study of technology OpenFlow protocol firstly, information has been on its features, functionality, architecture, development platforms, etc. which has allowed knowing in detail everything about this technology. Subsequently, we analyzed the active role that its cutting edge of technology involved has not just as an application easy to implement for small and medium enterprises, but as an exemplary model of new global technological schemes

Moreover, given the great research interest that currently exists on the protocol in question, it has been analyzed a scenario composed by different groups of tests, so that it is possible to appreciate the applicability of it around different environments.

This work is structured by a brief introduction that condenses the most relevant information about the OpenFlow Protocol since its beginning, predecessors, current state, up to implementation of its basic features in a home environment which will lead to a series of conclusions and technical comments. Once the link between the state of the art technology and the implementation of the protocol gets established, the present work takes a step forward and it is developed a detailed analysis of future applications envisioned for this protocol and its possible projections.

Índice general

Índice de Figuras:

Índice de Tablas:

1 Introducción:

La virtualización ha cambiado la manera como se hace computación en la actualidad, por ejemplo muchos datacenters hoy por hoy están totalmente virtualizados de manera que puedan proveer de forma ágil toda la información contenida en la red y en caso de "desastres" informáticos relacionados con pérdida de información poder reaccionar muy rápidamente hacia el proceso de recuperación. Alineado con esta llamativa ventaja está el hecho de que la masificación de esta tecnología no parece dar marcha atrás y por el contrario un estudio de Gartner indica que mientras el 12% de los x86 están virtualizados actualmente, ese número crecerá hasta un 61% para el 2013 [1].

En concreto, los objetivos que se han pretendido cubrir con este TFM han sido los siguientes:

- Conocer y especificar las características técnicas e históricas de la virtualización de redes así como predecesores y causas de la aparición del protocolo OpenFlow.
- Obtener una visión global del estado del arte actual de la tecnología correspondiente a dicho protocolo.
- Estudiar las diferentes aplicaciones para la tecnología en cuestión y los principales grupos de trabajo.
- Implantar un escenario con OpenFlow, donde se transformará un controlador hub en un switch controlador con capacidad de aprendizaje dinámico.
- Evaluar las tablas de flujo, mensajes y la funcionalidad de la prueba en cuestión.
- Describir brevemente mediante un análisis técnico las ventajas y bondades que aporta el protocolo así como también la simplificación de su uso mediante una simple integración con VNX.

La mayor parte de este TFM recoge un estado del arte tanto del protocolo OpenFlow como de los requerimientos. Para ello, ha sido imprescindible conocer bien a fondo cuales son los pilares o fundamentos sobre los que se basa esta tecnología y qué estándares han aportado activamente para la consolidación de este protocolo,

Una vez conseguido un nivel de conocimiento suficiente para poder abordar el tema, se procedió a investigar las herramientas presentes en estos entornos y las posibles aplicaciones de los mismos a pequeña escala, dado que verificar la implantación de estas tecnologías constituía unas de las principales motivaciones detrás de este trabajo.

El plan de trabajo seguido se puede observar en la Tabla I, dónde se muestra las diferentes actividades realizadas y los meses en los que se han estado trabajando en ellas.

Tabla 1: Esquema de las Tareas del TFM

Nombre de tarea	Duración	Comienzo	Fin
Inicio TFM	**172 días**	**mar 15/11/11**	**mié 11/07/12**
Estudio de Tecnologías de virtualización	25 días	mar 15/11/11	lun 19/12/11
Revisión Bibliografía Open Flow	30 días	mar 17/01/12	lun 27/02/12
Evaluación Requerimientos Técnicos	40 días	mar 28/02/12	lun 23/04/12
Preparación de SW para Pruebas	**13 días**	**mar 24/04/12**	**jue 10/05/12**
Instalación Sistema Operativo	13 días	mar 24/04/12	jue 10/05/12
Instalación Pack para Implantación	13 días	mar 24/04/12	jue 10/05/12
Realización de Pruebas (Trial)	25 días	vie 18/05/12	jue 21/06/12
Elaboración Informe Técnico	8,6 mss	mar 15/11/11	mié 11/07/12
Conclusiones	10 días	vie 22/06/12	jue 05/07/12
Fin TFM	0 días	jue 12/07/12	jue 12/07/12

En la siguiente figura, se puede observar el Diagrama de Gantt tareas mencionadas anteriormente con respecto al tiempo en el que se han realizado.

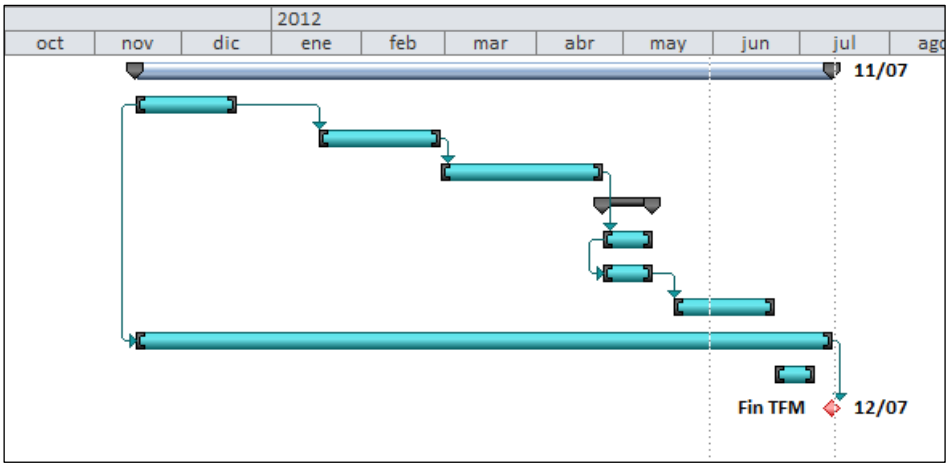

Figura 1 : Tareas del TFM en Diagrama de Gantt

2 Estado del arte:

Este apartado del estado del arte se ha dividido en dos partes muy diferenciadas. En primer lugar, se expondrá en qué estado se encuentran las diferentes tecnologías de red utilizadas y desplegadas en

relación a la virtualización de redes punto que contiene el primer subapartado de este estado del arte. Por otro lado, en el segundo bloque compuesto por varios subapartados, donde se describirá en qué consiste el protocolo OpenFlow, qué ventajas aporta, aplicaciones y despliegues de servicios actuales; herramientas asociadas a su implantación así como detalles de la prueba a realizar.

2.1 Virtualización de Redes.

2.1.1 Origen de la necesidad de redes programables:

Las Redes se han convertido en un elemento crítico de la infraestructura de los negocios, hogares y escuelas de la actualidad, este éxito puede considerarse tanto una bendición como una complicación para los investigadores de este campo; por un lado su trabajo se vuelve más relevante e imprescindible pero al mismo tiempo la capacidad de tener un impacto notable se hace cada vez más complicada. La reducción de impacto en este campo está relacionada a la enorme masificación de la tecnología mediante equipos, protocolos y canales en prácticamente cada entorno productivo, lo que le ha hecho lo suficiente globalizada y estandarizada como para hacer práctico y/o viable la introducción de nuevas ideas. Actualmente no es realmente factible "experimentar" con nuevos protocoles de red, que sean capaces de generar la suficiente confianza y respaldo como para poder expandirse como alternativas a los ya existentes operativos; esto ha traído como resultado que la mayor parte de nuevas ideas en el campo de networking sean meramente experimentales y muy pocas realmente probadas y verificadas, de este hecho surge la expresión de que la infraestructura de red está "osificada". [1]

Al reconocer el problema, la comunidad de networking trabaja arduamente para desarrollar redes programables, es gracias a esas iniciativas que nace uno de los proyectos pioneros en el ámbito de redes virtuales como es GENI [3] una estructura virtual de gran envergadura para experimentar nuevas arquitecturas y sistemas distribuidos. Estas redes programables requieren switches y otros equipos de igual capacidad programable (mediante virtualización) ya que es necesario sean capaces de procesar paquetes de múltiples redes aisladas de forma simultánea.

Por ejemplo en GENI está previsto que a un investigador se le asignen un porcentaje de recursos a través de toda la red, un porcentaje de enlaces, de elementos de procesamiento de paquetes (Ejemplo: Routers) y usuarios finales; cada programador dentro de esta red utiliza sus recursos asignados como desee, de manera que de forma experimental se pueda probar una configuración nueva en redes de universidades, laboratorios industriales, redes de sensores y redes inalámbricas, por mencionar algunos escenarios.

Las redes virtuales programables podrían disminuir la barrera para la entrada de enfoques novedosos, aumentando la tasa de innovación en lo que se refiere a infraestructura, sin embargo hay que tener en cuenta que los planes para infraestructuras nacionales pueden resulta bastante costosos y pueden tomar años para desarrollarse, es partir de estas premisas que surge la necesidad de que exista una plataforma con el soporte virtual para las nuevas propuestas.

El enfoque que no se toma en cuenta (pero sería el más atractivo) es "persuadir" al sector comercial, quién no suele proveer plataformas abiertas y/o programables, de manera que sea factible como mencionamos anteriormente probar nuevos protocolos y generar comodidad y tranquilidad entre administradores de red, cabe destacar este enfoque es poco factible a corto plazo, dado que no está en la naturaleza de los equipos actualmente comerciales la capacidad de ser abiertos o adaptables.

La práctica del networking comercial está basada en que las interfaces externas son estrechas y toda la flexibilidad del switch es interna, esta componente interna varía según el vendedor y no brinda una plataforma apta para probar alternativas, esta protección de su estructura interna está alineada para evitar que sus delicadas arquitecturas y algoritmos puedan verse alterados por pruebas externas y por supuesto que alguna mejora rompa su cota de mercado. Entre las iniciativas existentes puede mencionarse un router programable ATCA, basado en virtualización llamado "Supercharged Planetlab Platform"; actualmente está siendo estudiando y desarrollado por la Universidad de Washington, puede utilizar procesadores de red para trabajar paquetes de muchas interfaces con velocidad de líneas simultáneas. Este enfoque puede tener un futuro viable más sin embargo para el modelo actual es muy costoso para llegar a ser desplegado de forma masiva.

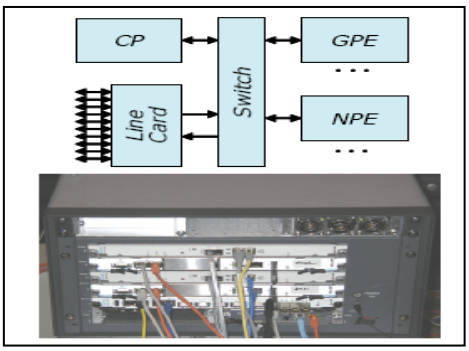

Figura 2: Organización del Sistema con el CP, GPE y NPE

Otra iniciativa es NetFPGA [4], una PCI de bajo coste con un FPGA de usuario programable capaz de procesar paquetes y cuatro puertos Ethernet Gigabit. NetFPGA está limitado a solo estas 4 interfaces, lo

cual es insuficiente para un armario de cableado, así que no vislumbra como una opción viable para una alternativa comercial.

Una visión factible sería buscar un switch que pudiese garantizar una flexibilidad que sea:

- Fácil de someter a implantaciones de bajo coste
- Capaz de soportar un amplio rango de investigaciones y pruebas
- Capaz de garantizar el aislamiento de tráfico experimental del tráfico de producción
- Consistente con la necesidad de los vendedores de plataformas cerradas.

2.1.2 Maquinas Virtuales y Virtualización:

2.1.2.1 Definición de Maquina Virtual:

Una máquina virtual (VM) [5] es un contenedor de software aislado que puede ejecutar su propio sistema operativo y aplicaciones como si fuera una computadora física. Para ejercer cierto control y administración se requiere de un "Administrador de máquinas virtuales", pueden verse también como el enlace entre el Gateway y los recursos [6], el Gateway no comparte recursos físicos directamente, pero depende de la tecnología de virtualización para abstraerlos.

Asimismo la Maquina virtual depende del "Motor de infraestructura virtual" (VIE) para administrar máquinas virtuales en un conjunto de recursos físicos. Generalmente los VIE son capaces de crear y detener máquinas virtuales en un clúster físico.

2.1.2.2 Virtualización:

La virtualización habilita un gran número de posibilidades, los Datacenters virtualizados dinámicos son un ejemplo, donde los servidores proveen un conjunto de recursos que son aprovechados como sean necesarios según la demanda y que las aplicaciones, almacenamiento y red cambien dinámicamente mientras se conoce la carga de trabajo y las demandas del negocio.

La virtualización de los servidores solamente es el inicio, las virtualizaciones se deben extender al almacenamiento y a las redes en general.

2.1.2.3 Virtualización de redes:

La virtualización de redes se realiza por medio de switches de virtualización [7], se divide el ancho de banda disponible en canales seguros, esto permite crear zonas seguras internas y consolidar la seguridad externa.

La virtualización de redes habilita la movilidad de aplicaciones y datos, no solo a través de los servidores sino también en las redes y en los Datacenters, también se debe tomar en cuenta la optimización de dicha virtualización para garantizar el alto rendimiento del mismo, es decir, configurar de manera optima los administradores de máquinas virtuales para maximizar la utilización de recursos.

También permite que los recursos de red estén disponibles como segmentos virtuales [5], con dispositivos o porciones de recursos como el repositorio de almacenamiento que es accedido como se necesite, independiente de su ubicación física o conexión física de la red.

2.2 OpenFlow:

2.2.1 Introducción al Protocolo:

La idea base es simple, se explota el hecho de que la mayor parte de los switches y routers Ethernet contienen tablas de flujo (típicamente construidas de TCAMs) que corren en línea para implementar firewalls, NATs, QoS (Quality of Service) y para recolectar estadísticas, mientras que la tabla de cada vendedor sea diferente, hemos identificado un conjunto interesante de funciones que corren en muchos switches y routers, el objetivo principal de OpenFlow es explotar estas funcionalidades comunes.

A pesar que la virtualización tiene un rol activo en la actualidad, cuando se comparan los diversos sistemas abiertos con los switches de diferentes casas podemos notar que al final del día existe un "gap" (hueco) entre los atributos existentes (ver Figura Nº 3) y los deseados por nuestra parte de cada una de estas tecnologías y por la necesidad de poder abarcar la mayor cantidad de propiedades posibles y de allí que el novedoso protocolo OpenFlow

	Performance Fidelity	Scale	Real User Traffic?	Complexity	Open
Simulation	medium	medium	no	medium	yes
Emulation	medium	low	no	medium	yes
Software Switches	poor	low	yes	medium	yes
NetFPGA	high	low	yes	high	yes
Network Processors	high	medium	yes	high	yes
Vendor Switches	**high**	**high**	**yes**	**low**	**no**

Figura 3: Comparación de Sistemas Abiertos vs. Switches Comerciales

OpenFlow dota básicamente de un protocolo abierto para programar la tabla de flujo en diferentes switches y routers, un administrador de red puede dividir el tráfico en flujos de producción e investigación. Los investigadores controlan sus propios flujos al escoger las rutas que sus paquetes siguen y el procesamiento que reciben; en este sentido los investigadores pueden probar y experimentar con nuevos protocolos, nuevos modelos de seguridad, esquemas de direccionamiento e incluso alternativas a la tradicional IP (Ejemplo: HIP [9]); mientras en la misma red el tráfico de producción es aislado y procesado de forma tradicional y estándar.

2.2.2 OpenFlow Switch:

El camino de datos (datapath) de un switch OpenFlow consiste de una tabla de flujo y una acción asociada con cada entrada de flujo. El conjunto de acciones soportadas por un switch OpenFlow es extensible pero se establecen unos requisitos mínimos para todos los switches. Para que sea posible la implementación del protocolo OpenFlow debemos partir de la premisa de que se debe contar con un Switch OpenFlow, cuyos requerimientos básicos se explican a continuación:

Un switch OpenFlow consiste básicamente de 3 partes[2]: primero, una Tabla de Flujo, con una acción asociada con cada entrada de flujo la cual le indica al switch como procesarlo; segundo, un Canal Seguro que conecta al switch con proceso de control remoto que recibe el nombre de "controlador" (controller), permitiendo que los paquetes y comandos puedan ser intercambiados entre el controlador y el switch a través el protocolo OpenFlow, el cual proporciona un camino abierto y estandarizado que permite la comunicación en cuestión, por lo cual mediante el uso del protocolo el controlador puede añadir, actualizar y borrar "entradas de flujo" tanto de forma reactiva como proactiva.

