MONITORIZACION DE EVENTOS ESPECIFICOS EN REDES DEFINIDAS POR SOFTWARE

Edy Alberto Guerrero Garzon
Franc Edicson Salamanca Lopez

Director: Ing. Luis Alberto Eraso Portilla

Fundación Universitaria de Popayán
Programa de Ingeniería de Sistemas
Línea de Investigación en Sistemas Telemáticos Inteligentes

Popayán, Septiembre de 2018

MONITORIZACION DE EVENTOS ESPECIFICOS EN REDES DEFINIDAS POR SOFTWARE

Monografía de grado para optar al título de:

INGENIERO DE SISTEMAS

Edy Alberto Guerrero Garzón
Franc Edicson Salamanca López

Director: Ing. Luis Alberto Eraso Portilla

Fundación Universitaria de Popayán
Programa de Ingeniería de Sistemas
Línea de Investigación en Sistemas Telemáticos Inteligentes
Popayán, Septiembre de 2018

Agradecimientos

En el marco de la terminación de esta gran etapa universitaria agradezco en primer lugar a Dios, por permitirme haber podido llegar hasta este punto y lograr un objetivo más en mi vida, a la Fundación Universitaria de Popayán en cabeza de sus docentes y equipo administrativo que como institución me abrieron las puertas para crecer como profesional, mis más sinceros agradecimientos a mi director de tesis, Dr. Luis Eraso Portilla por su esfuerzo y dedicación, quien con sus conocimientos, experiencia, paciencia y su motivación ha logrado que pueda hacer realidad este sueño, por último y de forma muy especial darles mil gracias a mis padres, familiares y demás compañeros de carrera, que me apoyaron siempre en este proceso y fueron mi gran motivación para continuar con esta formación personal y profesional.

Edy Alberto Guerreo G.

Es difícil llegar al final de una etapa y tener las palabras adecuadas para poder agradecer a las personas correctas por la terminación de este ciclo. Durante este camino me he encontrado con todo tipo de personas que han hecho agradable el tiempo, pero también otras me han hecho recordar que la vida no es fácil, siempre encontraremos retos por superar, y cuando menos lo piensas te estancas. Por eso dar el más sincero agradecimiento a nuestro director de trabajo de grado el Ing. Luis Eraso Portilla por su paciencia y conocimiento, a mi compañero de tesis Edy ya que siempre fue un apoyo en el transcurso de la carrera, a mis padres porque siempre fueron esa motivación constante que me impulsaba para salir adelante. También quiero dar un agradecimiento especial a Yenny porque me animo e impulso a estudiar.

Franc Edicson Salamanca Lopez.

Resumen

El crecimiento exponencial que sufre la tecnología está lleno de grandes retos, uno de ellos y tal vez entre los más importantes es la forma en cómo se gestiona, se trasmite y se monitorea la información, por la cantidad tan inmensa de datos y dispositivos en conexión, las redes de computadoras se vuelven cada vez más complejas y difícil de administrar, por tal razón es pertinente adoptar nuevas estrategias en cuanto a infraestructura de red como la implementación de las SDN (Software Defined Networking), y protocolos de red OpenFlow, que buscan dar solución a muchos inconvenientes que presentan las redes tradicionales, por su capacidad de ser programables, permiten optimizar la capacidad de la red separando el plano de datos del plano de control y centralizando el control de la red en un solo controlador SDN, asi mismo permiten simplicidad y mejora en los procesos, innovaciones en la trasmisión y gestión de datos con total flexibilidad. [27]

No obstante aun siendo las SDN tecnologías más efectivas, y con buenas ventajas de escalabilidad, presentan más interrogantes y paradojas al momento de su gestión y monitoreo de sus procesos; por tal razón este trabajo busca como objetivo principal mostrar a partir de una simulación de red SDN, con componentes reales virtualizados bajo la plataforma MININET y uso del protocolo Open Flow, la implementación de una función de monitorización (algoritmo) que permite medir el retraso oscilante de un componente detallado dentro de la red denominado JITTER, para buscar alternativas de mejora y rendimiento de esta.

Las configuraciones están realizadas en un entorno totalmente virtual y programable y la implementación de la función y medición se hace bajo la herramienta Wireshark.

Palabras clave: SDN, OpenFlow, Monitorización, MiniNet, Jitter.

Abstract

The exponential growth suffered by technology is full of great challenges, one of them and perhaps among the most important is the way how information is managed, transmitted and monitored, by the immense amount of data and devices in connection , computer networks are becoming increasingly complex and difficult to administer, for this reason it is pertinent to adopt new strategies in terms of network infrastructure such as the implementation of SDN (Software Defined Networking), and OpenFlow network protocols, which seek To solve many inconveniences that traditional networks have, the SDN's due to their ability to be programmable, they allow to optimize the capacity of the network by separating the data plane from the control plane and centralizing the control of the network in a single SDN controller. simplicity and improvement in processes, innovations in transmission and data management with total flexibility. [27]

However, even though the SDN technologies are the most effective, and with good scalability advantages, they present more questions and paradoxes when managing and monitoring their processes; for this reason this work seeks as a main objective to show from an SDN network simulation, with real virtualized components under the MININET platform and use of the Open Flow protocol, the implementation of a monitoring function (algorithm) that allows to measure the oscillating delay of a detailed component within the network called JITTER, to look for alternatives for improvement and performance of this.

However, even though the SDN technologies are the most effective, and with good scalability advantages, they present more questions and paradoxes when managing and monitoring their processes; for this reason this work seeks as a main objective to show from an SDN network simulation, with real virtualized components under the MININET platform and use of the Open Flow protocol, the implementation of a monitoring function (algorithm) that allows to measure the oscillating delay of a detailed component within the network called JITTER, to look for alternatives for improvement and performance. The configurations are made in a totally virtual and programmable environment and the implementation of the function and measurement is done under the Wireshark tool.

Key Words : *SDN, OpenFlow, Monitoring, MiniNet, Jitter.*

Contenido

CAPÍTULO 1. INTRODUCCION 11
1.1. Descripción del problema 13
1.2. Justificación 16
1.3. Objetivos 17
1.3.1. Objetivo general 17
1.3.2. Objetivos específicos 17
1.4. Aportes del trabajo 18
CAPÍTULO 2 ESTADO DEL ARTE 19
2.1. Marco teórico. 19
2.1.1. Redes de computadoras 19
2.1.2. Redes definidas por software 19
2.1.3. Gestión de redes 22
2.1.4. Virtualización de Redes 23
2.1.5. Monitorización de redes 24
2.2. Trabajos relacionados 25
2.2.1. A Network Monitor and Controller using Only OpenFlow 25
2.2.2. Decentralized monitoring for large-scale Software-Defined Networks 26
2.2.3. Hardware Modules for Packet Interarrival Time Monitoring for Software Defined Measurements 27
2.2.4. An efficient flow monitoring algorithm using a flexible match structure 28
2.2.5. A monitoring framework for 5G service deployments 29
2.2.6. Interactive monitoring, visualization, and configuration of OpenFlow-based SDN 30
2.2.7. Adaptive flow monitoring in SDN architecture 31
2.2.8. Outsourcing the Routing Control Logic: Better Internet Routing Based on SDN Principles 32
CAPÍTULO 3. ARQUITECTURA DE UNA RED SDN 34
3.1. Emuladores y Simuladores de Red SDN 35
3.1.1. MiniNet 37
3.1.2. Xming 40
3.1.3. PuTTY 41

3.2. OpenFlow 42
3.2.1. Arquitectura OpenFlow 43
3.2.2. Switch OpenFlow 44
3.3. Controladores SDN 45
3.4. Entornos de análisis y monitoreo en redes SDN 49
3.4.1. OpenNetMon 49
3.4.2. Wireshark 50
3.5. Jitter 53
3.5.1. Como Calcular el JItter 57
CAPÍTULO 4. INSTALACION DE LA RED SDN 59
4.1. Descripción de la red 59
4.1.1. Mapa de la red 59
4.1.2. Configuración de la red 60
4.1.3. Esquema de direcciones ip 60
4.2. Configuración de entornos SDN 60
4.2.1. Software utilizado 60
4.2.2. Configuración máquina virtual 61
4.2.3. Entorno de simulación 64
4.3. Topología de red 66
CAPÍTULO 5. MONITOREO DE UNA RED SDN 72
5.1. Monitorización de la red 72
5.1.1. Desplegando controlador SDN POX 73
5.2. Desplegando OpenNetMon 76
5.2.1. Instalación de OpenNetMont 76
5.2.2. Corriendo OpenNetMon 77
5.3. Monitorización del Jitter 78
5.3.1. Diagrama de flujo 78
5.3.2. Medición del Jitter 79
5.3.3. Evaluación de la función 79
CAPÍTULO 6. CONCLUSIONES 81
Referencias 82