Es importante segmentar los switches en dos categorías principales: Switches OpenFlow que no soportan procesamiento de nivel 2 y nivel 3 y Switches Ethernet y los habilitadores de OpenFlow, estos poseen interfaces del protocolo como una característica extra.

Switches OpenFlow dedicados: un switch OpenFlow dedicado un elemento simple de "camino de data" (datapath) que envía paquetes entre puertos, de acuerdo como se defina en un proceso a control remoto. En la Figura Nº4 se puede apreciar un switch OpenFlow "ideal"; en este dispositivo los flujos son ampliamente definidos y son limitados solo por las capacidades de una implementación en particular de la tabla de flujos. Por ejemplo: un flujo podría ser una conexión TCP , todos los paquetes de una dirección MAC o IP particular, todos los paquetes con la misma etiqueta de VLAN o todos los paquetes que vengan del mismo puerto. Para experimentos relacionados a paquetes que no son IPv4, un flujo puede definirse como todos los paquetes que tengan en común una cabecera (aunque no esté estandarizada).

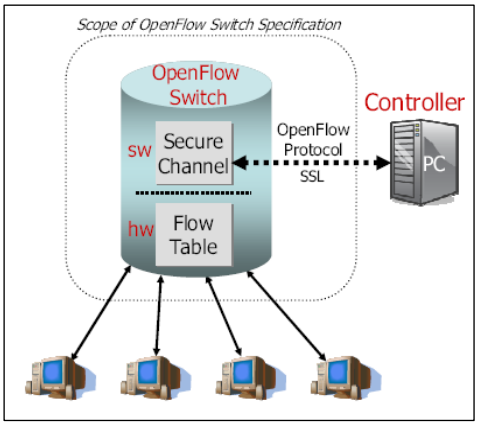

Figura 4: Switch OpenFlow Ideal

Cada entrada de flujo puede tener una acción asociada simple, las tres básicas (soportadas por todos los switches OpenFlow) son:

1. Enviar el flujo de paquetes a un puerto (o puertos) establecidos. Esto permite que los paquetes sean enrutados a través de la red. En la mayoría de los switches esta función se espera tome lugar en la velocidad de línea.

2. Encapsular y enviar estos flujos de paquetes al controlador ("controller").El paquete se envía por el canal seguro donde se encapsula y se envía al controlador, típicamente se usa por el primer paquete en un nuevo flujo, para que el controlador pueda decidir si el flujo debe ser añadido a la tabla de flujos, en algunos experimentos puede ser utilizado para enviar todos los paquetes al controlador de manera que sean procesados.

3. Lanzamiento de este flujo de paquetes, estos pueden ser utilizados para seguridad, para frenar la negación de ataques de servicios o para reducir el descubrimiento de tráfico falso de broadcast por parte de los usuarios finales.

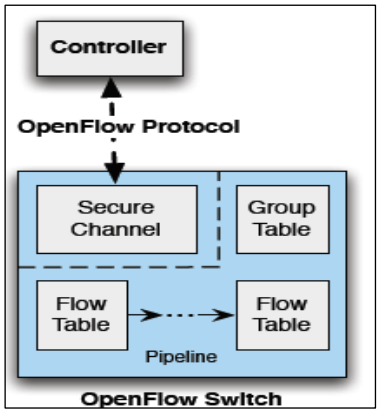

Figura 5: Estructura de un Switch OpenFlow

2.2.3 Tablas de OpenFlow:

2.2.3.1 Tabla de Flujos:

Cada tabla de flujo en el switch contiene un conjunto de entradas, cada una de estas representan campos coincidentes (match fields), contadores (counters) o una serie de instrucciones que aplicar a los paquetes coincidentes.

Cada tabla de flujo (Tabla Nº1) contiene los siguientes componentes:

Tabla 2: Principales componentes de una entrada en una tabla de flujo

Match Fields	Counters	Instructions

a) Campos coincidentes (Match Fields): para detectar los paquetes, estos consisten en el puerto de entrada y cabeceras así como opcionalmente metadata especificada por una tabla previa.
b) Contadores (counters): permiten actualizar los paquetes coincidentes.
c) Instrucciones (Instructions): permiten modificar el conjunto de acciones o el procesamiento canalizado.

La coincidencia de paquetes comienza en la tabla de primera fila y puede continuar a tablas de filas adicionales. Las entradas hacen la coincidencia de paquetes según orden de prioridad, con el resto de coincidencias que se ubican según cada tabla utilizada. Si una entrada de coincidencia es "encontrada" las instrucciones asociadas con la fila específica se ejecutan, si no se encuentra coincidencia en alguna de las filas de tabla, la salida dependerá de la configuración del switch: es posible el paquete sea remitido al controlador por la canal OpenFlow, desechado o puede continuar a verificar otras filas de tablas.

Un switch alienado con OpenFlow puede ser de dos formas: Solo OpenFlow o un Híbrido OpenFlow. Los switches "solo OpenFlow" soportan únicamente operaciones OpenFlow, en esos switches todos los paquetes son procesados por la tubería ("pipeline") OpenFlow y no pueden ser procesados de alguna forma alternativa.

Los switches híbridos OpenFlow soportan tanto operaciones con OpenFlow como aquellas de switching Ethernet estándar, por ejemplo: Switching Ethernet L2 tradicional, aislamiento de VLAN, enrutamiento L3, ACL y procesamiento de calidad de servicios. Estos switches deberían proveer de un mecanismo de clasificación aparte de OpenFlow que enrute tramas tanto para una tubería OpenFlow como para una estándar. Por ejemplo un switch puede utilizar una etiqueta VLAN o un puerto de entrada del paquete para decidir si procesar el paquete utilizando una tubería u otra. Así como también podría dirigir todos los paquetes directamente a la tubería

OpenFlow, esto será determinado según el mecanismo de clasificación sea configurado.

Un switch híbrido OpenFlow puede también permitir a un paquete ir desde la tubería OpenFlow a la tubería "normal" a través de los puertos virtuales "NORMAL" e "INUNDACIÓN". La tubería OpenFlow de cada switch contiene múltiples tablas de flujo que a su vez contiene varias. La tubería de procesamiento OpenFlow define como los paquetes interactúan con aquellas tablas. Un switch OpenFlow con una única tabla de flujo es válido, en este caso el procesamiento de tubería se ve altamente simplificado.

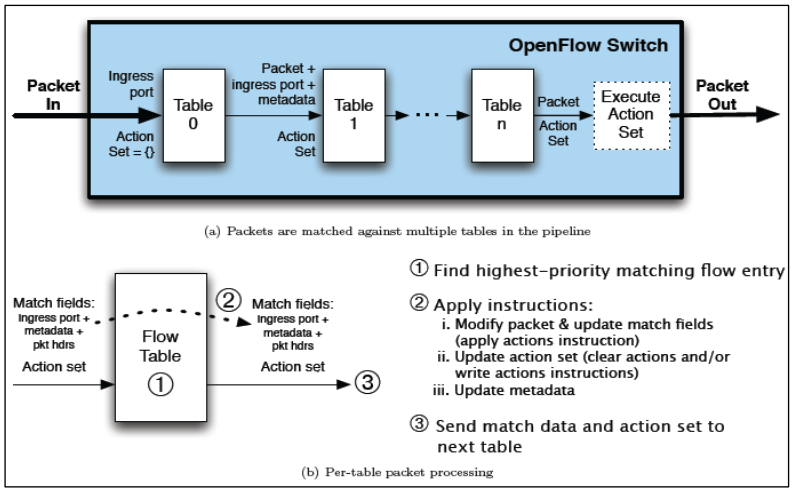

Figura 6: Flujo de paquetes del procesamiento Pipeline

2.2.3.2 Tabla de Grupo:

Una tabla de grupo consiste en "entradas de grupo". La habilidad de un flujo para apuntar a un grupo permite a OpenFlow representar métodos adicionales de transmisión.

Cada entrada de grupo contiene lo siguiente:

Tabla 3: Estructura de una Entrada de Grupo (Identificador de grupo, Tipo de grupo, Contadores y Cubos de acción).

Group Identifier	Group Type	Counters	Action Buckets

a) Identificador de Grupo: un Integer de 32 bits sin firmar que sirve únicamente para identificar al grupo.
b) Tipo de Grupo: Determina la semántica del grupo.
c) Contadores: Permiten la actualización de la información una vez que los paquetes son procesados por un grupo.

d) Acciones asociadas: Representados por una lista ordenada de acciones, donde cada una contiene un conjunto de acciones a ser ejecutadas así como también parámetros asociados.

2.2.3.2.1 Tipos de Grupos:

Se encuentran definidos los siguientes tipos de grupos.

"Todos": Ejecutan todas las acciones dentro del grupo. Este grupo se utiliza para transmisión multicast o broadcast. El paquete se clona de forma efectiva para cada cubo; un paquete por cubo es procesado. Si un cubo dirige un paquete explícitamente fuera del puerto de ingreso, este paquete clonado es declinado. Si el controlador desea transmitir por el puerto de ingreso, el grupo debería incluir un cubo "extra" que incluya una acción de salida al puerto virtual OFPP_IN.

"Seleccionado": Ejecuta una acción del grupo. Los paquetes son enviados a un cubo único del grupo, basado en un algoritmo de selección de interrupción computada. Toda la configuración y estado del algoritmo de selección es externo a OpenFlow. Cuando un puerto en específico de un cubo de un grupo determinado se vienen abajo, el switch puede restringir la selección de cubos dentro del conjunto remanente (aquellos con acciones de transmisión a puertos activos) en lugar de paquetes declinados destinados a ese puerto. Este comportamiento puede reducir la ruptura de un link o switch caído.
"Indirecto": Ejecuta una acción definida en este grupo. Permite múltiples flujos o grupos para apuntar a un identificador de grupo más rápido, soportando convergencia más rápida y eficiente (Por ejemplo: para siguientes saltos de transmisión IP). Este tipo de grupo es idéntico al grupo "Todos" solo que con un solo cubo.

"Conmutación por error rápida": Ejecuta el primer cubo "vivo" (activo).Cada acción está asociada con un puerto específico y/o un grupo que controlar su estado activo . Esto le permite al switch cambiar la transmisión sin requerir un round-trip (ida y vuelta) al controlador. Si ningún cubo está vivo, los paquetes son descartados. Este tipo de grupo debe implementar el mecanismo de "conexión de vida".

2.2.4 Canal OpenFlow:

El canal OpenFlow es la interfaz que conecta cada switch OpenFlow a un controlador. A través de esta interfaz, el controlador configura y administra el switch, recibe eventos del switch y envía paquetes fuera de este.

Entre el camino de datos y el canal OpenFlow, la interfaz se implementa de forma específica, sin embargo todos los mensajes del canal OpenFlow deben poseer un formato acorde al protocolo OpenFlow. El canal OpenFlow es usualmente encriptado utilizando TLS (Transport Layer Security), pero este canal puede correr directamente en TCP (Transmission Control Protocol). La capacidad de soportar múltiples controladores simultáneos no está actualmente definida.

2.2.5 Visión general de Protocolo OpenFlow:

El protocol OpenFlow soporta tres (03) tipos de mensajes, de controlador-a-switch, asíncrono, y simétrico, cada uno con múltiples sub-tipos. Los mensajes Controlador-a-switch son iniciados por el controlador y utilizados para administrar o inspeccionar directamente el estado del switch. Los mensajes asíncronos son iniciados por el switch y utilizados para actualizar el controlador de los eventos de red y cambiar al estado de switch. Los mensajes simétricos inician ya sea por el switch o por el controlador sin solicitud. A continuación una descripción detallada de los mensajes presentes en OpenFlow.

2.2.5.1 Mensajes Controlador-a-Switch:

Los mensajes Controlador/switch son inicializados por el controlador y pueden o no pueden requerir una respuesta por parte del switch.

Características: El controlador puede requerir las capacidades de un switch al transmitir una solicitud de características; el switch debe dar una respuesta que especifique las capacidades y características del switch. Esto es comúnmente realizado al momento que se establece el canal OpenFlow.

Configuración: El controlador es capaz de establecer y pedir parámetros de configuración en el switch. El switch solo responde a una solicitud emitida por el controlador.

Estado- de-Modificación: los mensajes de estados de modificación son enviados por el controlador par administrar el estado de los switches. El principal propósito es añadir/borrar y modificar flujos/grupos en las tablas OpenFlow y establecer las propiedades del switch.

Lectura-Estado: los mensajes de lectura-estado son utilizados por el controlador para recopilar estadísticas del switch.

Salida-de-Paquete: estos son utilizados por el controlador para enviar paquetes fuera del puerto especificado en el switch y enviar paquetes recibidos a través de mensajes

del tipo “Entrada-de-Paquetes”. Los mensajes tipo “salida-de-paquete deben contener un paquete lleno o una identificación de buffer (regulador) que haga referencia al paquete almacenado en el switch. El mensaje debe contener de igual forma una lista de acciones que deben ser aplicadas en el orden en el que son especificadas; una lista de acciones vacía (que no posea ningún tipo de especificaciones) declina el paquete.

Barrera: Una solicitud de barrera/ o mensajes de respuesta son utilizados por el controlador para asegurar que las dependencias de los mensajes se han cumplido o que las notificaciones para operaciones se han recibido completamente.

2.2.5.2 Mensajes Asíncronos:

Los mensajes asíncronos son enviados sin que el controlador los solicite al switch. Los switches transmiten mensajes asíncronos al controlador para denotar la llegada de un paquete, el estado del switch cambia, o denota un error. Los cuatro (04) tipos de mensajes asíncronos se describen a continuación:

Entrada-de-Paquete: Para todos los paquetes que no tienen una coincidencia de entrada de flujo, un evento de “Entrada-de-Paquete” puede ser enviado al controlador (dependiendo de la tabla de configuración). Para todos los paquetes transmitidos al puerto virtual del controlador, siempre se transmite un evento de “Entrada-de-Paquete” al controlador. Si el switch tiene suficiente memoria para amortiguar los paquetes que son transmitidos controlador, el evento contendrá alguna fracción de la cabecera del paquete (la cual tiene un valor por defecto de 128 bytes) y un identificador de buffer que utilizará el controlador cuando el switch se encuentre listo para transmitir el paquete. Los switches que no soporten amortiguación interna (internal buffering) deben transmitir todo el paquete al controlador como parte del evento. Los paquetes amortiguados serán usualmente procesador a través de un mensaje de tipo “Salida-de-Paquete” por parte del controlador, o expirarán de forma automática después de un intervalo de tiempo.

Eliminación de Flujo: cuando se añade una entrada de flujo al switch por un mensaje de modificación de flujo, un valor de tiempo al vacío indica cuando la entrada debe ser removida debido a la falta de actividad, de igual forma que el tiempo indica en que momento la entrada debe ser removida, independientemente de su estado de actividad/inactividad.

Estatus del Puerto: Se espera el switch transmita mensajes de estatus del puerto a los controladores a medida que cambie el estado de configuración de puerto. Estos eventos incluyen cambios en los eventos de estatus del puerto (Por ejemplo: Si es ejecutada alguna modificación directamente por un usuario).

Error: El switch está en capacidad de notificar al controlador de la existencia de problemas mediante la utilización de mensajes de error.

2.2.5.3 Mensajes Simétricos:

Los mensajes simétricos son transmitidos sin que haya solicitud previa, en cualquier sentido.

Hola: Los mensajes "hola" son intercambiados entre el switch y el controlador al inicio del establecimiento de la conexión.

Eco: La respuesta de eco/mensajes de respuesta puede ser transmitida por parte del switch o del controlador y deben retornar como una respuesta de eco. Pueden ser utilizados para medir la latencia o el ancho de banda de una conexión controlador-switch, de igual manera para verificar el tiempo de vida.

Experimentación: los mensajes de "experimentación" proporcionan una manera estándar para que los switches OpenFlow ofrezcan funcionalidades dentro del espacio de mensajes OpenFlow. Esta área se encuentra en período de pruebas y está previsto sean incorporadas mas funcionalidades en futuras revisiones del protocoloo.

2.2.5.4 Establecimiento de la conexión:

El switch debe ser capaz de establecer comunicación con un controlador con una dirección IP de usuario configurable, utilizando un puerto en específico. Si el switch conoce la dirección IP del controlador, el switch inicia una conexión TCP o TLS estándar con el controlador. El tráfico hacia y desde el canal OpenFlow no fluye a través la tubería "OpenFlow ". Por este motivo, el switch debe identificar el tráfico de entrada local antes de verificarlo con las tablas de flujo. En especificaciones de futuras versiones del protocolo, será descrito un protocolo de descubrimiento de controlador dinámico en el cual la dirección IP y el puerto de comunicaciones con el controlador son determinados durante el tiempo de ejecución.

Cuando se establece una conexión OpenFlow, cada lado de la conexión debe transmitir un mensaje OFPT_HELLO con la versión de campo en su valor más alto dentro del protocolo OpenFlow soportado. Al recibir este mensaje, el receptor debe calcular la versión de protocolo OpenFlow a utilizarse, tomando como parámetro que sea una versión más baja que la del emisor y la del receptor.

Si la versión que se establece es soportada por el receptor, entonces la conexión puede proceder, de otra forma el receptor debe responder con un mensaje OFPT_ERROR con un campo de tipo de OFPET_HELLO_FAILED con el campo de código que diga

OFPHFC_COMPATIBLE y opcionalmente un string de tipo ASCII que explica la situación de la data y posteriormente se terminaría la conexión.

2.2.5.5 Interrupción de la Conexión:

En el caso que el switch pierda el contacto con el actual controlador, como resultado de una respuesta de eco cuyo tiempo se haya vencido, o una sesión de TLS finalizada o cualquier tipo de desconexión establecida, debería este buscar el contacto con otros controladores de respaldo. No está determinado en que orden el dispositivo contacta a los diferentes controladores respaldo.

El switch debería inmediatamente entrar ya sea en \"modo seguro de fallos" o \"modo independiente de fallos", si pierde la conexión con el controlador, dependiendo de la implementación del switch y de su configuración en el \"modo seguro de fallos", el único cambio en el comportamiento del switch es que los paquetes y mensajes destinados al controlador de corriente son declinados. Los flujos continúan hasta expirar de acuerdo a sus tiempos de espera dentro del \"modo seguro de fallos", en dicho modo el switch procesa todos los paquetes utilizando el puerto OFPP_NORMAL, en otras palabras, el switch actúa como un switch legacy Ethernet o como un router.

En el momento se conecta un controlador nuevo, se mantiene la entrada de flujo existente. El controlador tiene la opción de borrar todas las entradas de flujo si así lo desea. La primera vez que un switch se inicia, operará ya sea en \"modo seguro de fallos" o en \"modo independiente de fallos".

2.2.5.6 Encriptación:

El switch y el controlador pueden comunicarse a través de una conexión TLS. La conexión TLS es iniciada en el switch por el controlador de arranque, que esta localizado por defecto en el puerto TCP 6633. El switch y el controlador mutuamente autentican mediante el intercambio de certificados firmados por una llave privada especificada. Cada switch debe ser configurable por el usuario con un certificado para autenticar el controlador (certificado de controlador) y el y otro para autenticar hacia el controlador (certificado de switch).

2.2.5.7 Manejo de Mensajes:

El protocolo OpenFlow proporciona una confiable entrega y procesamiento de mensajes, pero no provee reconocimiento automático o garantía de procesamiento de mensaje.

"Entrega de Mensaje": se garantiza la entrega del mensaje, a menos que haya un fallo total de la conexión, en este caso el controlador no debe asumir nada acerca del estado del switch (Ejemplo: el switch podría haberse ido a modo fail\standalone).

“Procesamiento de Mensaje”: los switches deben procesar cada mensaje recibido por un controlador en su totalidad, generando posiblemente una respuesta. Si un switch no puede procesar completamente un mensaje recibido por parte de un controlador, debe ser enviado de vuelta un mensaje de error. Para mensajes de “salida –de-paquetes”, el procesamiento pleno del mensaje no garantiza que el paquete incluido realmente salga del switch. El paquete incluido puede ser silenciosamente suprimido después del procesamiento debido a congestión por parte del switch, políticas QoS o si se envía a un puerto bloqueado o inválido.

Además, los switches deben enviar al controlador todos los mensajes generados por cambios de estado interno, como los flujos removidos o los mensajes de “entrada-de-paquete”. Sin embargo, los paquetes recibidos en puertos de datos que deben ser transmitidos al controlador pueden ser abortados debido a la congestión o política de QoS dentro del switch y generar mensajes de “no-entrada”. Estas supresiones podrían ocurrir para paquetes con una acción de salida explícita hacia el controlador. Estas supresiones podrían también ocurrir cuando un paquete falla en coincidir cualquier entrada en una tabla y la acción por defecto de dicha tabla es enviar al controlador.

Los controladores son libres de abortar mensajes, pero deben responder a mensajes “hello” y “eco” para prevenir que el switch abandone la conexión.

Pedido de Mensaje: el pedido puede ser asegurado a través del uso de mensajes de barrera. En la ausencia de mensajes de barrera, los switches pueden arbitrariamente reorganizar mensajes para maximizar el rendimiento, los controladores no deben depender de una orden específica de procesamiento.

En particular, los flujos pueden ser insertados en tablas en un orden distinto que los mensajes recibidos por el switch. Los mensajes no deben ser reorganizados a través de un mensaje de barrera y dicho mensaje de barrera debe ser procesado solamente cuando el mensaje previo ya lo haya sido. Específicamente:

1. Los mensajes previos a la barrera deben ser plenamente procesados antes de la misma, incluyendo enviar cualquier respuesta o errores resultantes.

2. La barrera debe por lo tanto ser procesada y una respuesta de barrera enviada.

3. Los mensajes después de la barrera pueden posteriormente comenzar a ser procesados

Si dos mensajes del controlador dependen uno del otro (por ejemplo un flujo añadido que le siga una "salida-de-paquete" OFPP_TABLE), deberán estar separados por un mensaje barrera

2.2.6 Utilización de OpenFlow:

Como es el caso con muchas plataformas experimentales, el conjunto de experimentos o pruebas excederá aquellos que realmente se están llevando a cabo dado que muchos son primeras versiones y/o investigaciones universitarias cerradas por ahora, sin embargo se tratará de dar un resumen suficiente para verificar el impacto de esta plataforma en nuevas aplicaciones arquitecturas.

Ejemplo Nº1: Administración de la Red y Control de Acceso, se usará de referencia "Ethane"[10] como el primer ejemplo dado que fue la investigación que podría decirse que marcó el inicio de OpenFlow, de hecho un switch OpenFlow puede verse como una generalización de un switch de camino de datos de "Ethane". "Ethane" utilizaba una implementación específica de un controlador, adecuado para administración y control de red, que maneja la entrada y enrutamiento de flujos. La idea básica de "Ethane" es permitir a los administradores de red definir una política con "anchura" de red en el controlador central, la cual es forzada al realizar decisiones de control de admisión por parte de cada flujo nuevo. Un controlador verifica un nuevo flujo en contra de un conjunto de reglas, algunos ejemplos podrían ser: "Los invitados puede comunicarse usando HTTP, pero solo a través de un proxy de red" o "Los teléfonos IP no pueden comunicarse con portátiles", por nombrar reglas de este tipo. Un controlador asocia paquetes con sus remitentes al administrar todos los enlaces entre nombres y direcciones, esencialmente toma los DNS, DHCP y autentica a todos los usuarios cuando estos se unen; al mantener un control de cuales puertos de switch (o Puntos de Acceso) a los que ellos están conectados.

Podría ser posible plantearse una extensión de "Ethane" en donde una política dicte que flujos particulares se envíen a los procesos del usuario en un controlador, por ende permitiendo a los investigadores la capacidad de hacer procesado específico en la red.

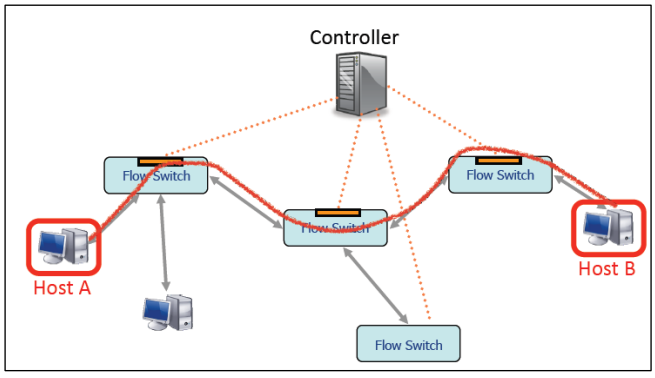

Figura 7: Esquema de Ethane, control por flujo, centralizado y reactivo.

Ejemplo Nº2: Vlans, OpenFlow podría fácilmente suministrar a los usuarios su propia red aislada, de la misma forma lo hacen las VLANS. El enfoque más sencillo es declarar deforma estática un conjunto de flujos donde se especifiquen los puertos accesibles por una VLAN determinada. El tráfico identificado por parte de un usuario en específico (Ejemplo, aquel originado de puertos de switch específicos o de una dirección MAC específica) es etiquetado por lo switches (a través de una acción) con su ID de VLAN respectivo.

Un Enfoque más dinámico podría ser utilizar un controlador para administrar la autenticación de usuarios y utilizar el conocimiento de la ubicación de los usuarios para etiquetar el tráfico durante el funcionamiento.

Ejemplo Nº3: Clientes de VOIP inalámbricos, para este ejemplo se considera como base un nuevo mecanismo de Hand-off para móviles con Wifi habilitado. En este experimento los clientes de VOIP establecen una conexión nueva a través de la red OpenFlow. Un controlador es implementado para rastrear la ubicación de los clientes, re-enrutar conexiones al reprogramar las tablas de flujos a medida que el usuario se mueve a través de la red, permitiendo un Hand-Off "sin costuras o parches" desde un punto de acceso hacia otro.

Ejemplo nº4: Una red NO-IP, en los ya previamente explicados se tomaba la premisa que la red era IP, pero OpenFlow no exige que los paquetes sean de un formato en específico, solo quela tabla de flujo pueda emparejarse con la cabecera del paquete. Esto permitiría experimentos donde se cambien los esquemas de nombres, direccionamientos y enrutamiento. Hay varias maneras que un switch con OpenFlow habilitado pueda soportar tráfico NO IP. Un ejemplo: los flujos podrían ser identificados utilizando su cabecera Ethernet (dirección MAC), un nuevo valor de Ether Type o a nivel IP por una nueva versión del protocolo (IPv6, IPv4, etc.). Generalmente se espera que los switches del futuro permitan a un controlador crear una máscara genérica (offset+valor+máscara) de manera que permita a los paquetes ser procesados según una especificación del investigador/usuario en cuestión.

Ejemplo Nº5: Procesar paquetes en lugar de flujos. Los ejemplos previamente mencionados se refieren a flujos donde el controlador toma las decisiones acerca de cuando comienza el flujo. Hay sin embargo experimentos que requieren que se procese cada paquete sea procesado independientemente. Por ejemplo un sistema de detección de intrusos que inspecciona cada paquete, un mecanismo de control de congestión o cuando se modifica el contenido de los paquetes, como en los casos se convierte de un protocolo a otro.

Existen dos formas básicas de procesas paquetes en una red con OpenFlow habilitado. Lo principal (y más simple) consiste en forzar todo los paquetes de un flujo

para pasar a través de un controlador. Para hacer esto un controlador no añade una nueva entrada de flujo dentro del switch, solo permite que por defecto se envíe cada paquete al controlador. Esto tiene la ventaja de que proporciona flexibilidad, por el mismo costo del rendimiento. Esto podría aportar una forma útil de probar la funcionalidad de un nuevo protocolo, pero es poco factible de que sea de mucho interés para despliegues en grandes redes

La segunda manera de procesar paquetes es enrutarlos a un switch programable que efectúe el procesamiento de paquetes, por ejemplo un router programable basado en NetFPGA [4]. La ventaja es que los paquetes pueden ser procesados en tasa de línea en una forma definible por el usuario. La Figura Nº 5 explica como puede llevarse a cabo esta configuración, en la cual el switch habilitado para OpenFlow opera esencialmente como un patch-panel para permitir a los paquetes acceder al NetFPGA. En algunos casos la tabla NetFPGA (una tabla PCI que se conecta a un PC Linux en este ejemplo) puede ser remplazada en el departamento de cableado junto un switch con OpenFlow habilitado o (más probablemente) en un laboratorio.

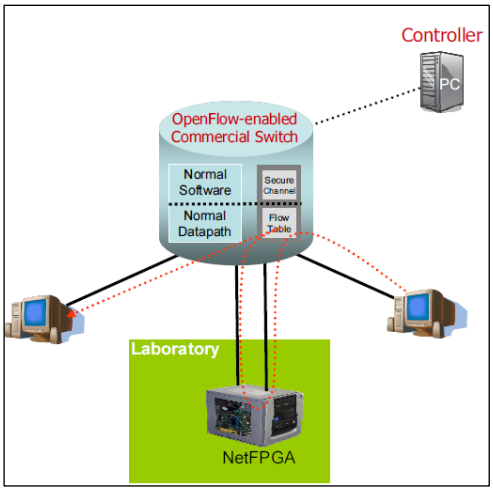

Figura 8: Switch con OpenFlow con conexión a NetFPGA

2.2.7 Equipos que soportan OpenFlow:

Como se pudo comentar en el apartado anterior, nivel experimental se han planteado pruebas a los siguientes niveles:

2.2.7.1 Experimentos en el Nivel de flujo:

- Protocoles de enrutamiento definidos por el usuario
- Control de Admisión
- Control de Acceso de Red
- Administración de Red
- Administración de energía

- Movilidad VOIP y hand-off

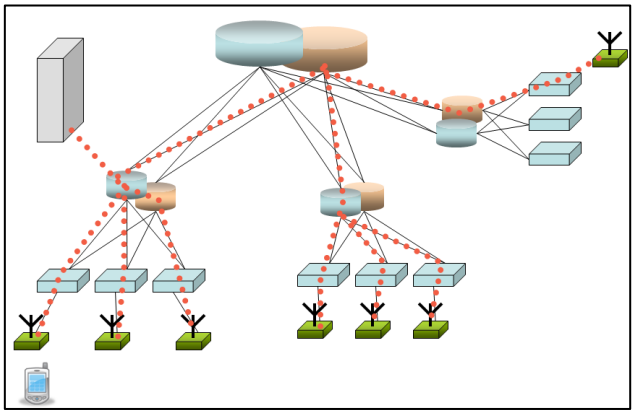

Figura 9: Ejemplo experimental en el nivel de flujo (Movilidad).

2.2.7.2 Experimentos en el Nivel de paquete:

- Lento: El controlador maneja el procesamiento de paquetes
- Rápido: Redirección del flujo a través de hardware programable
- Routers modificados, firewalls, NAT, control de congestión, entre los factores habilitados para "experimentar".

A nivel comercial existen equipos actualmente en mercado con interesantes características de soporte de OpenFlow (equipos G8254 y G8264T, de la casa IBM y HP) [11], [12]:

Características generales del G8264 / G8264T:

G8264:

- Equipo optimizado para aplicaciones que requieren un alto ancho de banda y baja latencia
- Soporta arquitectura " Virtual Fabric" así como el protocolo OpenFlow
- Capacidad de hasta 64 SFP (Small form-factor pluggable transceivers) de 10 Gbs más 10 puertos con un factor de forma 1U.
- Pruebas futuras para 4 QSFP (Quad-small form-factor pluggable transceivers) de 40 Gb y adición de más puertos.

G8264T

- Capacidad para 48 conexiones 10GBase-T además de cuatro (04) conexiones QSFP de 40 Gb más una con factor de forma 1U.

- Conectividad flexible y de bajo coste con opción para entornos con conexiones de 10 Gb, que soportan distancias de hasta 100 metros.

Diseñados con parámetros de alto rendimiento, el RackSwitch G8264 y el RackSwitch G8264T de IBM son ideales para el procesamiento de data de hoy en día, como las aplicaciones en nube y las altas y voluminosas tasas de datos. Ambos equipos son switches diseñados para la empresa que ofrecen una buena tasa para líneas, un alto ancho de banda en switching, filtrado y procesamiento de colas de tráfico sin que se vea retrasado dicho tráfico de data; posee la capacidad de amortiguar grandes centros de data así como también posee redundancia de potencia y ventiladores de manera que sea posible garantizar la disponibilidad para aquellos negocios sensibles al tráfico de datos. El G8264 y el G8264T soportan tecnología IBM VMready, estándares y soluciones para administrar máquinas virtuales (VMs) y también entornos de tecnología en la nube (cloud).