Lista de Figuras

Figura 1. Arquitectura de redes SDN [47] 20
Figura 2. Virtualización de red. .[48]..... 23
Figura 3. Arquitectura SDN (SDN and Openflow for beginners)..... 34
Figura 4. Topología red virtual vs Topología de física 55]..... 36
Figura 5. Comando para habilitar mininet 38
Figura 6. Comando para crear una topología de red..... 39
Figura 7. Arquitectura de topología básica en mininet 40
Figura 8. Arquitectura de red Openflow 43
Figura 9. Composición y estados de la tabla de flujo 44
Figura 10. Escenario de simulación de red SDN con controlador POX..... 47
Figura 12. Captura de paquetes Wireshark..... 51
Figura 11. Filtros de captura Wireshark 51
Figura 13. Gráfico de trafico opción I/O graphs Wireshark..... 52
Figura 14. Diferencia absoluta posición del borde reloj o señales..... 54
Figura 15. Efecto Jitter de 2.48832 GHz, en señal de reloj – Representación del dominio del tiempo (Hewlett Packard Journal – Copiright 1995)..... 54
Figura 16. Comparación trafico uniforme de paquetes *con la fluctuación del retardo de llegada (Jitter)*..... 55
Figura 17. Árbol de tipos de Jitter 56
Figura 18. Diagrama de red..... 59
Figura 19. Máquina virtual con mininet 62
Figura 20.Inicio de sesión Ubuntu 14.04..... 62
Figura 21. Configuración de red Virtual Box..... 63
Figura 22. Dirección ip SO Ubuntu (cliente mininet)..... 64
Figura 23. Configuración PuTTY..... 65
Figura 24. Cliente PuTTY X11 65
Figura 25. Entorno de simulación 66
Figura 26. OpenNetMon 66
Figura 27. OpenNetmon núcleo..... 66
Figura 28. Topología implementada..... 67
Figura 29. Nodos creados..... 68
Figura 30.Configuración h1, ifconfig..... 68
Figura 31. Enviando paquetes desde h1 a h2..... 68
Figura 32. Tabla de flujo vacía..... 69
Figura 33.Habilitando flujo entre h1 y h2..... 69
Figura 34. Haciendo ping h1 a h2 con puertos abiertos 70
Figura 35. Tabla de flujo 70
Figura 36. Analizando tráfico a través de wireshark..... 71
Figura 37. Verificación de procesos..... 73
Figura 38. Verificando instalación de POX..... 73
Figura 39. Ingresando carpeta POX..... 74
Figura 40. Desplegando controlador POX 74
Figura 41. Abriendo ventanas para cada host..... 74

Figura 42. Haciendo ping desde host 1 a host 2 75
Figura 43. Comprobando ping en el host 2 76
Figura 44. Creando directorio OpenNetmon 76
Figura 45. Verificación de carpeta 77
Figura 46. Clonando desde repositorio 77
Figura 47. OpenNetMon desplegado 77
Figura 48. Diagrama de flujo para medida del jitter basado en diagrama de tesis [85]..... 78
Figura 49. Monitorización de Jitter 79
Figura 50. Diagrama de medición y validación de datos Jitter 80

Lista de Tablas

Tabla 1. Controladores SDN y sus características basadas en tabla del artículo [51] 46
Tabla 2. Componentes del controlador POX basados en tablas de artículos [50] [58] 49
Tabla 3. Tipos de Jitter 57
Tabla 4. Esquema de direcciones ip 60
Tabla 5. Software utilizado 61
Tabla 6. Topologias 67

CAPÍTULO 1. INTRODUCCION

Actualmente, las redes definidas por software (SDN) son parte de un escenario de infraestructura dinámica que controlan recursos informáticos y de red. A pesar de encontrarse en un estado de desarrollo, su rápido crecimiento y la adopción del protocolo OpenFlow han hecho que se rompa un paradigma y se llegue a una transición en la cual se pretende pasar de grandes y caros equipos con firmware propietario a equipos genéricos, que simplemente conmutarán señales, interpretarán mensajes OpenFlow y gestionarán información dedicada en sus tablas de flujo con muchísima más flexibilidad y eficiencia que la de antes, es tal el auge que muchas de las compañias creadoras de software y empresas dedicadas a soluciones de virtualización y computación en la nube han desarrollado sus propias soluciones o adoptado el estándar OpenFlow para redes SDN.[39]

El surgimiento de SDN (Software Defined Networking) está dado bajo los parámetros de mejora, facilitar la programabilidad de la red y facilitar las tareas de gestión son su principal objetivo, así pues SDN propone separar y desacoplar el plano de control del plano de datos,[45] se fundamenta en que la funcionalidad del plano de datos y reenvío de paquetes está integrada a la red de conmutación, mientras que la funcionalidad del plano de control que esta para controlar dispositivos de red se coloca dentro de un componente de software lógicamente centralizado denominado controlador, este plano brinda una interfaz programable para desarrollar aplicaciones de gestión, en lugar de proporcionar una interfaz de configurable para adoptar propiedades de la red. [51]

Desde el punto de vista de la administración, esta programabilidad agregada abre la oportunidad de reducir la complejidad de la configuración distribuida y facilitar las tareas de administración de red [45], sin embargo el estado de la red cambia continuamente y este comportamiento dinámico presenta desafíos, uno de ellos y entre los más importantes es Monitorear para mantener el estado actualizado de los recursos, ese orden es esencial, ya que es el primer paso hacia la tarea posterior de controlar la red. Con este enfoque y algunos detalles de configuración y administración, SDN proporciona una mayor flexibilidad además de otros beneficios [4], SDN ayuda a monitorear y controlar la red porque proporciona un controlador de software central con una vista general de la red lo que realmente genera muchas ventajas pero aun así no deja de ser un desafío. [39]

El monitoreo de red es tan importante en la administración de una red, porque las aplicaciones de administración requieren estadísticas precisas y oportunas sobre los recursos de red en diferentes niveles de agregación [51], además, conocer el estado de la red ayuda a decidir, planificar y usar los recursos con la calidad

adecuada. La monitorización a corto, mediano o largo plazo, ayuda a recopilar datos sobre las condiciones de la red.

Con la implementación de la arquitectura SDN, de herramientas de software que emulan el estado de una red y las capacidades de OpenFlow se pueden desarrollar soluciones de monitoreo sofisticadas y efectivas que contrarresten los problemas de la red. Estas redes se caracterizan por su gran escala y diversidad del tráfico generado. Pero, en estas redes la monitorización no es una tarea fácil debido a que es necesario realizar el desarrollo de aplicaciones para medir un evento en específico (jitter, delay, sobrecarga, perdida de paquetes. etc.) dependiendo de la necesidad que se presente en la red. Además, es una tecnología relativamente nueva lo que implica que aún es objeto de investigación. Por esta razón en este documento se pretende crear una solución de monitoreo basada en una función algorítmica de medida que monitoree y recopile información de un evento o fluctuación especifica que se presenta en la red (Retardo entre llegada de paquetes) denominado Jitter.

La estructura del documento está dividida en secciones centradas todas hacia el mismo objetivo que es la monitorización en una red SDN, El desarrollo metodológico del proyecto investigativo se fundamenta en una revisión sistemática de la literatura con levantamiento de información precisa y estructurada en etapas de la siguiente forma. En la primera y segunda se describen brevemente los conceptos y componentes que conforman la investigación, y las bases para el desarrollo del proyecto, en la tercera se hace un muestreo de la herramientas adecuadas para el desarrollo de la investigación, en la cuarta evidencia el montaje de la red SDN virtualizada y funcional donde se medirán los parámetros y en la quinta se hara la implementación de la función de monitoreo programada en Phyton y guiada por el controlador Pox que recoge datos precisos y mediciones del Jitter, finalmente como resultado saldrán las conclusiones de la investigación y los aportes para posibles implementaciones a futuro.

1.1. Descripción del problema

El crecimiento considerable de conexiones a internet en los últimos años, concretamente 3.174 millones de usuarios conectados en el año 2015 0. Así como el auge de nuevas tecnologías como el internet de las cosas (IoT) sobre el cual se estima que 10 billones de objetos estén conectados para el año 2020 [11]. Han llevado a un aumento considerable en el tráfico de red, lo cual implica un conjunto de nuevos retos a afrontar en el diseño, implementación y gestión de las redes de computadoras.