El Rackswitch G8264 es ideal para aplicaciones sensibles a latencia como son los clusters de alto rendimiento computacional (comunes en operaciones financieras); de igual forma ambos equipos están en capacidad de soportar protocolos de última tecnología como DCB/CEE (Data Center Bridging/Converged Enhanced Ethernet) de manera que puedan operar en redes con canales de fibra sobre Ethernet (FCoE) adicionalmente a iSCS (Internet Small Computer System Interface) y NAS (Network Attached Storage).

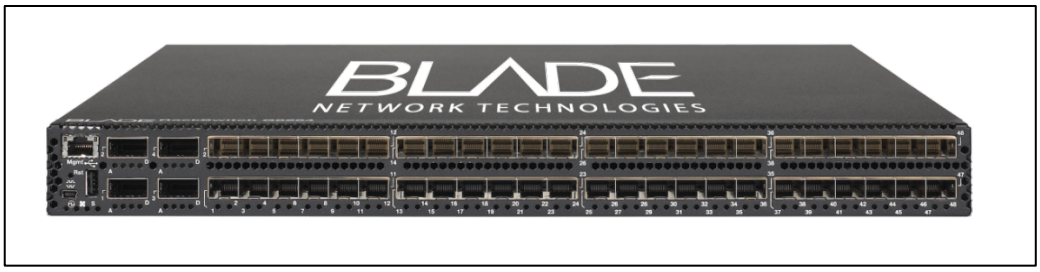

Figura 10: IBM RackSwitch G8264

2.3 Open vSwitch

Open Vswitch es un software multicapa bajo la licencia de código abierto Apache 2, la meta de este grupo es implementar [13] un switch con calidad de producción que pueda soportar interfaces de administración estándares y que abra a programación y control externo las funciones de transmisión.

Open Vswitch está adecuado para funcionar como un switch virtual en entorno VM. Adicionalmente para exponer control estándar e interfaces de visibilidad a la capaz de

red virtual, se diseñó para soportar distribución a través de múltiples servidores físicos. Open vSwitch soporta múltiples tecnologías de virtualización basadas en Linux incluyendo Xen/Xen Server, KVM y Virtual Box.

El grueso de código se escribe en C de plataforma-independiente y es fácilmente portado a otros ambientes. La versión actual de Open vSwitch suporta las siguientes funcionalidades:

- Modelo de VLAN estándar 802.1Q con "trunk" y puertos de acceso.
- Unión de NIC con o sin LACP (Link Aggregation Control Protocol) en el switch de subida de enlace.
- NetFlow,sFlow(R), SPAN (Switched Port Analyzer),RSPAN (Remote Switched Port Analyzer) y ERSPAN (Encapsulated Switched Port Analyzer) para el incremento de visibilidad.
- Configuración de Qos (Calidad de Servicio) y políticas de dicho proceso
    - Túneles con soporte de CAPWAP (Control and Provisioning of Wireless Access Points), GRE (Generic Routing Encapsulation) y GRE sobre IPsec.
    - Conectividad de gestión de fallos bajo el estándar 802.1ag
    - Soporte de OpenFlow y otras extensiones.
    - Configuración de Bases de datos de forma transaccional para enlaces con C y Python.
    - Capa de compatibilidad para Puente de código Linux
    - Transmisión de alto rendimiento utilizando el módulo kernel de Linux.

El módulo de Kernel Linux incluido soporta Linux 2.6.18 y versiones superiores, con pruebas enfocadas en la versión 2.6.32 con Centos y parches de Xen. Open Vswitch también tiene soporte para Citrix Xenserver y hosts de la versión de Linux Red Hat Enterprise.

Open vSwitch también puede operar, bajo un costo de su rendimiento, enteramente dentro de espacio de usuario (userspace) sin ayuda de un módulo de Kernel. Esta implementación de usuario de espacio debería ser más fácil de llevar al puerto que la versión del switch basado en Kernel, se considera en fase experimental.

Los principales componentes de esta distribución son los siguientes:

- Ovs-vswitchd, un daemon ("demonio") implementado por el switch, junto a un módulo Kernel de Linux para switching basado en flujo.

Servidor-ovsdb, un servidor de base de datos ligera que realiza consultas al ovs-vswitchd para obtener información de su configuración.

- Ovs-brcompatd, un daemon ("demonio") que permite al ovs-switch actuar como reemplazo de un bridge Linux en distintos ambientes, junto a un módulo de Kernel Linux para interceptar las llamadas ioctls del bridge.
- Ovs-dpctl, una herramienta para configurar el módulo Kernel del switch.
- Scripts y especificaciones para constuir RMPs (Red Hat Package Managers) para servidores Citrix Xenserver y Linux Red Hat Enterprise. El RPMs de Xen server le permite a Open vSwitch ser instalado en un servidor Citrix Xen como reemplazo de switch, con funcionalidades adicionales.
- Ovs-vsctl, una utilidad para consultas y actualizaciones de la configuración del ovs-vswitchd.
- Ovs-appctl, una utilidad que envía comandos para los daemons ("demonios") del Open Vswitch operativos.
- Ovsdbmonitor, una herramienta GUI para supervisión remota de bases de datos OVS y tablas de flujo OpenFlow.

Open vSwitch también proporciona las siguientes herramientas:

- Ovs-controller, un simple controlador de OpenFlow.

- Ovs.ofctl, una utilidad para consultas y control de switches y controladores OpenFlow.

- Ovs-pki, una utilidad para crear y administrar la infraestructura de llave pública para switches OpenFlow.

- Un parche de tcpdump que le permite analizar mensajes OpenFlow.

2.4 VNX:

VNX es una herramienta de virtualización abierta con propósito general diseñada para ayudar a la construcción de bancos de pruebas de forma automática. Permite la definición y despliegue automático de escenarios de redes realizados con maquinas virtuales de diferentes tipos (Linux, Windows, FreeBSD, etc) que se interconectan siguiendo una topología definida por el usuario, posiblemente conectada a redes externas. [14]

VNX ha sido desarrollada por el RTSI (Grupo de Redes y Servicios de Telecomunicación e Internet) del DIT (Departamento de Ingeniería Telemática) de la UPM. VNX es una herramienta útil para pruebas de aplicaciones y servicios de redes sobre complejos bancos de pruebas hechos de nodos y redes virtuales, así como también para crear complejos laboratorios de redes que permiten a los estudiantes

interactuar con escenarios de redes realistas. Como otras herramientas similares que impulsan la creación de escenarios de redes virtuales (Ejemplo: GNS3, MLN o Marionnet), VNX proporciona una forma de administrar bancos de pruebas evitando la compleja inversión y administración requerida para crearlas a nivel físico ("real").

VNX está compuesto por dos partes:

- Un lenguaje XML que permite describir el escenario de red virtual (Lenguaje de especificación VNX)
- El programa de VNX, que analiza la descripción de escenario y construye y administra el escenario virtual sobre una máquina Linux.

VNX viene con una versión distribuida (EDIV) que permite el despliegue de escenarios virtuales sobre clusters de servidores Linux, mejorando la escalabilidad de los escenarios compuestos por decenas o hasta cientos de máquinas.

VNX se construye como continuidad de la herramienta VNUML (VNUML (Virtual Networks over User Mode Linux) y brinda nuevas funcionalidades que sobrepasan las principales propiedades de VNUML:

- Integración de nuevas plataformas de virtualización que permitan a las máquinas virtuales correr sobre distintos sistemas operativos (Windows, FreeBSD,etc) además de Linux, en este sentido:
  - VNX utiliza libvirt para interactuar con las capacidades de virtualización del host, permitiendo la utilización de la mayoría de las plataformas de virtualización disponibles para Linux (KVM, Xen, etc.)
  - Integra las plataformas de virtualización de router Dynamips y Olive para permitir emulaciones limitadas de los routers CISCO y Juniper.
- Administración individual de las máquinas virtuales.
- Autoconfiguracón y capacidades de ejecución de commandos para diversos sistemas operativos: Linux, FreeBSD y Windows (XP Y 7).

Pueden revisarse los comandos y consideraciones necesarias para la instalación de VNX en el Apéndice B de este trabajo.

3 Propuesta de Prueba de Implementación:

En la actualidad, las empresas se ven en una constante búsqueda para mejorar sus estructuras de costos y maximizar la utilización de recursos, por lo tanto la adopción de sistemas y protocolos abiertos representa una de las actividades con mayor dinamismo en los últimos años; tomando en cuenta también que una vez una propuesta es aceptada e implantada plenamente en el sector educativo es solo cuestión de tiempo para que se expanda dentro del ámbito empresarial..

Entre los puntos clave de este trabajo se encuentra la implantación a pequeña escala de una aplicación de OpenFlow, así como su simplificación.

Antes de definir los pasos para dicha instalación, es necesario hacer una revisión y análisis, de las herramientas necesarias para esta actividad.

3.1 Objetivos:

Antes de comenzar con el marco teórico de esta propuesta de implantación de OpenFlow, se expone un conjunto de objetivos que se ha intentado abordar en los siguientes subapartados:

- Conocer y especificar las características técnicas e históricas de la virtualización de redes así como predecesores y causas de la aparición del protocolo OpenFlow.

- Obtener una visión global del estado del arte actual de la tecnología correspondiente a dicho protocolo.

- Estudiar las diferentes aplicaciones para la tecnología en cuestión y los principales equipos comerciales.

- Implantar un escenario con OpenFlow, donde se transformará un controlador hub en un switch mediante el software Open Vswitch y se enlazará con el entorno virtualizado creado en VNX.

- Describir brevemente mediante un análisis técnico las ventajas y bondades que aporta el protocolo así como también la simplificación de su uso mediante una simple integración con VNX.

3.2 Visión General:

En el tutorial que se usó de guía, la idea principal era poder comparar diferentes funcionalidades mediante: la transformación de un controlador Hub en un switch controlador con capacidades cognitivas, luego a un switch con capacidades cognitivas con flujo acelerado y posteriormente la extensión de un ambiente de un switch único a un ambiente de múltiples switches con múltiples hosts.

Entre las principales actividades de prueba será posible:

- Visualizar las tablas de flujo con dpctl
- Analizar mensajes OpenFlow con Wireshark
- Simular entornos multi-switch
- Establecer puntos de referencia del controlador.

3.3 Pre-requisitos:

Las pruebas alineadas con el objetivo global del TFM no requieren un vasto equipamiento, un simple equipo con 1GB mínimo (recomendación de más de 2GB) de memoria RAM y al menos 5GB de espacio libre de Disco Duro (de ser posible más, se recomienda esta opción).

Las pruebas pueden ser llevadas a cabo en diferentes sistemas operativos (Linux, OSX, Windows), más sin embargo se recomienda Linux, porque involucra la instalación de menos paquetes y herramientas (dado que el mismo OS trae por defecto muchos de los requerimientos), para la máquina de pruebas seleccionada se ha elegido el sistema Operativo Linux en su distribución Ubuntu 12.04

3.4 Herramientas de Software Requeridas:

Se requiere que las herramientas de Software sean descargadas de forma individual, dentro de las mismas se incluyen las siguientes:

Software de virtualización

- Una terminal que soporte SSH
- Un servidor X
- Una Imagen VM.

En el tutorial que sirve como referencia se encuentra una imagen comprimida de Virtual Box (formato vdi.), el Virtual Box básicamente permite correr una máquina virtual dentro de una máquina física.

Podemos ver resumidos los requerimientos de Sistema Operativo, Software y Terminal en la siguiente Tabla:

Tabla 4: Requerimientos a nivel de SW para las pruebas.

Tipo de S.O.	Versión del S.O.	Software de Virtualización	Servidor X	Terminal
Windows	7	VirtualBox	Xming	PuTTY
Windows	XP	VirtualBox	Xming	PuTTY
Mac	OS X 10.6 Snow Leopard	VirtualBox	X11	Terminal.app
Mac	OS X 10.5 Leopard	VirtualBox	X11	Terminal.app
Mac	OS X 10.4 Tiger	VirtualBox	X11	Terminal.app
Mac	OS X 10.3 Panther	VirtualBox	Mac X11 server o XQuartz	Terminal.app
Linux	Ubuntu 10.04	VirtualBox	X server instalado por defecto.	gnome terminal + SSH built in

3.4.1 Virtual Box:

Es un software de virtualización para arquitecturas x86/amd64, creado originalmente por la empresa alemana innotek GmbH. Actualmente es desarrollado por Oracle Corporation como parte de su familia de productos de virtualización. Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como «sistemas invitados», dentro de otro sistema operativo «anfitrión», cada uno con su propio ambiente virtual.

Entre los sistemas operativos soportados (en modo anfitrión) se encuentran GNU/Linux, Mac OS X, OS/2 Warp , Microsoft Windows, y Solaris/OpenSolaris, y dentro de ellos es posible virtualizar los sistemas operativos FreeBSD, GNU/Linux,OpenBSD, OS/2 Warp, Windows, Solaris, MS-DOS y muchos otros.

La aplicación fue inicialmente ofrecida bajo una licencia de software privativo, pero en enero de 2007, después de años de desarrollo, surgió Virtual Box OSE (Open Source Edition) bajo la licencia GPL 2. Actualmente existe la versión privativa Oracle VM Virtual Box, que es gratuita únicamente bajo uso personal o de evaluación, y esta sujeta a la licencia de "Uso Personal y de Evaluación Virtual Box" (Virtual Box Personal Use and Evaluation License o PUEL) y la versión Open Source, VirtualBox OSE, que es software libre, sujeta a la licencia GPL.

VirtualBox ofrece algunas funcionalidades interesantes, como la ejecución de maquinas virtuales de forma remota, por medio del Remote Desktop Protocol (RDP), soporte iSCSI, aunque estas opciones no están disponibles en la versión OSE.

En cuanto a la emulación de hardware, los discos duros de los sistemas invitados son almacenados en los sistemas anfitriones como archivos individuales en un contenedor llamado Virtual Disk Image, incompatible con los demás software de virtualización.

Otra de las funciones que presenta es la de montar imágenes ISO como unidades virtuales ópticas de CD o DVD, o como un disquete.

Tiene un paquete de controladores que permiten aceleración en 3D, pantalla completa, hasta 4 placas PCI Ethernet (8 si se utiliza la línea de comandos para configurarlas), integración con teclado y ratón.

3.4.2 SSH (Secure Shell) :

SSH™ (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación

remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas. [15]

SSH está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la shell de comando, tales como telnet o rsh. Un programa relacionado, el scp, reemplaza otros programas diseñados para copiar archivos entre hosts como rcp. Ya que estas aplicaciones antiguas no encriptan contraseñas entre el cliente y el servidor, evite usarlas mientras le sea posible. El uso de métodos seguros para registrarse remotamente a otros sistemas reduce los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto.

El Terminal SSH: conecta con el Tutorial de OpenFlow, creado utilizando Putty sobre Windows o SSH de OSX/Linux.

3.4.2.1 Características de SSH:

El protocolo SSH proporciona los siguientes tipos de protección:

- Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.
- El cliente transmite su información de autenticación al servidor usando una encriptación robusta de 128 bits.
- Todos los datos enviados y recibidos durante la sesión se transfieren por medio de encriptación de 128 bits, lo cual los hacen extremamente difícil de descifrar y leer.
- El cliente tiene la posibilidad de reenviar aplicaciones X11 [2] desde el servidor. Esta técnica, llamada reenvío por X11, proporciona un medio seguro para usar aplicaciones gráficas sobre una red.

Ya que el protocolo SSH encripta todo lo que envía y recibe, se puede usar para asegurar protocolos inseguros. El servidor SSH puede convertirse en un conducto para convertir en seguros los protocolos inseguros mediante el uso de una técnica llamada reenvío por puerto, como por ejemplo POP, incrementando la seguridad del sistema en general y de los datos.

Red Hat Enterprise Linux contiene el paquete general de OpenSSH (openssh) así como también los paquetes del servidor OpenSSH (openssh-server) y del cliente (openssh-clients). Consulte el capítulo titulado OpenSSH en el Manual de administración del sistema de Red Hat Enterprise Linux para obtener instrucciones sobre la instalación y el desarrollo de OpenSSH. Observe que los paquetes OpenSSH requieren el paquete OpenSSL (openssl). OpenSSL instala varias bibliotecas criptográficas importantes, permitiendo que OpenSSH pueda proporcionar comunicaciones encriptadas.

3.4.3 Xterm:

El programa Xterm es un emulador de terminal para el sistema de ventanas windows X, proporciona compatibilidad con terminales DEC VT102/VT220 y Tektronic 4014 para programas que no pueden usar el sistema Windows de forma directa. El terminal Xterm, está conectado aun host en la red virtual. [16]

3.4.4 Dpctl:

Utilidad de línea de comandos que transmite rápidamente mensajes OpenFlow, aplicación útil para la visualización del puerto de switch y de los estados de flujos, además permite también la inserción de entradas de flujo, es soportada por el tutorial oficial del wiki de OpenFlow, más sin embargo en instalaciones del paquete actual se usa el paquete ovs.

3.4.5 Wireshark:

Wireshark es el analizador de protocolos más utilizado que hay actualmente, permite la captura y búsqueda interactiva del tráfico que atraviesa una red, actualmente se considera un estándar a nivel no solo educativo sino también industrial y comercial. [17]

3.4.6 Iperf:

Utilidad de comandos utilizara para la evaluación de rendimientos en las comunicaciones de una red local y posterior optimización de los parámetros, Con IPerf es posible medir el ancho de banda y rendimiento de una conexión entre dos host. Se trata, básicamente de una herramienta cliente-servidor. [18]

3.4.7 Mininet:

Es una plataforma de emulación de red, que crea redes definidas por software (Tipo OpenFlow por ejemplo) totalmente escalables (cuyas dimensiones pueden ser de hasta cientos de nodos, según la configuración deseada) que están contenidas en una PC que utiliza procesamiento Linux. [19]

Mininet permite crear, interactuar, personalizar y compartir de forma rápida un prototipo de red definido mediante software al mismo tiempo proporcionar un camino fácilmente adaptable para la migración a hardware.

3.4.8 Cbench:

Cbench es un framework para evaluación comparativa, pruebas y análisis de clusters computacionales paralelos basados en Linux, al mismo tiempo (como es en este caso) se utiliza para probar el establecimiento de flujo de controladores OpenFlow.[20]

3.5 Escenario y Pruebas iniciales:

Para la prueba inicial se utilizó una laptop HP modelo: Pavilion DV6-3150US, equipada con un procesador Intel i5, a la cual se le procedió a instalar el sistema operativo Linux en su distribución Ubuntu 12.04 junto con el paquete de Virtual Box y la herramienta VNX. Inicialmente una vez instaladas se sometieron a pruebas dichas herramientas para verificar su compatibilidad con el sistema operativo y equipo en cuestión.

La red base que se emulará inicialmente incluye 3 hosts y un switch, una vez dominados todos los aspectos de este entorno, se procederá al establecimiento de un controlador OpenFlow, dicha topología se tomó como base del tutorial de http://www.openflow.org/wk/index.php/OpenFlow_Tutorial el cual se adjunta traducido y completado en el Apéndice A del trabajo.

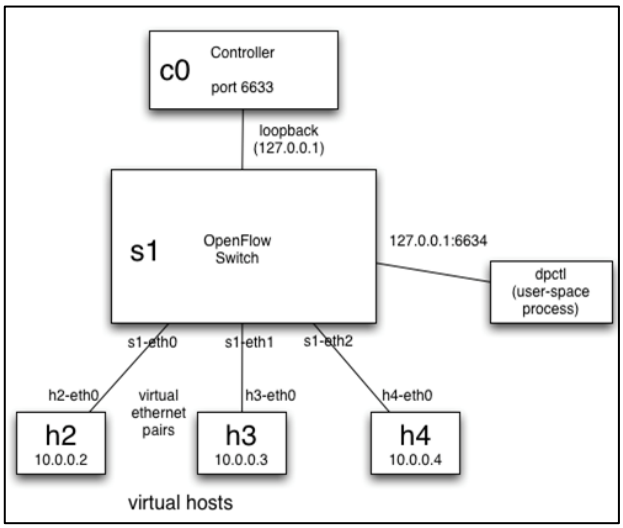

Figura 11: Escenario de red para pruebas del Tutorial OpenFlow

Inicialmente se siguieron los pasos del tutorial referenciado en el apéndice "A", donde se pudo verificar el funcionamiento de Virtual Box, mediante la creación de la maquina virtual de nombre: Vmopenflowtut1, y de allí se verifico la utilización de OpenFlow a través de la herramienta de virtualización Mininet

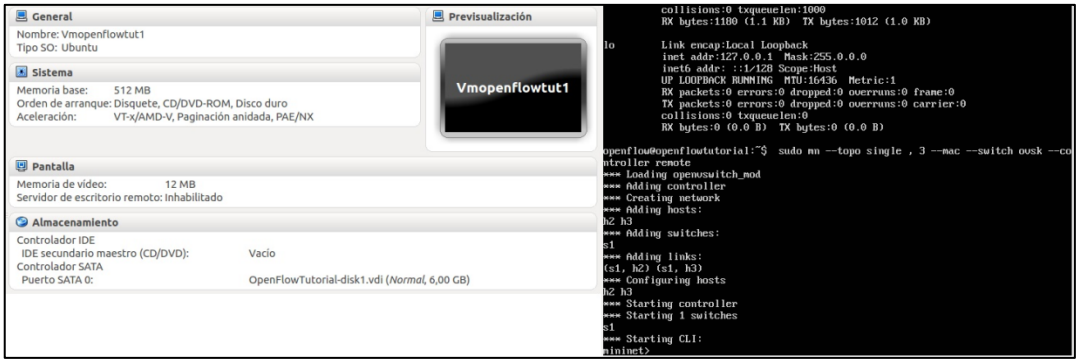

Figura 12: Creación de Máquina Virtual Vmopenflowtut1 con Virtual Box basada en Imagen de tutorial OpenFlow.

Se verificó la conectividad del entorno virtual a través de ping entre los hosts de la topología y se siguieron las instrucciones de la referencia nombrada; una vez clarificado el funcionamiento del protocolo y de los equipos que forman parte de él se procedió a diseñar la homologación de dicho escenario con la herramienta VNX , la cual al mismo tiempo se combinaría con Open vswitch para que este software le de las funcionalidades del protocolo OpenFlow a los Switches a ser instalados.

3.6 Simplificación a través de Vinculo con la herramienta VNX:

El primer paso constituyó en diseñar la red a ser probada, a continuación se adjunta código XML que se creó para que la herramienta VNX pudiese crear el escenario de virtualización deseado (se creó bajo el nombre "openflow1").

```
<?xml version="1.0" encoding="UTF-8"?>
<vnx xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="/usr/share/xml/vnx/vnx-2.00.xsd">
        <global>
                <version>2.0</version>
                <scenario_name>openflow1</scenario_name>
                <automac offset=""/>
                <vm_mgmt type="none" />
        </global>
        <net name="S1" mode="virtual_bridge" type="lan"/>
        <vm name="h2" type="libvirt" subtype="kvm" os="linux">
                <filesystem
type="cow">/usr/share/vnx/filesystems/root_fs_ubuntu</filesystem>
                <mem>256M</mem>
                <if id="1" net="S1">
                        <ipv4>10.0.0.2/24</ipv4>
                </if>
                <forwarding type="ip"/>
        </vm>
        <vm name="h3" type="libvirt" subtype="kvm" os="linux">
                <filesystem
type="cow">/usr/share/vnx/filesystems/root_fs_ubuntu</filesystem>
                <mem>256M</mem>
```

```
                <if id="1" net="S1">
                        <ipv4>10.0.0.3/24</ipv4>
                </if>
                <forwarding type="ip"/>
        </vm>
        <vm name="h4" type="libvirt" subtype="kvm" os="linux">
                <filesystem
type="cow">/usr/share/vnx/filesystems/root_fs_ubuntu</filesystem>
                <mem>256M</mem>
                <if id="1" net="S1">
                        <ipv4>10.0.0.4/24</ipv4>
                </if>
                <forwarding type="ip"/>
        </vm>
</vnx>
```

Una vez creado el código correspondiente al escenario, se abre la terminal de la PC de ejecución del proyecto, en donde se entra como usuario root y se abre el código a través de la herramienta VNX.

```
cd /root/openflow
sudo vnx -f openflow.1xml -v -create
```

Mediante los comandos previos hemos inicializado el escenario virtual, donde se desplegarán 3 terminales (1 por consola de host) y en el cual la tipología creada corresponde con la siguiente gráfica:

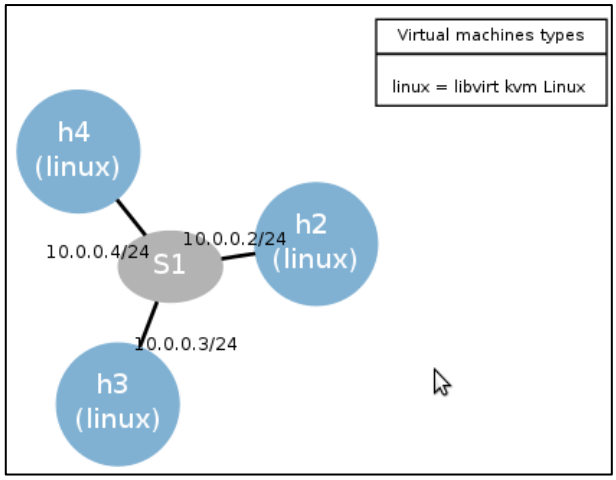

Figura 13: Diseño de red de escenario virtual obtenido mediante VNX

Una vez establecidos los hosts, puede verificarse la accesibilidad entre ellos a través de ping; el siguiente paso corresponde a la instalación del software Open Vswitch, el

cual permitirá al bridge S1 funcionar como un equipo con protocolo OpenFlow monitorizado por su controlador correspondiente; a continuación de explica la instalación de Open vswitch en la distribución Ubuntu versión 12.04 LTS

```
$ wget http://openvswitch.org/releases/openvswitch-<build>.tar.gz
$ tar xf openvswitch-<build ver>
$ cd openvswitch-<build ver>
libssl-dev iproute tcpdump linux-headers-`uname -r`
$./boot.sh
$./configure – -with-linux=/lib/modules/`uname -r`/build
$make
$sudo make install
$insmod datapath/linux/openvswitch.ko
$sudo touch /usr/local/etc/ovs-vswitchd.conf
$mkdir -p /usr/local/etc/openvswitch
$ovsdb-tool create /usr/local/etc/openvswitch/conf.db
$ovsdb-server /usr/local/etc/openvswitch/conf.db \
–remote=punix:/usr/local/var/run/openvswitch/db.sock \
–remote=db:Open_vSwitch,manager_options \
–private-key=db:SSL,private_key \
–certificate=db:SSL,certificate \
–bootstrap-ca-cert=db:SSL,ca_cert –pidfile –detach –log-file
$ovs-vsctl –no-wait init
$ovs-vswitchd –pidfile –detach
```

En el momento que terminemos la instalación , cuando ejecutemos el comando "show" podremos ver todas las interfaces y sus conexiones. Una vez finalizada la instalación es necesario mediante los comandos "init" y "set –controller"iniciar el proceso de OpenFlow e instaurar el bridge Net0 como controlador.

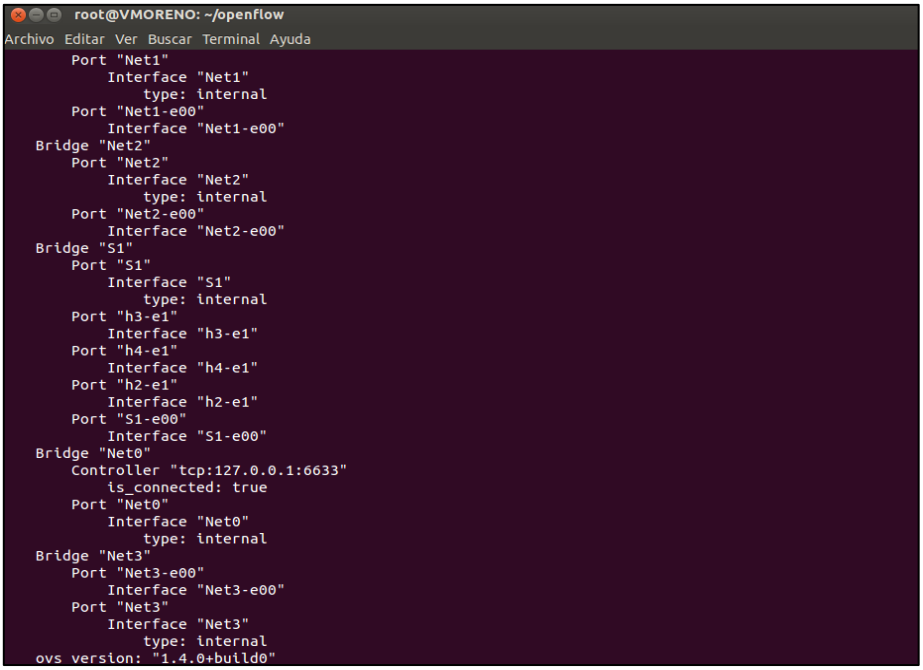

Figura 14: Escenario virtualizado con Openvswitch

Puede verificarse la conectividad y el funcionamiento del protocolo a través del analizador wireshark:

```
sudo wireshark &
```

Al ejecutar la apertura del software wireshark, podremos visualizar el trafico ICMP generado por las pruebas de conectividad entre los diferentes hosts (a través de ping), así como también posibilita comprobar el enlace entre el switch y su controlador; a continuación se adjunta el comando necesario para establecerlo manualmente y se exponen las trazas de tráfico mencionados

```
ovs-vsctl set-controller Net0 tcp:127.0.0.1:6633
```

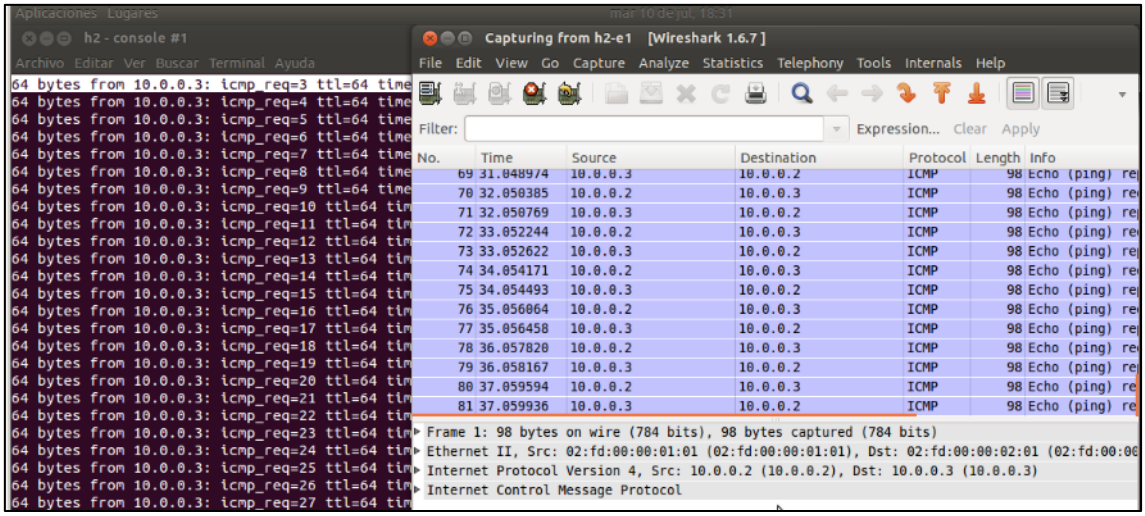

Figura 15: Confirmación del tráfico ICMP al hacer pruebas ping entre los hosts.