Entre los principales retos a afrontar Kreutz et al [9] describe: i) la complejidad de configuración de los dispositivos de red; esto es, para implementar las políticas de configuración de la red, los operadores necesitan configurar cada dispositivo de forma individual usando comandos de bajo nivel y muchas veces específicos del proveedor [5]. ii) La capacidad de respuesta a fallas, sobrecargas de tráfico y cambios en la red; esto es, la red debe garantizar niveles de disponibilidad y elasticidad a través de la configuración de rutas alternativas en caso de fallas; asimismo, se deben garantizar el desempeño y la escalabilidad asignando y reservando el ancho de banda suficiente, además de configurar y ejecutar políticas de calidad del servicio (QoS) [6]. Y finalmente iii) la capacidad de la red para reconfigurarse de forma autónoma; es decir, la capacidad de la red para detectar cambios en cualquiera de sus componentes o en su entorno los cuales puedan tener o resultar en una violación de los objetivos de gestión y de esta manera activar la configuración adecuada, o más precisamente, los mecanismos de adaptación sin perturbar el rendimiento de la red [9].

Como una alternativa para alcanzar estos retos, enfoques de redes basados en la virtualización de componentes y la abstracción de funcionalidades se han venido trabajando en los últimos años; concretamente las Redes Definidas por Software (SDN) y la Virtualización de Funciones de Red (NFV). Las SDN tienen como base fundamental de su arquitectura la separación en tres planos horizontales de funcionalidades: el plano de datos, el plano de control y el plano de aplicación [11]. Además de un plano vertical para la gestión el cual se incluye en algunos trabajos y especificaciones recientes [11-14]. Mientras que la idea principal de las NFV es desacoplar los dispositivos físicos de red de las funciones que se ejecutan sobre estos [16]. Aunque las SDN y NFV son altamente complementarias, éstos son conceptos diferentes, dirigidos a abordar diferentes aspectos de una solución de red basada en software. Sin embargo, es posible combinarlos en una solución de red que pueda conducir a un mayor valor [16].

A pesar de los beneficios que ofrecen las SDN; aún existen cuestiones acerca de cómo gestionar dichas redes. En [9, 16,17] se describen las principales

problemáticas en lo que respecta a la gestión, haciendo una comparación de cómo éstas se presentan en las redes tradicionales y en las SDN. En resumen, las principales cuestiones son: el arranque y configuración de la red; disponibilidad y elasticidad; programabilidad; rendimiento y escalabilidad, aislamiento y seguridad; flexibilidad y desacoplamiento, planeación de la red; y finalmente monitorización y virtualización. [25]

En lo que respecta a la monitorización de red, las redes tradicionales con respecto a las SDN se diferencian principalmente porque las primeras se basan en el rastreo de utilización de recursos, identificación de interrupciones y activación de alarmas, además las soluciones de monitorización requieren una instalación y configuración de software y herramientas de bajo nivel en cada dispositivo monitorizado [10]. Mientras que las SDN han introducido un conjunto de herramientas simples y reutilizables para la recopilación de estadísticas de red en diferentes niveles de granularidad, lo cual las hace adecuadas para una amplia gama de tareas dentro de la gestión, incluyendo la monitorización de red. Así mismo, las SDN se encargan también de localizar parámetros funcionales de aplicaciones novedosas y visualizar la zona o el dominio de los controladores de red. [20 - 10]

En las SDN la parte física del monitoreo y visualización es muy similar a las redes tradicionales pero la parte lógica es más compleja ya que esta puede ser completamente rediseñada bajo un requisito diferente [10]. Es así como, los routers basados en flujo SDN (ejemplo, OpenFlow) permiten a los operadores de red especificar de manera flexible los flujos a monitorear basándose en diferentes campos de paquetes (ejemplo, direcciones IP de origen o destino) y contar el número de bytes o paquetes para estos flujos. [20].

En las redes IP actuales, es fácil dibujar un mapa de alcance analizando la información de enrutamiento y las tablas de flujo, incluso antes de que el tráfico comience a fluir esto sólo es posible porque el reenvío y el comportamiento de los routers es predecibles. Por el contrario, para las SDN, cuando la implementación interna de un protocolo no se conoce, o ha sido definida por el software, estas tareas se vuelven complejas.[20] Porque, en primer lugar, los contadores basados en flujo se mantienen en la TCAM (por sus siglas en ingles Ternary Content Addressable Memory; Memoria especializada de alto rendimiento utilizada para aumentar la velocidad en la rutas de búsqueda, reenvio de paquetes y las lista de control de acceso, comúnmente se encuentra en equipos de redes tales como enrutadores y conmutadores y su enfoque esta dado a la capacidad para almacenar y consultar datos utilizando tres entradas diferentes.[21]) relativamente costosos y registran un alto consumo de energía, como tal, solo se puede usar un número limitado de entradas para las mediciones. Otro problema es el ancho de banda limitado entre el router y el controlador SDN, que limita la obtención de flujo

a no más de unos pocos miles por segundo. Finalmente, los switches también pueden exhibir imprecisiones al actualizar los contadores de flujo. [20 - 22].
Al momento de monitorizar una red, los principales parámetros que por lo general se miden son; el flujo de datos, Jitter, retardo, ancho de banda, entre otros. En el caso del Jitter, este es particularmente importante porque la variación de retardo, puede causar problemas con la ejecución de aplicaciones debido a que los paquetes enviados de la misma trama no evitan rutas potencialmente inestables entre el transmisor y el receptor. [1].

Por otro lado, un factor limitante para el caso de las topologías de red SDN a gran escala, son las limitaciones de escalabilidad considerables, debido a la cantidad de tráfico y la carga de procesamiento que convergen a un único controlador/administrador [10]. Esto conlleva a que las prácticas de monitoreo sean más difíciles de implementar y no se cumpla a cabalidad con los objetivos propuestos para estas; aunque en la literatura se pueden encontrar varios trabajos que proponen funciones de monitoreo en topologías de redes y centros de datos a gran escala. Por otra parte no se ha encontrado mucha información y trabajos enfocados a dar una solución de monitoreo sobre redes SDN de tamaño reducido también conocidas como redes livianas.

Por lo tanto, con el tamaño y la estructura de las redes aumentando actualmente, los administradores de red a menudo se enfrentan a un constante diluvio de datos, que deben monitorizarse para comprender el estado de la red. Y con la complejidad e inconvenientes que tiene la monitorización de una red SDN surge el siguiente interrogante; ¿Es posible monitorizar el Jitter en una red SDN de tamaño liviano desarrollando una función de monitorización que permita reducir costos de operación para mejorar la flexibilidad y el rendimiento de esta?

1.2. Justificación

El presente proyecto está enfocado en la investigación, el análisis, desarrollo e implementación de una función de monitoreo que permita monitorizar el flujo de datos y el retardo entre paquetes denominado JITTER en una red SDN de tamaño liviano, haciendo uso de distintas herramientas que permitan realizar un diseño completo y funcional de una función de monitorización siguiendo una arquitectura de red flexible.

Por lo tanto, la importancia de este proyecto se basa principalmente en la ruptura del paradigma actual, en el cual las redes tradicionales generan una dependencia de uso existente en el campo de las redes, buscando ampliar la percepción de una red SDN y formularlo como la siguiente generación de las redes de datos actuales. Para lograr dar inicio a un nuevo camino donde las redes dinámicas y flexibles ofrezcan mayor usabilidad, mejoras y reducción de costos, promoviendo el uso de software libre.

Actualmente la estructura de las redes en la mayoría de instituciones no ha cambiado su enfoque tradicional, el esquema de red aun es descentralizado lo que impone un sistema de administración que no satisface las necesidades tanto para quien la administra como para el usuario. El trabajo de gestión e incluso de monitorización de los diferentes equipos es por separado y se tiene en cuenta únicamente un plano de control distribuido, que por supuesto no provee la flexibilidad, escalabilidad, y dinámica que las redes actuales requieren.

La monitorización de redes SDN emergen como una nueva solución para optimizar las tareas de administración de red, reduciendo los costos de hardware, mediante la automatización de procesos encadenados en la recepción, análisis y presentación de estadísticas recopiladas. La implementación de un adecuado sistema de monitoreo SDN mediante la virtualización de los centros de datos, incrementa en un gran porcentaje las ventajas de las SDN ya que aumentan la flexibilidad, la utilización de recursos, reducen los gastos generales y los costos de infraestructura.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una función de monitoreo para una red de datos liviana bajo la tecnología SDN mediante una simulación virtualizada.

1.3.2. Objetivos específicos

- Conocer el estado actual de las redes definidas por software (SDN), componentes y el impacto que generan en su arquitectura, mediante la implementación de una metodología de investigación enfocado a literatura.