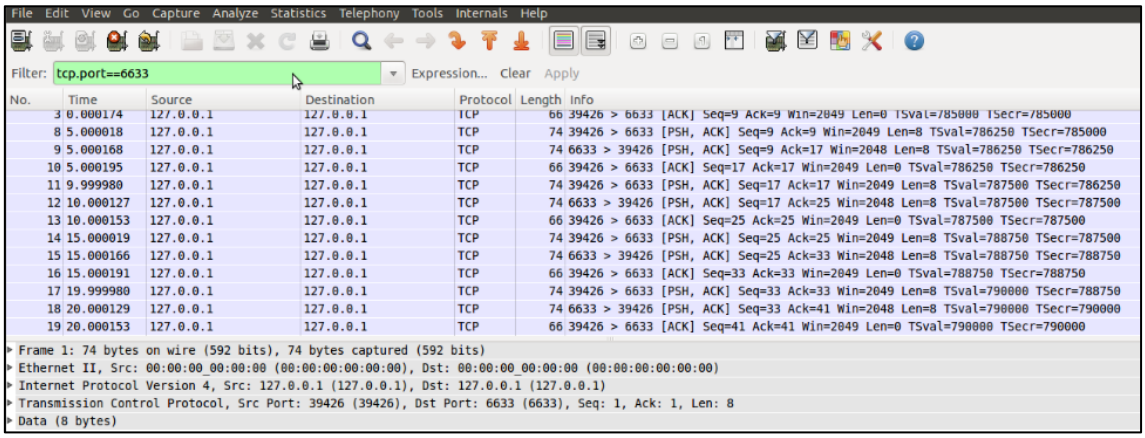

Figura 16: Captura de Pantalla con tráfico TCP entre Controlador y Switch

Al quedar demostrado el tráfico entre los elementos de la red, puede comprenderse la filosofía de trabajo del estándar OpenFlow en la cual la información viaja a través de un canal seguro para conectar los hosts de una determinada red mediante un controlador.

4 Conclusiones:

En este capítulo se analizará básicamente el caso práctico realizado en el trabajo y se comparará con tutorial usado como referencia, así como también se dará un vistazo a que usos y capacidades podrían atribuírsele a OpenFlow, especialmente ahora que existe un particular e intenso interés en el mismo en el entorno del mundo de investigación.

Para poder llevar a cabo este proyecto, ha sido necesario realizar un gran esfuerzo de búsqueda de información, ya que el estándar OpenFlow, aunque cuenta con muchas fuentes de documentación existente, fue necesaria una extracción de las partes fundamentales de las mismas, traducción y análisis y resumen de muchas otras.

Un aspecto motivador de este TFM ha sido el carácter investigador, que ha solicitado para poder solventar los problemas que han ido apareciendo sobre la marcha (producto del mismo desconocimiento de la tecnología). Además, al ser un tema tan innovador, dicha investigación contribuye directamente con el enriquecimiento personal del autor y proporciona una tecnología novedosa y dinámica.

En primera instancia, el tutorial corresponde una buena opción para acercarse por primera vez a la tecnología de virtualización de redes y poder tener una idea general del hardware y software que se necesitan para poder crear escenarios de pequeña escala, se tiene contacto con los siguientes instrumentos de software: virtual box, mininet,SSH y según se elija que opción se desea para el controlador: NOX, POX, Beacon, entre otros. Sin embargo al momento de crear desde el principio la topología deseada, a través de la herramienta VNX (que solo requiere 2GB y conocimientos de lenguaje XML) es posible crear entornos de red totalmente a nuestro gusto (de hecho en su documentación ejemplo no se limita a casos de un mismo fabricante).

En el caso del tutorial la elección del controlador queda a libertad de elección según los conocimientos de lenguaje y capacidades que posea la maquina en procesamiento y disco; más sin embargo un punto fundamental de este trabajo era demostrar la veracidad del funcionamiento del protocolo para usuarios principiantes e incluso sin gran conocimiento previo y pudiendo constatar la presencia de " canal seguro" que le da al protocolo en estudio tanta importancia y que es uno de los motivos principales para que esté siendo considerado por grandes casas de fabricación de hardware.

lo más conveniente sería analizar y comprender toda la filosofía detrás del mismo, posteriormente examinar su utilidad para por último poder tener en mente implantarlo; a miras a futuro la continuación lógica de este trabajo sería una red a mediana escala donde el controlador se llevará de forma independiente (con recomendación del uso de POX) y tuviese control de un ambiente de red compuesto por diferentes sub-redes cada una con su respectivo switch OpenFlow correspondiente.

Al establecer claramente las bondades del estándar es indudable se abre la puerta a numerosas e innovadoras aplicaciones, de hecho entre los proyectos a futuro que se tienen planteados desde diferentes grupos de investigación y que se observan a través del análisis del mismo pueden mencionarse:

- Arquitecturas flexibles en las cuales el Switch OpenFlow sea la conexión central entre redes de varios niveles y que represente un enlace transparente para otros switches (como es el caso del aún experimental FlowVisor) [30]

- Implementación de OpenFlow a través de redes de ISPs, donde se puedan garantizar servicios de calidad en sus dos focos principales: Ingeniería de tráfico y control de VPNs a nivel 2 3. La versatilidad y capacidad de adaptación que ofrece OpenFlow representaría una opción interesante para lidiar con las complicaciones que trae la variedad de protocolos para un mismo sistema, como lo han hecho muchas tecnologías empresariales actuales la simplificación mediante unificación de estándares es una alternativa latente e importante (en la actualidad se lleva en especial atención poder lograr la implementación de MPLS a través de OpenFlow) [21]

Como es notable las posibilidades que ofrece el protocolo son amplias y el mismo se encuentra en un punto donde diversas alternativas pueden plantearse y es través de investigación universitaria con pequeñas y medianas iniciativas donde podrían estar las claves para poder masificar y promover su uso dentro del entorno empresarial.

Apéndice A:

Los pasos seguidos para verificar la funcionalidad de la plataforma pueden verse simplificados en el esquema de la Figura Nº 12, se seguirán y explicarán todos los pasos esquematizados y posteriormente es que se planteará una posibilidad de integración con VNX para simplificación.

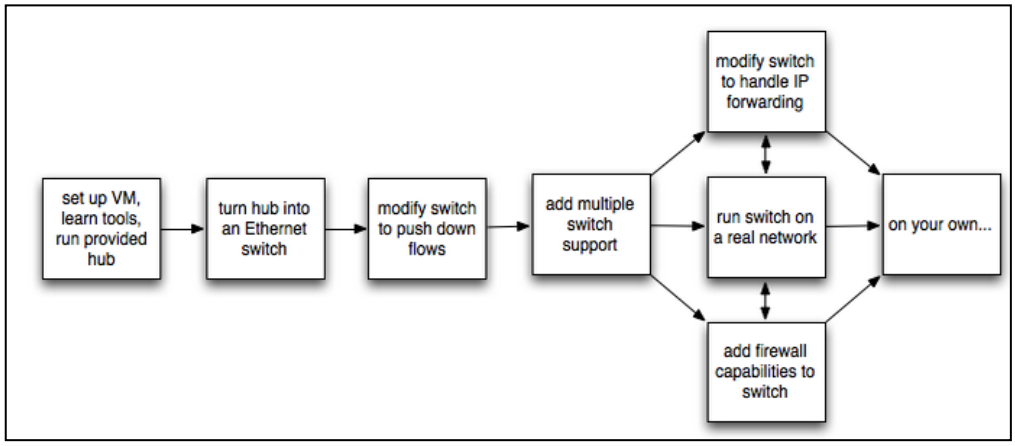

Figura 17: Secuencia de pasos para ejecución de tutorial OpenFlow

En primera instancia a través de Mininet se establece la topología de 3 hosts y 1 switch, junto con la dirección MAC de cada host (equivalente a su dirección IP) y señalar a un controlador remoto que por defecto señala al localhost, a continuación el paso a paso de como es el establecimiento de la topología a través de Mininet :

- Creación de 3 hosts virtuales, cada uno con una dirección IP separada
- Creación de un software OpenFlow único con una kernel con 3 puertos
- Conexión de cada host virtual al switch con un cable Ethernet virtual
- Establecimiento de la dirección MAC de cada host, en equivalencia con su dirección IP
- Configuración del switch OpenFlow para conectarlo con un controlador remoto.

Una vez familiarizado con el entorno inicial y de las herramientas habilitadas , inicialmente se realizarán pruebas varias básicas para entender el funcionamiento y posteriormente es que se podrá explorar la configuración de topologías con distintos elementos propios de OpenFlow

Utilización de dpctl:

Dpctl es una utilidad que viene con la distribución de referencia de OpenFlow, esta permite la visibilidad y control de una única tabla de flujos. Es especialmente útil para "depurar" (debugging), al visualizar estados y contadores de flujos. La mayoría de switches OpenFlow pueden iniciarse con un puerto pasivo de escucha, desde el cual se

puede sondear el switch, sin tener que añadir un código de depuración (debugging) al controlador.

A continuación, podemos visualizar el comando "show" que conecta al switch y arroja su estado y capacidades.

```
$ dpctl show tcp:127.0.0.1:6634
$ dpctl dump-flows tcp:127.0.0.1:6634
```

Prueba Ping:

Como prueba inical se plantea hacer un "ping" desde el host3 (h3) hasta el host 2 (h2).

Para este caso, se debe tomar en cuenta que la tabla de flujos no esté vacía y que el controlador esté conectado al switch , de manera que se evite el fracaso del ping

```
mininet> h2 ping -c3 h3
```

Para la corrección y verificación:

```
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
$ dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
$ dpctl dump-flows tcp:127.0.0.1:6634
mininet> h2 ping -c3 h3
```

Es posible no se obtengan respuesta de ping, como consecuencia de que las entradas de flujo expiren antes del inicio de la prueba de pin, la cual tiene un tiempo por defecto de 60 segundos, lo que quiere decir que el flujo expira luego de 60 segundos se no hay entrada de tráfico, una solución es instalar una entrada de flujo con mayor tiempo de salida.

```
$ dpctl add-flow tcp:127.0.0.1:6634
in_port=1,idle_timeout=120,actions=output:2
```

Wireshark:

La imagen VM incluye un analizador Wireshark para OpenFlow pre-instalado, como se discutió previamente wireshark es sumamente útil en la observación y análisis de mensajes de protocolos, así como también para depuración (debugging) en general.

Se inicia una nueva terminal SSH y se conecta a la Maquina Virtual a través de X11, se indicarán siempre los comandos necesarios para sistemas Linux/MAC.

```
$ ssh -X openflow@[guest ip address]
$ sudo wireshark &
of
```

Es posible que se obtengan mensajes de "advertencia" al utilizar la herramienta wireshark como usuario root, esto no representa ningún inconveniente y debe aceptarse; dentro de dicha herramienta es posible al elegir la opción de captar y visualizar el tráfico transmitido, el comando "of" permitirá establecer un filtro para el control de tráfico OpenFlow.

Arranque de controlador y visualización de mensajes en Wireshark:

Una vez que se tenga el analizador wireshark analizando todo el tráfico, es que debe arrancarse el controlador de referencia OpenFlow en la terminal SSH.

```
$ controller ptcp:
```

El commando $ controller ptcp: inicia un simple controlador que actúa como un switch con habilidades cognitivas sin instalar entradas de flujo. A partir de esto, es posible visualizar muchos mensajes expuestos en wireshark, a continuación un breve resumen de los posibles mensaje a obtener:

Tabla 5: Mensajes intercambiados entre switch y controlador

Mensaje	Tipo	Descripción
Hello	Controller->Switch	Una vez establecido del handshake de TCP, el controlador envía su número de versión al switch.
Hello	Switch->Controller	El switch contesta con su número de versión soportada.
Features Request	Controller->Switch	El controlador consulta cuales puertos están disponibles
Set Config	Controller->Switch	En este caso, el controlador consulta al switch para enviar expiraciones de flujos.
Features Reply	Switch->Controller	El switch contesta con una lista de puertos, la velocidad de los mismos y las tablas y acciones soportadas.
Port Status	Switch->Controller	Permite al switch informar al controlador acerca de cambios en velocidad de puertos o conectividad, este tipo de mensaje puede ser ignorado, dado que más que cualquier utilidad asociada parece representar un Bug.

Dado que todos los mensajes son transmitidos a través de hosts locales utilizando Mininet, determinar el transmisor del mensaje puede ser confuso cuando hay un entorno de muchos switches emulados. Sin embargo, esto no representará un inconveniente dado que en la topología planteada establece un solo switch. El controlador está en el puerto standard OpenFlow (6633) mientras que el switch está en un puerto de nivel-de-usuario.

Visualización de mensajes OpenFlow por Ping:

En esta sección será posible visualizar los mensajes generados en respuesta a los paquetes, es necesario habilitar los mensajes de echo-request/reply , se adjuntan los comandos a ser aplicados para esta acción y posteriormente hacer un ping para visualizar los mensajes OpenFlow que están siendo utilizados.

```
of && (of.type != 3) && (of.type != 2)
mininet> h2 ping -c1 h3
```

En la ventana de Wireshark, es posible se visualicen varios tipos de mensajes, a continuación se exponen los principales tipos:

Tabla 6: Mensajes OpenFlow

Mensaje	Tipo	Descripción
Packet-In	Switch->Controller	Un paquete ha sido recibido y no coincidió con ninguna de las entradas de la tabla de flujo del switch, causando que el paquete sea enviado al controlador.
Packet-Out	Controller->Switch	El controlador envía el paquete a uno o varios puertos del switch.
Flow-Mod	Controller->Switch	Le indica al switch a añadir un flujo particular a su tabla de flujos.
Flow-Expired	Switch->Controller	Expiración de tiempo de espera, como resultado de un período de inactividad.

Inicialmente se identificará una solicitud ARP que omitida en la tabla de flujo, lo cual genera un mensaje de broadcast Packet-Out. Posteriormente la respuesta ARP es devuelta junto con la dirección MAC, la cual es ahora conocida por el controlador, es posible hacer caer un flu8jo hacia el switch a través de un mensaje de tipo Flow-Mod. Luego a través del switch se empujan flujos para los paquetes ICMP , después las solicitudes de ping van directamente a través del camino de datos y no deberían

incurrir mensajes adicionales; con los flujos conectando a los hosts no hay participación del controlador.

A continuación se expone el commando para lanzar el ping a través de la consola Mininet, es posible sea necesario emitir dicho ping varias veces y comparar tiempos de expiración.

```
mininet> h2 ping -c1 h3
```

Controlador de punto de referencia con iperf:

Iperf es una herramienta de línea de comandos para verificar la velocidad entre dos ordenadores. En la misma se establece el punto de referencia del controlador, después se comparará el mismo con el suministrado por el hub controlador y con el switch basado en flujo (una vez implementado).

A continuación se muestra la secuencia de comandos para habilitar el iperf, comparar en los diferentes hosts virtuales y el rendimiento de un switch de usuario-espacio.

```
mininet> iperf
mininet> exit
$ sudo mn --topo single,3 --mac --controller remote --switch user
mininet> iperf
mininet> exit
```

Creación de un Switch con algoritmo de aprendizaje:

Una vez verificada la funcionalidad, se pasa al siguiente nivel de aplicación de red, se provee con código de inicio para un controlador de hub, una vez que el usuario se familiariza con el hub, modificará el mismo para que actúe como un switch L2 con capacidades cognitivas. En la aplicación a proponer el switch examinará cada paquete y aprenderá el mapeo de puerto fuente. Por lo tanto, la dirección MAC fuente estará asociada al puerto. Si el destino del paquete se asocia con algún puerto, el paquete será transmitido al puerto dado, sino será inundado a través de todos los puertos del switch.

Posteriormente se convertirá el switch en uno basado en flujo, donde al visualizar un paquete con una fuente y destino conocidos causará una entrada de flujo para ser empujado hacia abajo.

Para la prueba establecida, se indicarán las diferentes opciones viables: NOX (correspondiente al lenguaje Python) o Beacon (correspondiente al lenguaje Java) para la implementación del controlador, el principal factor a ser tomado en cuenta para la elección del lenguaje, será trabajar con aquel se esté más familiarizado.