- Proponer una función de monitoreo para una red con arquitectura SDN virtual la cual permita recopilar, analizar y clasificar información relacionada a los parámetros de “Jitter” de la red.

- Evaluar la función propuesta en términos de exactitud y optimización de recursos.

1.4. Aportes del trabajo

El alcance de del Proyecto incluye:

- Levantamiento de información sobre los parámetros específicos en las redes definidas por software (SDN), con estándares de investigación enfocados en monitoreo por medio de la revisión sistemática de la literatura.
    - Estructura metodológica para el desarrollo de la investigación

- Documento técnico de instalación, en donde se explica paso a paso la configuración del entorno virtual de la red SDN, herramientas y entornos de red en funcionamiento, y pruebas en software de monitorización.

- Función de monitoreo determinístico de un parámetro de red SDN Jitter, en modelo de código computacional.

- Estructura del artículo de investigación en borrador basada en la monografía para ser publicada en revista científica

CAPÍTULO 2 ESTADO DEL ARTE

2.1. Marco teórico.

2.1.1. Redes de computadoras

Una red de computadoras o red informática es un conjunto de equipos y/o dispositivos conectados por medio de cableado, señales, transmisiones o cualquier otro método de transporte de datos, para compartir información (archivos, colecciones), recursos (programas, discos, impresoras) y servicios (Internet, acceso a bases de datos, internet, Email, juegos, chat, etc.) [24].

Por consiguiente, la tarea principal de una red es contribuir para que los usuarios puedan tener acceso instantáneo a los recursos que se ofrecen desde las compañías, en cualquier momento y en cualquier lugar. [5] Dichos recursos no solamente incluyen los datos tradicionales, sino también de video, de voz entre otros. Las redes son aquella parte esencial que unen el mundo porque respaldan la forma en que se aprende, se trabaja y se comunica tanto así que con el correr de los años, las redes de datos se han expandido y transformado progresivamente para mejorar la calidad de vida de las personas en todo el mundo [35].

Existen redes de todo tamaño. Pueden ir desde redes simples, compuestas por dos computadores, hasta redes que conectan millones de dispositivos.[23] Dependiendo de la extensión, se pueden organizar en redes de área local (LAN), que son aquellas que conectan dispositivos u ordenadores en un área relativamente pequeña y predeterminada como una habitación, una oficina o hasta un edificio por medio de cables u ondas y también existen la redes de área extendida (WAN) cuyos parámetros están definidos para cubrir mayores distancias de comunicación entre 100 y 1000 kilómetros aproximados, lo que les permite brindar una conectividad a todos sus clientes en varias ciudades e incluso a un país entero por lo general estas utilizan sistemas de comunicación de radio, servicios de proveedores de internet (ISP por sus siglas en inglés), empresas de telefonía, cable e incluso satélites.[35]

2.1.2. Redes definidas por software

Las redes definidas por software (SDN) se basan en el paradigma de redes programables. Esto a través de una abstracción que permite la separación del plano de control del plano de datos mediante interfaces estandarizadas. El propósito principal de una red SDN es reducir la complejidad de la red y permitir el desarrollo de innovaciones más rápidamente tanto en el plano de control como en

el de datos. Ofreciendo de esta forma aplicaciones que controlan la red de forma programática [25].

Generalmente, una red SDN posee un modelo básico de arquitectura determinada en capas las cuales son administradas de manera lógica centralizada. una arquitectura en capas, se puede dividir en diferentes de ellas de tipo lógicas las cuales son; capa de aplicación, capa de control y capa de infraestructura [7]:

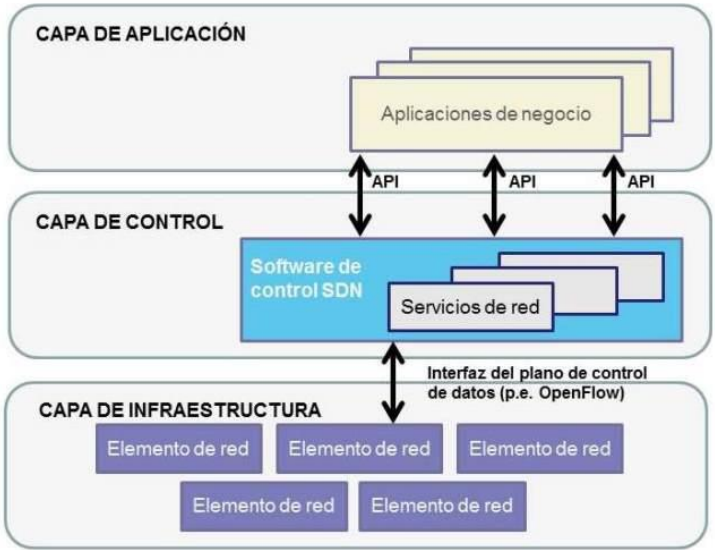


Figura 1. Arquitectura de redes SDN [47]

Capa de aplicación (Aplication Layer), que establece aplicaciones para automatizar tareas de configuración, provisión y despliegue de los servicios que se pretendan configurar. Esta capa se comunica con el controlador por medio de una API (*Application Programming Interface*), permite brindar una visión global de las condiciones actuales de la red, con el fin de mejorar la transmisión de datos.

Su importancia está ligada a mantener realizadas las operaciones de enrutamiento a nivel de administración, es decir las reglas propias en almacenamiento y aprovisionamiento de aplicaciones, que cumplen con el modelo de operaciones de los centros de datos, logrando mejorar la automatización, y paralelamente garantizar la seguridad de la red en tiempo real.

Capa de control (Control Layer), es la encargada de permitir la visión centralizada de la topología que representa cada porción lógica de la red de datos. Esto le permite a la capa siguiente, gestionar el flujo de datos, donde actores como el protocolo estándar de comunicación *OpenFlow*. Siendo de las primeras interfaces de comunicación, definida entre el control y la capa de reenvío

(Forwarding) o de paquetes en los dispositivos tales como switches y routers, tanto físicos como a nivel virtual.

- **Controlador SDN (SDN Controller)**

  El controlador SDN [7] es una entidad lógica centralizada y responsable de un conjunto de tareas, incluyendo la extracción y el mantenimiento de una visión global de la topología de red y de su estado, así como de la creación de instancias lógicas de reenvío adecuadas para un determinado escenario.

  En la práctica, este controlador gestiona las conexiones a todos los elementos de la red, de manera que se encarga de instalar, modificar y eliminar las entradas de reenvío de las tablas de los switches conectados, mediante el uso de mensajes de control específicos del protocolo. Este es el elemento más crítico e importante de la arquitectura SDN. Sus tareas básicas incluyen el descubrimiento de los dispositivos, es decir:

  - Información de la topología de red,
  - distribución de la configuración,
  - Implementación de políticas definidas por el administrador de red y las aplicaciones disponibles.

Capa de infraestructura (Infrastructure Layer), está compuesta por elementos de red, que son gestionados por un sin número de protocolos de comunicación. Desde este punto el protocolo OpenFlow simplifica la configuración al administrador de red, siendo un estándar de código abierto que busca el desarrollo de las futuras redes programables.

El controlador SDN, el cual mantiene una vista global de la red toma todas las decisiones de procesamiento de los flujos de datos entre aplicaciones en la capa superior. Actualmente OpenFlow mantiene su visión como un protocolo estándar que permite la comunicación entre el plano de control y el plano de datos [7].

En este sentido la red se comporta como una arquitectura dinámica, manejable, económica y adaptable; ideal para las aplicaciones de naturaleza dinámica de hoy en día que usan gran ancho de banda con características destacables como:

● Directamente programable: el control de la red es directamente programable porque se desacopla de las funciones de reenvío.

● Ágil: la abstracción del control del reenvío permite a los administradores ajustar dinámicamente el flujo de la red ante las necesidades cambiantes.

● Configuración programable: SDN permite a los administradores de red configurar, administrar y optimizar los recursos de red rápidamente de manera dinámica con lineamientos de software y programas automatizados [50]

● Estándar abierto y neutral: en la figura [47], se muestra como SDN es un estándar abierto que simplifica el diseño y las operaciones de red, donde las instrucciones son proveídas por el controlador SDN en lugar de múltiples dispositivos y protocolos específicos.

2.1.3. Gestión de redes

La Gestión de red se define como el conjunto de actividades dedicadas al control y vigilancia de recursos de telecomunicación. Su principal objetivo es garantizar un nivel de servicio en los recursos gestionados con el mínimo costo de implementación [26]. Dentro de la gestión de redes se incluye el despliegue, integración, coordinación del hardware y software, así como los elementos humanos para probar, configurar, monitorizar, evaluar y controlar los recursos de dicha red; todo esto con el fin de cumplir con los requerimientos establecidos, el desempeño operacional y la excelente calidad de servicio [26].

Las principales funcionalidades de gestión de redes están relacionadas con las operaciones clásica, es decir, las funcionalidades de gestión fallas, de la configuración, responsabilidad, desempeño, y seguridad. Estas también son conocidas como FCAPS por sus correspondientes siglas en inglés [3].

Por lo tanto, funcionalmente, un sistema de gestión de red está compuesto por:

- Aplicación de monitorización: son aquellas funciones de la gestión y monitorización visibles por el usuario
- Función de gestión: es el módulo que recupera la información desde los elementos de red
- Función agente: Es el módulo que almacena información de uno o más elementos de red, y comunica la información al gestor
- Objetos gestionados: Se define como la información que representa los recursos y actividades de los elementos de red [26]

Así mismo dentro de la tarea de monitorización también se debe incluir un aspecto relevante y este es el parámetro de control cuya función específica dentro de la gestión de redes y monitoreo es de modificar parámetros e invocar acciones en los recursos gestionados. [4].
En lo que respecta a las SDN, dada la capacidad que se tiene de controlar la red de forma programática. Estas nos ofrecen nuevas alternativas para abordar las tareas de gestión, principalmente en lo que respecta al arranque y configuración

de la red; disponibilidad y elasticidad; programabilidad; rendimiento y escalabilidad, aislamiento y seguridad; flexibilidad y desacoplamiento, planeación de la red; y por ultimo monitorización y virtualización. [25]

2.1.4. Virtualización de Redes

El concepto de virtualización surge en el campo de la computación, y su principal función es separar tanto como sea posible los recursos computacionales de los recursos físicos que estos necesitan. [27] Los recursos físicos son abstraídos como un conjunto de recursos a partir de los cuales se crean entidades virtuales y asegurando su redundancia a través de sus copias.

En este sentido, la virtualización de red es un proceso desarrollado como una técnica de la comunicación que permite encapsular una unidad de proceso (programa, sistema operativo, red de datos) y representarlo en forma numérica digital y electrónica para su ejecución dentro de otro entorno en un equipo anfitrión que emula el entorno real transparentemente. [27 - 28] Lo conveniente es que la transparencia del sistema o red emulada sea casi exacta al sistema físico real.

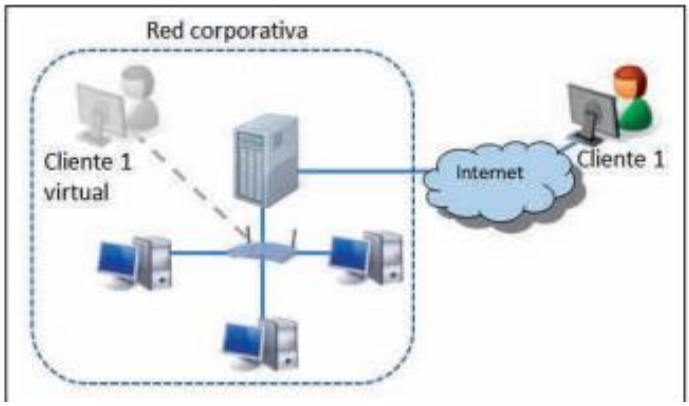


Figura 2. Virtualización de red. .[48]

Las principales ventajas de la virtualización de red son el ahorro de costos de infraestructura, la simplificación de la gestión (solo se debe gestionar un único punto (equipo anfitrión) en lugar de interactuar con un entorno de equipos interconectados), la configuración, la corrección de errores, actualizaciones y mejoras en las arquitecturas etc. [29]

La virtualización de funciones de red (NFV por sus siglas en inglés) es considerada como una de las tecnologías más importantes en la comunicación de la red. Se le denomina el pionero para lograr una utilización eficaz de los recursos con la disminución de los gastos operativos y los gastos de capital mediante la

disociación de las funcionalidades de redes de redes virtuales coexistentes Facilita el despliegue rápido de nuevos servicios y nuevas tecnologías. [30]

2.1.5. Monitorización de redes

La monitorización es la parte de la gestión de redes que se enfoca en la observación, análisis de estados, representación y mantenimiento de la información trasmitida en la red. Para ello hace uso de aplicaciones que se encargan de verificar la topología de la red, el estado de los dispositivos, rendimiento general de la red, realizar un análisis estadístico de tráfico y la detección de anomalías. [26][31 - 32 - 33]. Esto se hace con la necesidad de identificar los problemas en la red y poder darle una solución de forma automática o autónoma. Por ejemplo, para definir qué tipo de tráfico controlar, se puede mirar qué aplicación está consumiendo el ancho de banda Ethernet. Esto permite tener un control efectivo del tráfico de red para ser capaz de detectar posibles atascos en la red, y cerrar aplicaciones que consumen mucho ancho de banda cuando se necesitan los recursos para otras aplicaciones, así como limitar el ancho de banda disponible para cada equipo. [34].

La monitorización de la red está relacionada directamente con la observación y análisis del estado y comportamiento de los sistemas intermedios y finales, y de las subredes que constituyen la configuración a gestionar [26].

Generalmente puede dividirse en dos tipos; activo y pasivo, el ping y la ruta de seguimiento (traceroute) son un ejemplo básico del monitoreo activo, actualmente se usa el protocolo de red simple (SNMP) desarrollado principalmente para recolectar información estadística sobre los nodos involucrados en un punto central de la red. Por otro lado, el monitoreo pasivo tiene cuatro etapas que son un proceso cíclico donde se involucra la observación de paquetes, medición de flujo y exportación, recolección de datos y análisis de datos. [36]

La configuración para gestionar la monitorización en redes consiste en tres áreas de diseño importantes; acceso a la información monitorizada, que busca cómo definir la información de monitorización, la segunda es el diseño de mecanismos de monitorización, que trata el cómo obtener información de dichos recursos de la manera más óptima, y la tercera es la aplicación de la información monitorizada, que define como emplear la información monitorizada en las distintas áreas funcionales de gestión. [26]. Por otro lado, la monitorización del tráfico de red consiste en realizar el control de envió y reenvió de paquetes de datos, mediante dispositivos físicos que se encargan de copiar los paquetes que fluye a través de ellos para ser analizados sin la necesidad de interferir con operaciones de reenvió. De esta manera, los paquetes de datos que se transmiten pueden ser entregados

a su destino final [37]. Cuando se monitorea una red la seguridad permite la confidencialidad, integridad, disponibilidad de la información, así como también desplegar nuevas funcionalidades y manipular el comportamiento de la red. Para esto se hace necesario que los administradores de red hagan la configuración de cada uno de los dispositivos físicos involucrados utilizando comandos de bajo nivel especificados por parte del proveedor [38].

2.2. Trabajos relacionados

La mayoría de los trabajos relacionados citados a continuación hacen énfasis en el monitoreo de la tecnología SDN, sus características desventajas y aprovechamiento de recursos que brinda la misma, por ende, es importante dar a conocer los diferentes avances que han aportado dichas investigaciones. El monitoreo de estas redes ha generado gran interés en la industria porque de ello depende la funcionalidad, flexibilidad, y seguridad de la información en una red.

2.2.1. A Network Monitor and Controller using Only OpenFlow

En el artículo [39] proponen como hipótesis de investigación que OpenFlow (OF) proporciona la suficiente información para tener calidad de servicio (QoS) básica sin la necesidad de software adicional en los switches, utilizando una API de la southbound. Sin embargo, El problema nace en el estado de la red, donde su comportamiento presenta desafíos debido a su cambio constante, es aquí donde la monitorización es necesaria para mantener actualizado el estado de los recursos ya que esto permite dar el primer paso para controlar la red.

Entonces las principales soluciones SDN, agregan un agente de software dentro del switch, sin importar si estas soluciones sean propietarias o no, lo que hacen al agregar un agente de software es contradecir la idea de que las SDN favorecen el hardware básico, por lo cual se hacen las siguientes preguntas ¿qué se puede hacer con respecto a la monitorización? ¿Sin agregar más software? [39].

Para dar solución a estas preguntas, los aportes más interesantes empiezan por hacer uso de OF minimizando el hecho de agregar software al switch, para crear el OFFMonitor, y también el uso de una aplicación de ingeniería de tráfico en un router QoS. Para esto se monitorizando el tráfico de la red a través del router, con el fin de mejorar el enrutamiento a partir de los parámetros establecidos (mejorar QoS) [39] Donde se virtualiza la red utilizando la herramienta mininet, en el cual primero hacen las mediciones con OOFMonitor en diferentes situaciones de topología, y luego muestran los resultados utilizando el router QoS, para ambos casos evalúan el retardo de los datos, el Jitter (variación en el tiempo en la llegada de los paquetes) y el porcentaje de perdida de paquetes.