Discusión de opciones para Controladores:

Opción de Elección de Controlador Nº 1: NOX con Python

La primera opción planteada para el controlador es utilizar NOX, una plataforma de controlador que permite escribir el controlador en Python, en C++ o en alguna combinación de ambos lenguajes. NOX es una plataforma open-source que simplifica la creación de software para controlar o monitorear redes. Los programa escritos en NOX (utilizando C++ o Python) tienen control de nivel de flujo de la red, esto significa que ellos pueden determinar cuales flujos son permitidos en la red y que camino deben tomar. [21]

Un factor importante a tomar en cuenta (de hecho el motiva esta plataforma es la opción nº1 recomendada) es que NOX ya viene por defecto listo para utilizarse en la máquina virtual; para estos ejemplos y procedimientos ya no se utilizará la figura de controlador de referencia; es importante verificar que dicho controlador de referencia no está siento utilizado y NOX pueda usar el puerto 6633, a continuación los comandos necesarios en la ventana SSH para inciar Mininet, asegurar que está todo en fase inicial y proceder con el código base de inicio del Hub Python

```
$ sudo killall controller
mininet> exit
$ sudo mn -c
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
$ cd ~/nox/build/src
$ ./nox_core -v -i ptcp: pytutorial
```

Básicamente con la secuencia anterior se le indica a NOX que inicie la aplicación "tutorial", para exponer la información de debug y escuchar de forma pasiva nuevas conexiones en el puerto estándar OpenFlow (Nº 6633). Es posible tome cierto tiempo para que los switches se conecten, cuando un switch OpenFlow pierde su conexión a un controlador, generalmente incrementa el período de intentar contactar un controlador a un máximo de 15 segundos; dado que el switch OpenFlow no está aún conectado, este retardo puede variar entre 0 y 15 segundos. Si esta espera es demasiado grande el switch puede ser configurado para esperar no más de una cantidad de "N" segundos utilizando el parámetro de max-backoff. Alternativamente, es posible salir de Mininet para remover el (o los) switch (es), arrancar el controlador y luego arrancar el mininet para conectarlo inmediatamente; una vez que el Switch OpenFlow se haya conectado, NOX indicará en su pantalla algo parecido a esto:

```
00039|nox|DBG:Registering switch with DPID = 1
```

Si por lo contrario, el switch emite un mensaje de "envió de error en respuesta a la respuesta de la capacidad, asumiendo que no hay apoyo a la gestión", esto no involucra un problema. Open Vswitch tiene extensiones personalizadas para soportar la gestión de una base de datos, pero en este caso no se habilitan en el switch OpenFlow.

Verificación del comportamiento del Hub con tcpdump:

Mediante esta fase se verificará que los hosts pueden hacerse ping entre sí y que todos los hosts ven el mismo comportamiento de tráfico, para realizar esto se crean xterms para cada host y se visualiza el tráfico en cada uno. En la consola Mininet, se inician los siguientes xterms:

```
mininet> xterm h2 h3 h4
```

Se organiza cada xterm de manera que ellos sean visualizados en la pantalla instantáneamente, se indican los comandos para correr tcpdump en los xterms del host 3 y el host4 y luego en el xterm del host 2, transmitir un ping.

```
# tcpdump -XX -n -i h3-eth0
# tcpdump -XX -n -i h4-eth0
# ping -c1 10.0.0.3
```

Los paquetes de ping en este punto van al controlador, quien inunda todas las interfases excepto la transmisora, después de esto es posible visualizar paquetes ARP y ICMP idénticos correspondientes al ping en ambos extremos del tcpdump que está corriendo; se toma en cuenta que así trabaja un hub, este envía todos los paquetes a todos y cada uno de los puertos en una red.

En el caso que un host no-existente no responda, se obtiene un mensaje como el mostrado en la figura Nº, se deben identificar tres solicitudes ARP sin contesta en el xterm tcpdump. Si el código está apagado (off) posteriormente, las solicitudes sin contestar ARP son un indicativo que se están perdiendo paquetes de forma accidental.

```
# ping -c1 10.0.0.5
```

Controlador Benchmark Hub con iperf:

En esta prueba se toma como referencia un hub, inicialmente se debe verificar la accesibilidad, el Mininet debe correr al mismo tiempo que el tutorial NOX, se muestran a continuación los comandos para verificar conectividad y comprar el número de referencia del controlador con el que anteriormente visualizados; tomando en cuenta que cada paquete va hacia el controlador ahora.

```
mininet> pingall
mininet> iperf
```

Apertura del Código de Hub:

Es necesario dirigirse hacia la terminal SSH y detener el controlador Hub NOX, en el archivo **~/nox/src/nox/coreapps/tutorial/pytutorial.py** se realizarán las modificaciones correspondientes.

Cada vez que se cambia y guarda el archivo, es necesario reiniciar NOX, posteriormente se utilizan pings para verificar el comportamiento de la combinación del switch y el controlador como:

a) Hub

b) Switch Ethernet con habilidades cognitivas basado en controlador

c) Switch con habilidades cognitivas con flujo acelerado

Para el caso de a) y b), los hosts que no son el destino de un ping no deben mostrar tráfico tcpdump después de la solicitud ARP de broadcast oficial.

Aprendizaje de Python:

Python es un lenguaje de programación dinámico e interpretado, que permite trabajar rápidamente e integrar sistemas de forma bastante eficiente [22]

- Es dinámico e interpretado; no hay un paso separado de compilación, solo es necesario actualizar el código y volverlo a ejecutar.
- Utiliza las "sangrías" en lugar de "llaves" y puntos y comas para delimitar el código, Cuatro espacios denotan por ejemplo, el cuerpo de un bucle.
- Se escribe dinámicamente, no hay necesidad de pre-declarar variables y los tipos están administrados automáticamente.
- Tiene construidas tablas hash llamadas diccionarios, y vectores llamados listas.
- Es orientado a objetos e introspectivo, es posible fácilmente imprimir las variables de miembros y las funciones de un objeto en el tiempo de ejecución.
- Se ejecuta más lento que un código nativo porque es interpretado. Los controladores de rendimiento crítico podrían desear distribuir el procesamiento de nodos múltiples o conmutar a un lenguaje más optimizado.

A continuación se exhiben en las siguientes figuras, las operaciones más comunes de Python:

Inicializar el diccionario:

```
mactable = {}
```

Añadir un elemento al diccionario:

```
mactable[0x123] = 2
```

Verificar la afiliación del diccionario:

```
if 0x123 in mactable:
    print 'element 2 is in mactable'
```

Imprimir un mensaje de depuración en NOX:

```
log.debug('saw new MAC!')
```

Imprimir un mensaje de error en NOX:

```
log.error('unexpected packet causing system meltdown!')
```

Imprimir las variables de miembros y las funciones de un objeto:

```
print dir(object)
```

Comentar una línea de código:

```
# Prepend comments with a #; no // or /**/
```

Una vez que se tiene la configuración para este controlador y el conocimiento básico de Python asociado, se puede realizar distintas pruebas asociadas, a continuación las principales señaladas por el tutorial:

A. Envío de Mensajes OpenFlow con NOX
B. Análisis de paquetes con las librerías de paquetes NOX.

Opción de Elección de controlador Nº2: Beacon (Java)

Beacon es una plataforma controladora de Java, basada en OpenFlow que soporta tanto operaciones basadas en eventos como por thread. Beacon es fácil de desarrollar utilizando el entorno Eclipse, la máquina donde se pretenda realizar este controlador requerirña Java 6 JDK & JRE [23].

Ejecución del Tutorial de controlador:

En la máquina virtual, no se van a utilizar la referencia de controlador, la cual se ejecuta en el fondo. Los switches todos intentan conectarse al controlador e

incrementaran el período de sus intentos de contactar al controlador, hasta un máximo de 15 segundos. Dado que el switch de OpenFlow no ha sido aún conectado, este retardo podría estar entre 0 y 15 segundos. Si esta espera es muy larga, el switch puede ser configurado para esperar no más de "N" segundos utilizando un parámetro de max-backoff.

Al igual que el caso anterior se verificar la referencia del controlador y que el mininet esté plenamente "limpio" para iniciar.

```
$ sudo killall controller
mininet> exit
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote --ip
<your_host_ip>
```

El parámetro –ip define en cual dirección IP se localiza el controlador. Es necesario tomar en cuenta esa es la dirección IP que el host utilizará para comunicarse con la máquina virtual (VM), la dirección IP de la máquina virtual es (en este caso) 192.168.56.101 y la dirección IP del host es 192.168.56.1, por lo que (como se verá en el siguiente comando) se inicializa el Mininet con esta última.

```
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote --ip
192.168.56.1
```

Una vez inicializados estos parámetros de red, se inicia Beacon; se inicializar desde Eclipse en modo de depuración (debug).

Run -> Debug Configurations

Al estar dentro del menú correspondiente se ingresa al "tutorial beacon" para iniciar ambos y se escuchan las conexiones entrantes en el puerto OpenFlow estándar (el puerto 6633). En el momento el switch OpenFlow se haya conectado, la consola Eclipse debe imprimir lo expuesto en el siguiente cuadro de comandos:

```
osgi> 19:31:51.598 [pool-1-thread-1] INFO
n.beaconcontroller.tutorial.Tutorial - Registering switch with DPID 1
```

Verificación del comportamiento del Hub con tcpdump:

En esta etapa (al igual que con la opción de controlador anterior) se verificará la capacidad que tengan los hosts para hacerse ping los unos a los otros y que todos visualicen el mismo tipo de tráfico. Para hacer esto, se crean xterms para cada host y se observa el tráfico dentro de cada uno; a continuación como se realizara el proceso a través de Mininet con tres xterms:

```
mininet> xterm h2 h3 h4
# tcpdump -XX -n -i h3-eth0
# tcpdump -XX -n -i h4-eth0
# ping -c1 10.0.0.3
```

En este escenario, el paquete de ping se va hacia el controlador, el cual inunda todas las interfases con excepción de la transmisora. Se deben percibir paquetes ICMP y ARP idénticos, correspondientes al ping en ambos xterms , ejecutándose tcpdump.

Controlador Hub de referencia con iperf:

En esta sección se utilizará de referencia el código de hub, el cual es parte del haz del tutorial. En primera instancia se verifica la accesibilidad, se muestran los comandos asociados a esta acción a continuación

```
mininet> pingall
mininet> iperf
```

Apertura del código Hub e inicialización:

Para esta etapa, se debe entrar en Eclipse y detener la ejecución de Beacon; una vez hecho este paso se modifica el archivo **net.beaconcontroller.tutorial/scr/main /java/net/beaconcontroller/Tutorial.java**; a partir de este archivo se necesitarán aproximadamente 30 líneas de código para construir el switch con capacidades cognitivas. Cada vez que se hagan modificaciones y se guarde el archivo, es necesario verificar que se detiene y arranca la instancia en ejecución, luego se deben utilizar pings para verificar el comportamiento del dispositivo. Los hosts que no estén en el destino no deberían mostrar tráfico tcpdump después de la solicitud ARP de broadcast inicial.

Aprendiendo Java y Beacon:

Al igual que en la sección anterior, se enlistan los comandos bases relacionados con el lenguaje de esta opción de controlador (Java)

Para inicializar un Hashmap:

```
protected Map<Integer, Short> macToPort = new HashMap<Integer, Short>();
```

Para añadir un elemento al Hashmap:

```
macToPort.put(mac_address_key, port)
```

Para verificar un elemento del Hashmap:

```
if (macToPort.containsKey(mac_address_key))
   System.out.println("mac_address_key is in hashmap")
```

Para obtener un elemento del HashMap:

```
learned_port = macToPort.get(mac_address_key)
```

Para imprimir un mensaje de debug en Beacon:

```
logger.debug("Learned new MAC!")
```

En adición a estas nociones de Java, es conveniente (de hecho el tutorial lo resalta) tener información y detalles de las clases que son útiles en este tipo de pruebas con Beacon.

Tipos de Clases útiles de Beacon:

Dentro del directorio del tutorial de beacon 1.0.0, existen carpetas beacon-java.doc y openflowj-java.doc, las cuales poseen la documentación esencial para estas clases. A continuación se indican las principales asociadas a la implementación de códigos:

Para transmitir un mensaje a un switch OpenFlow,debe referirse a la clase IOES del switch. La entrada se transmite como una función de hub mediante:

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

Para Observar al OFMatch y el OFPacket para obtener información de la cabecera a través del OpenFlow Packet-in, se puede utilizar el siguiente comando:

```
match.loadFromPacket(pi.getPacketData(), pi.getInPort());
```

A lo largo de estos procedimientos, es posible se desee imprimir un dirección MAC o utilizarla como una llave para un HashMap. OFMatch suministrará la dirección como una matriz de byte, para estos procedimientos podrían utilizarse las siguientes funciones:

```
HexString.toHexString(byte[] bytes) // Convert a string of bytes to a ':'
separated hex string
Integer mac_address_key = Arrays.hashCode(mac_address); // Creates an integer
key from a bytes array.
```

Finalmente, en algún punto será necesario instalar un flujo en la red. Debe utilizarse OFFlowMod y OFActionOutput para hacerlo. A continuación se muestra como inicializar un mensaje de tio flow-mod para un switch en específico.

```
OFFlowMod fm = new OFFlowMod();
```

Es necesario utilizar el "establecedor" (setter) adecuado para construir el flow-mod requerido. Asumiendo que se conoce el puerto de salida para este flujo, este es un fragmento de como incluirlo en el mensaje flow-mod.

```
OFActionOutput action = new OFActionOutput();
action.setMaxLength((short) 0);
action.setPort(outPort);
List<OFAction> actions = new ArrayList<OFAction>();
actions.add(action);
fm.setActions(actions);
fm.setLength(U16.t(OFFlowMod.MINIMUM_LENGTH+OFActionOutput.MINIMUM_LENGTH));
```

Opción de Elección de controlador Nº3: Floodlight (Java)

Floodlight es un controlador, basado en Java, de tipo empresarial que cuenta con licencia Apache. Es soportado por una comunidad de desarrolladores de todo tipo de backgrounds. [24]

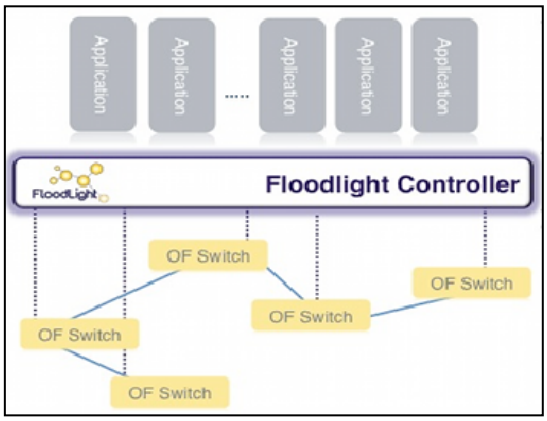

Figura 18: Controlador Floodlight de Java

Floodlight está diseñado para trabajar con una cantidad creciente de switches, routers, switches virtuales y puntos de acceso (APs) que soporten el estándar OpenFlow.

Pre-requisitos de Instalación:

Los requerimientos indicados, son los básicos establecidos según la web de Floodlight, donde en primera instancia es indispensable verificar la conectividad y que cada una

de las interfaces Ethernet en la máquina virtual en cuestión posea una IP asignada vía DHCP.

```
ping -c1 www.stanford.edu
ifconfig -a
sudo dhclient ethX
```

Una vez establecida la información relacionada a las interfaces y sus direcciones, se actualizan las fuentes y se instalan los pre-requisitos:

```
sudo apt-get update
time sudo apt-get install build-essential default-jdk ant python-dev
```

Instalación y ejecución:

Floodlight es muy simple de descargar (a continuación el código correspondiente a la descarga de la última versión) y si se asume la utilización de java se puede descargar directamente.

```
$ git clone git://github.com/floodlight/floodlight.git
$ cd floodlight
$ git checkout stable
$ ant;
$ java -jar target/floodlight.jar
```

Establecimiento de eclipse:

Es posible instalar, preparar y ejecutar floodlight a través de Eclipse. En lugar de configurarlo de forma manual.

```
$ ant eclipse;
```

El comando crea varios archivos: Floodlight.launch, Floodlight_junit.launch, .classpath, and .project , de los cuales se puede configurar un Nuevo proyecto de tipo Eclipse, como se muestra a continuación:

- Apertura de Eclipse y creación de un Nuevo workspace
- File -> Import -> General -> Existing Projects into Workspace. Se elige “Next”.