Debido a esto la desventaja más notable en este trabajo, es la limitación al uso de un solo protocolo en este caso OF, por lo cual la monitorización solo se está haciendo en el plano de control, dejando a la deriva los datos que fluyen a través del plano de gestión.

Análisis:

Con el enfoque de monitoreo SDN, la principal contribución que se hace es con un monitor de red Offmonitor que proporciona cuatro parámetros básicos de calidad de servicio QoS. Este monitor basado en OpenFlow, enfatiza el potencial del estándar proporcionando porcentaje de utilización, Retraso, Jitter y porcentaje de pérdida de paquetes, lo que lo asemeja en cierto modo a la propuesta de nuestra investigación solo que para nuestro caso implementamos OpenNetmon y Wireshark como herramienta de monitoreo y no Offmonitor. Otro aspecto importante que nos brinda es la forma en como el monitoreo puede integrarse con el control.

2.2.2. Decentralized monitoring for large-scale Software-Defined Networks

En el artículo [20] contrastan los beneficios del sistema de monitorización descentralizado basado en un caso de uso real de una red a gran escala, comparando el rendimiento en términos de monitorización de latencia, sobrecarga de tráfico, con una solución centralizada. Mostrando el impacto de la monitorización distribuida en el rendimiento de la aplicación. Así como también los "tradeoffs" entre aplicaciones de reactividad/precisión y monitorización de la escalabilidad/sobrecarga.

El problema se encuentra cuando las investigaciones sobre la implementación para la monitorización de mediciones de tareas específicas han sido soluciones basadas en la suposición de una red que está gestionada centralizadamente, lo cual es una limitación importante para el caso de las topologías de red. Ya que los enfoques centralizados tienen limitaciones de escalabilidad, debido a la gran cantidad de tráfico y carga de procesamiento que convergen en un solo controlador/administrador [20]

En este trabajo, llama la atención es la arquitectura implementada para la realización de una monitorización descentralizada, donde se enfocan en monitorizar el rendimiento en cuanto a términos de latencia y sobrecarga de tráfico, para la prevención de cuellos de botella. Usando la herramienta mininet, para emular la topología de red, incluyendo hosts, clientes y servidores de contenido, así como switches openflow. Además, usan gestores locales (LM – Local Managers), al igual que un módulo de monitorización (MM – Monitoring

Module) y la aplicación lógica de balanceo de carga, implementando también un conjunto de módulos de Python, y reúsan un pequeño conjunto de APIs del controlador SDN POX, todo esto con el fin de evaluar el retardo de la información [20]

Una desventaja que se puede apreciar es la necesidad de agregar a cada dispositivo un gestor local (LM), lo cual requiere que los datos recopilados sean sincronizados, lo cual deja la pregunta ¿qué pasaría si uno de estos LM llegara a fallar? Además de que esta implementada en una red a gran escala y que los resultados son favorables, no hay evidencia de que pueda aplicarse a un tipo de red más pequeña.

Análisis:

El principal aporte que este trabajo le da a nuestra investigación es la arquitectura implementada para la realización de una monitorización descentralizada, se asemeja en la forma en cómo se implementa en entorno virtual con herramientas como mininet y OpenFlow pero se diferencia porque se enfocan en monitorizar el rendimiento en cuanto a términos de latencia y sobrecarga de tráfico, para la prevención de cuellos de botella.

2.2.3. Hardware Modules for Packet Interarrival Time Monitoring for Software Defined Measurements

En el artículo [40] Se hace la implementación de una arquitectura de medición para redes SDN, ampliando el switch openFlow a la plataforma NetFPGA 1G para la elaboración de mediciones precisas y en tiempo real de los flujos TCP, lo que permite mediciones definidas por software (SDM) debido a sus ventajas para recopilar estadísticas sobre la marcha. Así como también se habla de mediciones simples y escalables sin depender de los hosts finales, facilitando la realización de un seguimiento del rendimiento en la granularidad de flujos.

El esfuerzo que se ha hecho para el desarrollo de aplicaciones de control SDN, ha dejado de un lado los sistemas de medición y seguimiento de tráfico con las SDN, sin quitarles importancia, pero se sabe que permiten la implementación de técnicas de ingeniería de tráfico. Por lo cual el problema que se pretende abarcar es el desarrollo de estas técnicas para las redes sabiendo que son un desafío debido a la complejidad y las restricciones en tiempo real de estas aplicaciones [40]

En este caso hacen uso de dos hosts A y B, dos switches con tráfico real para agregar retrasos realistas entre los paquetes enviados por A, y se agrega la plataforma NetFPGA entre un switch y el host B. Usaron intervalos de tiempo aleatorios, generando paquetes con la librería Libnet del lenguaje C para tener un mayor control del número de paquetes que llegaron. Se realizó con el fin de evaluar el retardo entre el host A y el host B [40]

Análisis:
El aporte de este trabajo es el desarrollo de la aplicación user-space que se encarga de leer los tiempos de registro de llegada de paquetes registrados en la SRAM para envió de paquetes TCP. Con el interés específico de monitorizar el tráfico de red en general para la medición del flujo en un solo punto ya que el controlador puede recopilar estadísticas de flujo sobre la marcha, se diferencia a nuestra investigación por la pretensión de desarrollo de un aplicativo de monitoreo de retardo a diferencia que nuestra propuesta que es hacer seguimiento con herramientas ya desarrolladas.

2.2.4. An efficient flow monitoring algorithm using a flexible match structure

En el presente artículo [41] Se hace el diseño de un algoritmo para el monitoreo de flujo eficiente mediante la explotación de tres mecanismos de sondeo, poll-single, poll-some y poll-all, enfocando principalmente el algoritmo a la estructura poll-some para identificar todos los flujos no cubiertos en un switch, y comparan algoritmos creados anteriormente.
Se han desarrollado algoritmos para recopilar información de estadísticas y el mecanismo pull-some aún no ha sido adoptado por ningún algoritmo de monitoreo de flujo existente. Por lo cual han desarrollado un algoritmo Heurístico llamado Critical Column First (CCF) para resolver el problema mínimo de estructura de coincidencia. El cual se aplica para monitorizar el flujo, reduciendo el costo de comunicaciones entre switches. Para esto crearon redes simuladas con 50 switches, construidas por el generador Erdos-Renyi, el cual se encarga de generar aleatoriamente las redes.

Análisis:
El principal aporte que hace a nuestro proyecto es el desarrollo un algoritmo Heurístico llamado Critical Column First (CCF) para recopilar información y estadísticas sobre el flujo eficiente mediante la explotación de tres mecanismos de sondeo, poll-single, poll-some y poll-all, su diferencia notable es el enfoque a monitorear porque las pretensiones de nuestro trabajo es realizar seguimientos a los parámetros del Jitter mediante una función computacional.

2.2.5. A monitoring framework for 5G service deployments

El tema central en este trabajo [42] es discutir las principales características de monitoreo de red, así como los principales requisitos impuestos a los sistemas de monitoreo de recursos con un enfoque SDN para apoyar futuros despliegues en servicios 5G. Además, se expone una arquitectura genérica de adquisición de datos para monitorear tanto física como virtualmente un servicio presentando la implementación de un marco de monitoreo mediante virtualización que sigue los principios de diseño ya establecidos.

En este documento, se identifican primero que todo los desafíos de escalabilidad de monitoreo impuestos por la visión de comunicación móvil 5G cuyas funcionalidades requieren un sistema de monitorización robusto y flexible que lo ayude en su eficiente desempeño y tal vez a otros sistemas del mismo tipo.

Por otro lado, Las características que manejan actualmente los sistemas de monitoreo de servicios no son las más adecuadas puesto que el monitoreo del modelo estructural de las tecnologías 5G requiere el aprovisionamiento de una serie de componentes y requerimientos y las redes inalámbricas actuales no están optimizadas para dar soporte a este modelo de despliegue, ni en términos de vida útil ni en costo-eficiencia. [43]

Conjugado a esto se da a conocer la implementación preliminar del sistema o marco de monitoreo holístico (Integración total) desarrollado en el marco del proyecto CHARISMA cuyo propósito es proporcionar una vista de extremo a extremo de los recursos para una plena optimización en la gestión de estos, así mismo contribuir con la mejora en las políticas de seguridad y aportar a los sistemas de toma de decisiones, brindar garantías tanto de SLA y gestión de fallas.

Análisis:

Como aporte en este trabajo, se propone una arquitectura genérica que brinda una solución de monitoreo integral a la red, esta solución propuesta ha sido diseñada para realizar adquisición de datos de monitoreo en tiempo real y recopilar información de eventos (métricas, kpi, alertas, retardos) que se producen en el entorno físico y virtual tanto de los recursos como la infraestructura, se diferencia a nuestra investigación en la implementación de su arquitectura a gran escala que pretende medir múltiples componentes de servicios 5G.

2.2.6. Interactive monitoring, visualization, and configuration of OpenFlow-based SDN

En el artículo [44] se plantea como tema central el sistema de administración y gestión de red dedicado a SDN, planteamiento dado a partir del análisis del tráfico de control en SDN cuyo objetivo es comprender mejor el efecto de la comunicación entre el controlador y los dispositivos, enfocándose en el impacto general en términos del consumo de recursos y el rendimiento de la red, donde la mayor prioridad es encontrar un enfoque interactivo para la gestión y administración de la red mediante el monitoreo, visualización, y configuración que incluye al administrador.

Las dificultades surgen a raíz de que las actividades de gestión para SDN, como el monitoreo, la visualización y la configuración pueden ser considerablemente diferentes de las redes tradicionales; incluso en algunas SDN no se enfatiza el tema de la gestión estructurada. Por otro lado, Un controlador normal de SDN, puede ser personalizado por los administradores de red según sus necesidades, dichas personalizaciones pueden tener un impacto en el consumo de recursos y el rendimiento del reenvío de tráfico por esta razón se hace necesario un enfoque a la gestión directamente en SDN que incluya al administrador como componente principal.

Por esta razón, En este artículo como solución al problema, se pretende Integrar las tres principales actividades de administración de red (configurar, reconfigurar y tomar decisiones) que permiten al administrador comprender y controlar la red, mediante el desarrollo y la implementación de un prototipo de gestión usando el controlador de Floodlight y un escenario de red emulado sobre Mininet que usa el protocolo OpenFlow. Este prototipo es capaz de realizar monitoreo con intervalos de sondeo configurables específicamente centrado en métricas relacionadas con el uso de recursos y controlar la carga del canal, asimismo presentar estadísticas agregadas en forma interactiva y las visualizaciones que enfatizan estas métricas.

Del mismo modo, al interactuar con visualizaciones, el administrador puede ajustar la red, mejorar la configuración del controlador y, por lo tanto, reducir el consumo de recursos y el uso del canal de control, de manera que los resultados obtenidos con la implementación del prototipo de monitoreo muestran que el enfoque puede ayudar al administrador a comprender mejor el impacto de la configuración de los parámetros relacionados con SDN en el rendimiento general de la red.

Análisis:

El aporte que esta investigación hace a nuestra propuesta es el desarrollo de un sistema de administración y gestión de red dedicado a SDN, mediante el monitoreo, visualización y configuración de OpenFlow, planteamiento dado a partir del análisis del tráfico de control en SDN cuyo objetivo es comprender mejor el efecto de la comunicación entre el controlador y los dispositivos, difiere en nuestra proyecto por la generalidad en que se miden los componentes y la no aplicabilidad de funciones para seguimiento de Jitter.

2.2.7. Adaptive flow monitoring in SDN architecture

Este trabajo [45] está basado en un procedimiento de seguimiento y monitorización de un método adaptativo del flujo en SDN descrito mediante Openflow. Argumento propuesto debido a que, en algunos casos, se requieren actividades de monitoreo para los flujos específicos que mejoran el diseño de la arquitectura de monitoreo y sirven para detectar los eventos especiales, además de que ayudan a gestionar las SDN de manera más precisa y eficaz.

El problema principal radica en que el flujo de las redes no es adaptativo y por esta razón se presenta una enorme carga que ralentiza los procesos de los dispositivos en las redes SDN. Además, el estándar actual de la arquitectura SDN obliga al switch a permanecer siempre activo desde el momento en que recibe la solicitud de supervisión de eventos del controlador SDN hasta que se captura el evento objetivo en el flujo correspondiente.

Dicho en otras palabras, el controlador SDN da las órdenes al switch SDN para monitorear los flujos específicos, y este sigue monitoreándolos hasta detectar los eventos especificados. Pero según el análisis esto no es lo más adecuado porque este mecanismo o evento especial podría alterar la filosofía fundamental de SDN, que recomienda que el switch SDN se concentre en el reenvío de datos, lo que da como resultado la degradación del rendimiento de toda la red SDN.

Como solución a dicho inconveniente se propone en este articulo la implementación de un nuevo método de monitoreo para flujo adaptativo en SDN que busca simplificar y optimizar la carga del switch SDN haciendo que el controlador SDN sondee dicho switch de forma adaptativa, en lugar de hacer que este espere los eventos todo el tiempo

Por consiguiente, este método puede contribuir a la reducción de costos y gestión eficiente de recursos para el servicio de red de proveedores, mejorando el manejo y optimización de la carga en los switches del sistema de red SDN.

A continuación, se evidencian las ventajas que propone el método de monitoreo de flujo adaptativo:

- La implementación del switch SDN puede ser simplificada
- El rendimiento del swich SDN se puede mejorar.
- El costo de construir la red SDN puede disminuir debido al reducido tamaño de la memoria, los ciclos de la CPU
- El período de construcción de la red SDN puede ser reducido

Análisis:

El método propuesto en este trabajo puede contribuir a la reducción de costos y gestión eficiente de recursos para el servicio de red de proveedores, mejorando el manejo y optimización de la carga en los switches del sistema de red SDN, se diferencia de nuestra propuesta en que se hará el seguimiento a un flujo específico y no continuo y para medir eventos como Delay y Jiter se necesita tener monitoreado continuamente la red.

2.2.8. Outsourcing the Routing Control Logic: Better Internet Routing Based on SDN Principles

En este trabajo [48], se describe como eliminar los interdominios, logrando un modelo de información descentralizado que interactúa por medio de protocolos de comunicación. Se plantea una lógica centralizada por medio de múltiples sistemas autónomos determinando el recorrido por una plataforma de control basado en SDN, principalmente por su eficiencia en decisiones de tráfico, detectando conflictos con políticas de seguridad y localización de fallas, permitiendo mayor control.

En la actualidad vemos como las redes de telecomunicaciones cada vez se tornan menos centralizadas y más abiertas antes diferentes cambios estructurales. El eliminar totalmente los interdominios basados en un modelo centralizado, donde los múltiples sistemas autónomos (AS), interactúan por el protocolo mediante el cual se intercambia información de encaminamiento o ruteo entre sistemas autónomos (BGP). Aunque el protocolo planteado aparenta ser muy prometedor, éste afronta varias problemáticas como lo son: completa distribución, capacidad de políticas de ejecución, dimensión, seguridad y complejidad [10].

Análisis:

El principal aporte que brinda a nuestro proyecto es el desarrollo de un modelo de información descentralizado que interactúa por medio de protocolos de comunicación como OpenFlow, plantean una lógica por medio de múltiples sistemas autónomos y su eficiencia se basa en decisiones de tráfico, detectando

conflictos con políticas de seguridad y localización de fallas, permitiendo mayor control.

2.2.9. PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks

En el trabajo [51], el tema principal es la gestión y administración que requieres los recursos de red porque esta necesitan estadísticas precisas y oportunas sobre dichos recursos en diferentes niveles de agregación. Sin embargo, la sobrecarga de la red para la recopilación de estadísticas debe ser mínima.

Las estadísticas precisas y oportunas son esenciales para muchas tareas de gestión de red, como el equilibrio de carga, la ingeniería de tráfico, la aplicación del Acuerdo de nivel de servicio (SLA), la contabilidad y la detección de intrusiones.

Con un enfoque hacia el desarrollo de aplicaciones de gestión de red, se propone PayLess, como un marco de monitoreo de bajo costo para redes SDN, este modelo proporciona una vista abstracta de la red y una manera uniforme de solicitar estadísticas sobre los recursos. En segundo lugar, PayLess se desarrolla como una colección de componentes conectables, la interacción entre estos componentes se abstrae mediante interfaces bien definidas.

Análisis:

Como aporte, principalmente Payless genera un campo de estudio que contribuye al desarrollo de la propuesta de monitoreo sigue el lineamiento de algunas herramientas de monitoreo inspiradas en OpenFlow y métodos de recolección de datos adaptativos de velocidad variable utilizados en redes de sensores se evalúan y comparan el rendimiento de la aplicación de monitoreo mediante la emulación de red con mininet, otra semejanza es el estudio del problema de compensación de recursos-precisión en el monitoreo de la red, para eso proponen un algoritmo de programación de recopilación de estadísticas de adaptación de frecuencia variable, similar al que se pretende implementar en nuestro estudio para medir eventos como el Jitter.