- Del "Select root directory"se hace click en "Browse". Se selecciona el directorio padre donde se ubicó floodlighter originalmente.
- Elegir la caja de "Floodlight"
- Hacer Click en "Finish"

Simulación de una Red:

Una vez que se tiene Floodlight en ejecución, se necesitan enlazarlo a una red OpenFlow, una vez más se utilizará como base , el programa Mininet; para ejecutarlo a través de un controlador propio se siguen los siguientes comandos (donde también se tiene la opción de a través de wireshark monitorear y filtrar paquetes.

```
$ sudo mn --controller=remote --ip=<controller ip> --
port=<openFlowPort 6633 by default>
$ ssh -Y openflow@<vm-ip>
$ sudo wireshark &;
```

Opción de Elección de controlador Nº4: Trema (Ruby):

Trema es un controlador de OpenFlow basado en Ruby que también proporciona entornos de prueba. Se presenta como una opción completa y fácil de usar para ingenieros y diseñadores no solamente para Ruby sino con también soporte para C. [25]

Pre-requisitos de Instalación:

Los requerimientos indicados, son los básicos establecidos para controladores anteriores y en conformidad con al web de Trema donde en primera instancia es indispensable verificar la conectividad y que cada una de las interfaces Ethernet en la máquina virtual en cuestión posea una IP asignada vía DHCP.

```
ping -c1 www.stanford.edu
ifconfig -a
sudo dhclient ethX
```

Una vez establecida la información relacionada a las interfaces y sus direcciones, se actualizan las fuentes y se instalan los pre-requisitos:

```
sudo apt-get update
sudo apt-get install rubygems1.8
```

Instalación y Ejecución:

La instalación es simple, se ejecuta gem y se instala Trema e inmediatamente se tendrá el ambiente de trabajo, existe la opción de instalación manual.

```
$ gem install trema
```

Instalación Manual:

```
$ tar xzvf trema-xyz.tar.gz
$ ./trema-xyz/build.rb
```

Creación del controlador:

Para escribir el controlador OpenFlow Trema, el procedimientoe nstá absado en simples scripts Ruby. Se configura y escribe el controlador OpenFlow añadiendo manipuladores de mensajes a la clase del controlador justo como en Rails.

```
Class MyController< Controller
 def packet_in dpid, message  # packet_in message handler
 …
 end
 def features_reply dpid, # features reply message handler
 …
 end
…
end
```

Ejecución del Controlador:

Es posible probar el controlador sin necesidad de compilación

```
$ ./trema run mycontroller.rb
```

Este simple comando de trema permite el bucle de: “codificar, ejecutar y depurar”.

Para estructurar la topología de la red a emular, con Trema network DSL, es posible describirla y configurarla para posteriormente ejecutarla.

```
# One virtual switch + two virtual hosts.
vswitch { dpid "0xabc" }
vhost “host1”
vhost “host 2”
link "0xabc","host1"
link "0xabc","host2"
./trema run mycontroller.rb -c network.conf
```

Opción de Elección de controlador Nº5: POX (Python)

POX una plataforma controladora de OpenFlow basada en Python.Puede considerarse como "el hermano menor" de NOX. En su núcleo se considera una plataforma de rápido desarrollo y creación de prototipos de software de control de red que utiliza el lenguaje Python; actualmente POX representa uno de los principales frameworks en boga (incluyendo NOX, Floodlight, Trema, entre otros) cuyo objetivo principal es configurar controladores OpenFlow. POX se encuentra en una fase de desarrollo activo y constante. Su objetivo principal es en los campos de investigación a través de pequeños proyectos. [26]

Características principales de POX:

- Interfaz OpenFlow basada en Python
- Componentes de muestra reutilizables para selección de camino, topología, descubrimiento, etc.
- Tiene como objetivo poder ser compatible tanto con Linux y MAC como con Windows.
- Soporta la misma GUI y las mismas herramientas de visualización que NOX
- Tiene un buen rendimiento si se compara con las aplicaciones de NOX escritas en Python.

Pre-requisitos de instalación:

Debe clonarse el último código (en función de las diferentes distribuciones disponibles en la web dentro de la maquina virtual:

```
git clone http://github.com/noxrepo/pox
cd pox
sudo./pox.py forwarding.l2_learning
```

Posterior al paso indicado, Mininet debe conectarse a POX y debe ser capaz de verificar la conectividad mediante los comandos de "ping" y "pingall".

Prueba del Controlador:

Una vez elegida la opción de controlador que mejor se adapte a nuestros requerimientos, es necesario verificar y probar su funcionamiento. Para ejecutar pruebas del controlador basado en un switch Ethernet, primero es esencial constatar que cuando todos los paquetes lleguen al controlador, solo los paquetes de broadcast (como los ARPs) y los paquetes con localidad de destino desconocida saldrán todos por puertos de no-entrada. Es posible hacer esto con tcpdump running o con un xterm para cada host.

Cuando ya el switch no tiene comportamiento de hub, debe trabajarse para bajar un flujo cuando se conocen los puertos de fuente y destino. Se puede utilizar dpctl para verificar los contadores de flujo, y si los pings subsecuentes se completan rápidamente, se sabrá que estos no están pasando a través del controlador. También se puede verificar este comportamiento al ejecutar iperf en Mininet y verificar que no se están transmitiendo mensajes OpenFlow packet-in. El ancho de banda reportado por iperf debe ser mucho más alto y debe coincidir el número el número de referencia con el que se conocía del controlador de learning switch como se explicó en secciones anteriores.

Expansión de Escenarios:

Entornos Multi-Switches:

Hasta ahora se ha ejemplificado el objetivo global del trabajo, ver una pequeña red y su configuración a través del protocolo OpenFlow; más sin embargo esto no involucra que no sea posible aspirar a más altas expectativas y es posible expandir el escenario planteado a switches múltiples.

Se inicia Mininet, en este caso con una topología diferente a la que hemos planteado desde el inicio (a múltiples switches) en conjunto con la opción ip (en caso que se utilice un controlador de tipo Beacon):

```
mininet> exit
$ sudo mn --topo linear --switch ovsk --controller remote
mininet> exit
$ sudo mn --topo linear --switch ovsk --controller remote --ip=<your_host_ip>
```

La Topología que se ha creado, corresponde a una estructura de 2 switches donde cada uno de los mismos tiene conectado un host. ,puede observarse en la siguiente Figura :

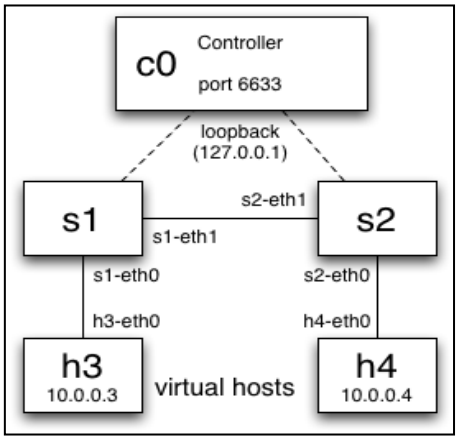

Figura 19: Escenario para prueba con entorno multi-switch

Posteriormente a la creación de la topología, debe modificarse el switch de manera que almacene una tabla de MAC-a-puerto para cada DPID. Esta estrategia solo trabaja en red de spanning tree, el retraso por establecer nuevos caminos entre hosts separados por largas distancias es proporcional al número de switch que hay de salto entre ellos. A través del siguiente comando se verificará si el controlador funciona:

```
mininet> pingall
```

Creación de un Router:

Para esta prueba, se construirá un switch/transmisor estático de capa 3; el cual no es exactamente un router, porque no decrementará el TTL de la IP y recalculará el checksum de cada salto. Es importante comentar que las acciones para ejecutar TTL y modificación del checksum se preveen para el OpenFlow versión 1.1; sin embargo puede considerarse un router porque permite coincidencia en función de los rangos IPs con máscaras establecidas, como un router normal.

Un router IP puede distinguirse de otros dispositivos de conmutación de paquetes en que el router examina la cabecera del protocolo IP como parte del proceso de conmutación. Generalmente remueve la cabecera de la capa de enlace, modifica la IP de la cabecera y sustituye la cabecera de la capa de enlace para re-transmisión. [27]

En esta opción de creación de router, se construirá un router completamente estático, sin capacidad de BGP o OSPF. Cada nodo de red tendrá una subred configurada; si un paquete está destinado a un host dentro de esa subred, el nodo actúa como un switch y transmite el paquete sin cambios a un puerto conocido o broadcast. Si un paquete está destinado a alguna dirección IP para la cual el router conoce el siguiente salto, debe modificarse el destino de capa 2 y transmitir el paquete al puerto correcto.

En esta recomendación de tipo "siguiente paso" dentro de las aplicaciones de OpenFlow, se aspira mostrar que con un dispositivo con OpenFlow habilitado la red es eficiente independientemente del estatus de las "capas"; es posible mezclas funcionalidades de switch, router y otras de altos niveles.

Topología:

Se plantea una topología de bastante más complejidad que en los ejercicios de la red a pequeña escala base de este Trabajo; se expone la estructura en la figura a continuación . Se requiere sea descrita en Mininet para su procesamiento.

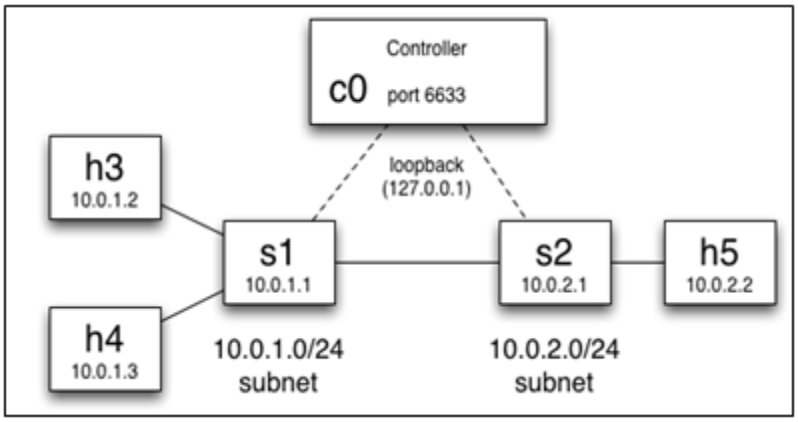

Ilustración 20: Escenario multi-switch extendido

A continuación se anexarán los comandos pertinentes al establecimiento en Mininet, en ejecución de una topología traicional y posteriormente buscar que coincida con la planteada en el ejercicio.; una vez establecida se debe verificar la conectividad entre todos los hostsa través de ping.

```
~/mininet/custom/topo-2sw-2host.py
$ cp ~/mininet/custom/topo-2sw-2host.py mytopo.py
$ sudo mn --custom mytopo.py --topo mytopo --mac
mininet> pingall
```

Establecimientos de Hosts:

Es necesario establecer la configuración IP en cada host virtual para obligarlos a transmitir al Gateway las IPS con destinos fuera de la subred configurada; es también requerido configurar cada host con una IP de subred, un Gateway y una máscara de red; para las interfaces correspondientes a los switches s1 y s2 el tráfico "desde" y "hacia" debe configurarse a través de OpenFlow, existen varias formas de hacer esto:

Comandos Unix via Mininet CLI:

Dentro del Mininet CLI, es posible transmitir commandos a los hosts para configurarlos:

```
mininet> h2 ifconfig h2-eth0 10.0.1.2/24
mininet> h2 route add default gw 10.0.1.1
mininet> h2 route -n
```

Comandos Unix regulares a través de un xterm:

Otra forma alternativa para el establecimiento de los hosts es a través de un xterm.

```
mininet> xterm h2
# ifconfig h2-eth0 10.0.1.2/24
```

Ejecutar un script de configuración:

Es posible ejecutar scripts en los propios hosts, por ejemplo si se tienen un script llamado "config_script" ,puede ser ejecutado en el host h2 desde el Mininet CLI:

```
mininet> h2 config_script
```

Con este script, es posible tener la información previamente guardada y configurada (por ejemplo basados en una dirección MAC es posible asignar un número de nodo) o tambiñen es posible ir añadiendo parámetros deseados.

Secuencia de Comandos CLI:

Para evitar la repetición de secuencias de commandos CLI, es posible crear un archive que contenga múltiples comandos CLI y luego ejecutar esa serie de comandos en Mininet:

```
mininet> source my_cli_script
```

o

```
# mn --pre my_cli_script
```

Un script CLI básico se ve algo así:

```
py "Configuring network"
h3 ifconfig h3-eth0 10.0.1.2/24
h4 ifconfig h4-eth0 10.0.1.3/24
h5 ifconfig h5-eth0 10.0.2.2/24
h3 route add default gw 10.0.1.1
h4 route add default gw 10.0.1.1
h5 route add default gw 10.0.2.1
py "Current network:"
net
dump
```

La secuencia de commandos CLI está diseñada para evitar exceso de comandos más no es un entorno de programación como tal; para tareas avanzadas, la API de Mininet propocionada acceso flexible a todas las capacidades de mininet.

La API de Mininet de Python proporciona una maneraeficiente de crear y configurar redes de forma programática.

ARP:

Un router generalmente debe responder a solicitudes ARP. Es común ver broadcasts de Ethernet que inicialmente son transmitidos al controlador. El controlador posiblemente deba construir respuestas ARP y transmitirlas a los puertos correspondientes , también e posible evitar tratar con ARP al añadir entradas de ARP estáticas en cada host de todos los nodos de su subred.

Routing Estático:

Una vez que se ha resuelto el ARP, es necesario resolver el enrutamiento de configuraciones estáticas, es posible buscar que coincidan las direcciones IP y remitir al puerto correspondiente.

Para el caso ejemplo, para el switch S1 se desean remitir paquetes destinados aun host anexo al puerto apropiado, mientras que los paquetes que tienen como destino la subred remota deben ser remitidos al switch S2.

ICMP:

Adicionalmente, el controlador puede recibir solicitudes eco ICMP (ping) para cada switch, a las cuales debería responder. Por último, los paquetes para subredes inalcanzables deben responder con mensajes de red ICMP inalcanzables.

Apendice B:

Instalación de VNX:

Se asume se cuenta con una distribucion Linux moderna (se recomienda de 10.04 en adelante) y un procesador con soporte de virtualización; es posible comprobar que se cuenta con dicho soporte utilizando el comando kvm-ok:

```
#kvm-ok
INFO: Your CPU supports KVM extensions
INFO: /dev/kvm exists
KVM acceleration can be used
```

Manualmente puede verificar a través del siguiente comando:

```
egrep '(vmx|svm)' --color=always /proc/cpuinfo
```

Si se visualizan las palabras vmx (para procesadores de fabricante Intel) o svm (para procesadores de fabricante ADM) en color rojo, el procesador cuenta con soporte de virtualización.

Un aspecto importante a tomar en cuenta es que debe revisarse que las extensiones de virtualización son controladas desde la BIOS. Es posible tener acceso a la configuración de la BIO y examinar si el soporte de virtualización se encuentra habilitado de no ser así posiblemente se obtenga el siguiente mensaje:

```
egrep '(vmx|svm)' --color=always /proc/cpuinfo
FATAL: Error inserting kvm_intel (...): Operation not supported
```

Bibliografía:

[1] P. Gelsinger. Keynote at Citrix Synergy. Mayo, 2009

[2] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, Rexford, J. Shenker Scott, J . Turner . “OpenFlow: Enabling Innvotacion in Campus Networks”. Marzo, 2008

[3] Global Environment for Network Innovations. Website: http://geni.net.

[4] NetFPGA: Programmable Networking Hardware. Website: http://netfpga.org.

[5] S. Microsystems y Vmware."The Business Benefits of Virtualization and Consolidation with Sun and Vmware". 2008

[6] Ben Pfaff, et al., Extending Networking into the Virtualization Layer. 2009

[7] A. Mendoza, Utility Computing Technologies, Standards, and Strategies. United States of America: Artech House, Inc., 2007

[8] A. di Constanzo, M. Assuncao R. Buyya, "Building a Virtualized Distributed Computing Infrastructure by Harnessing Grid and Cloud Technologies". 2009

[9] R. Moskowitz,P. Nikander,P. Jokela. RFC 5201: Host identity protocol. Abril,2008

[10] M. Casado, MJ. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker. "Ethane: Taking Control of the Enterprise," ACM SIGCOMM '07, August 2007, Kyoto, Japan

[11] http://www.ibm.com/us/en/

[12] HP Switch Software, OpenFlow Supplement, Software version K.15.05.5001. November 2011.

[13] Open vSwitch http://openvswitch.org

[14] http://web.dit.upm.es/vnxwiki/index.php/Main_Page

[15] http://web.mit.edu/newsoffice/2012/mobile-secure-shell-0628.html

[16] http://www.xfree86.org/4.0.1/xterm.1.html

[17] http://www.wireshark.org/about.html

[18] http://seguridadyredes.wordpress.com/

[19] http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet

[20] http://sourceforge.net/apps/trac/cbench/

[21] http://www.openflow.org/wk/index.php/OpenFlowMPLS

[22] http://www.noxrepo.org/

[23] http://www.python.org/

[24] https://openflow.stanford.edu/display/Beacon/Home

[25] http://floodlight.openflowhub.org/

[26] http://trema.github.com/trema/

[27] http://www.noxrepo.org/pox/about-pox/

[28] F. Baker, RFC 1812: Requirements for IP Version 4 Routers. Cisco Systems, March 1995