CAPÍTULO 3. ARQUITECTURA DE UNA RED SDN

En la actualidad el servicio de internet y la tecnología han crecido exorbitantemente con diversidad de complejidad, diseño, gestión y operatividad; a causa de esto, se presenta el surgimiento de nuevos niveles de habilidad de manejo de dichas tecnologías, en los últimos años, las redes definidas por software (SDN), han establecido un antes y después en cuanto a trasmisión de información se refiere, consiguiendo aplicabilidad por contar con buena facilidad de gestión y un flexible desarrollo de nuevas redes, las SDN presentan un nuevo diseño de arquitectura y tecnología comparados a la redes tradicionales, tanto así que es considerada como una de las principales propuestas para la viabilidad de Internet en el futuro [50]

El pasar del tiempo ha dado su lugar a la virtualización de redes, impulsada principalmente por NFV (*Networking Founfation Virtualitation*), que con un modelo que permite escalabilidad, portabilidad, economía, agilidad y adaptación de estándares de negocio orientados a la nube, se convierte en una solución prometedora que marcara los pasos de los próximos años.[52]. Para que las SDN se desempeñen no necesariamente requieren dicho modelo que lo soporte, pero adopta tal avance que permite nuevos caminos y estrategias para los futuros desarrollos.

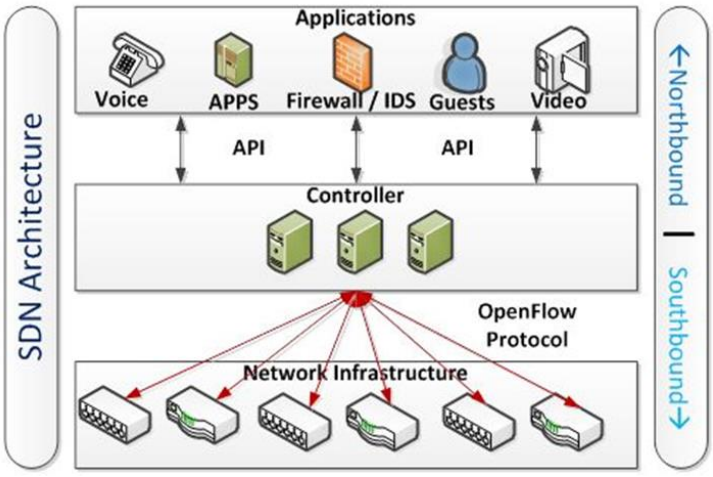


Figura 3. Arquitectura SDN (SDN and Openflow for beginners)

La arquitectura SDN, especificada por la ONF (Open Networking Fundations) que se ilustra en la *Figura 3,* muestra a un alto nivel, los puntos de referencia, capas e interfaces dirigidas al controlador, pero para que este modelo de arquitectura fuera eficiente fue necesario crear y estandarizar una interfaz de comunicaciones entre el control y el reenvío de datos[53], Para ello se creó el protocolo OpenFlow que permite el acceso directo a la gestión de datos de reenvío en dispositivos de red como switches y routers, tanto físicos como virtuales y de un modo abierto, ningún otro protocolo estándar tiene la funcionalidad y finalidad de OpenFlow que transfiere el control de los dispositivos de red a la lógica del software de control.[53]

La llegada de la virtualización flexibilizo la definición y el uso de los recursos, permitiendo definir y modificar en tiempo real, a nivel de software, una infraestructura completa basada en perfiles de aplicaciones y necesidades de rendimiento.[50]

En el desarrollo de este capítulo se pretende exponer criterios y tecnologías usadas para un entorno virtual SDN, además las diferentes versiones de software que han venido adelantando diferentes actores comprometidos con la siguiente generación de redes de datos.

3.1. Emuladores y Simuladores de Red SDN

La implementación de soluciones como SDN a través de una arquitectura de red flexible y un estándar abierto, permite evidenciar la agilidad y programabilidad de las tareas, al tiempo que reduce los costos de operación, mantiene el desarrollo de servicios, y permite a los administradores de red integrar con mayor rapidez las nuevas tecnologías, así mismo se ha propuesto a OpenFlow, como el protocolo estándar que permita conducir el tráfico organizado y los flujos de datos, y cómo estos flujos pueden controlarse según las necesidades específicas [54].

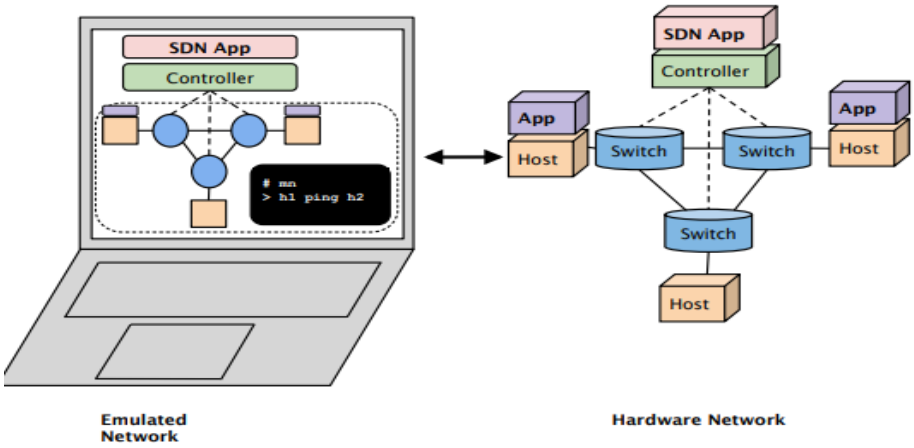


Figura 4. Topología red virtual vs Topología de física 55]

Por conveniencia, en una red SDN, debido a que la operación y la inteligencia están totalmente centralizadas en un controlador, la exactitud y la eficiencia de las funciones implementadas por el controlador deben estar completamente probadas antes de su uso en una red física., por consiguiente es factible para corregir y evaluar el rendimiento de todo el entorno de red y sus protocolos. [56]

Tomando en cuenta que las herramientas de simulación permiten este tipo de nuevas soluciones (SDN, Openflow), el camino de la virtualización es adecuado para planificar, probar y solucionar problemas de una red entera a través de la plataforma escalable de cualquier programador, sin la necesidad de interactuar directamente con el hardware de red. *Vea Figura (004).* [53] del mismo modo, el concepto de emular está ligado a ejecutar sin modificación alguna un código o procedimiento de forma interactiva en un hardware virtual dentro de un ordenador normal, con la ventaja que se obtiene comodidad, seguridad de procesos y realismo a bajo costo. [55]

Actualmente existen varios simuladores de entornos de red, que permitan emular topologías SDN para realizar pruebas y evaluar tanto protocolos como aplicaciones. MiniNet, es uno de los más conocidos por ser un emulador que crea una red de hosts, switches, controladores y enlaces virtuales. Los hosts corren un software de red Linux estandarizado, los switches soportan OpenFlow con gran flexibilidad para la configuración de ruteo y SDN, dicho entorno permite investigar, desarrollar, aprender, armar prototipos, testing, debugging, y otras tareas con el beneficio de tener una red experimental funcionando dentro de cualquier ordenador.[53]

Entre otros entornos virtuales conocidos se encuentra el simulador GNS3, que es un simulador de red gráfico que permite virtualizar, probar y solucionar problemas de una red a través de la plataforma escalable, lo que diferencia a GNS3 de los demás simuladores, es la habilidad para emular ruteo y conmutación como también incorporar verdaderas máquinas virtuales y conectarlas entre sí a través de un sistema de túneles lógicos. [53]

Así mismo sobresale, Estinet [57] que utiliza un enfoque único para probar las funciones y actuaciones de los controladores OpenFlow, este combina las ventajas de ambos enfoques de simulación y emulación para conseguir verdaderos resultados de precisión a precisión se refiere. En entornos de red programables.

De este modo, conociendo de primer plano las ventajas que cada entrono de simulación presenta y tomando en cuenta los parámetros establecidos para el desarrollo de este proyecto se admiten los siguietes simuladores.

3.1.1. MiniNet

Es una plataforma de emulación, que crea y simula entornos de red totalmente escalables con dimensiones según la configuración deseada, utiliza el kernel de Linux y otros recursos para emular elementos de la SDN como el controlador, los switches OpenFlow y los hosts. [56] dicha plataforma permite crear, interactuar, personalizar y compartir de forma rápida un prototipo de red definido mediante software al mismo tiempo que proporciona un camino fácilmente adaptable para la migración a hardware. [58]

Ésta herramienta que simula comportamientos de red [53] por su adopción del protocolo OpenFlow, permite ejecutar una colección de dispositivos de conmutadores, controlados y gestionados por un administrador de tipo Controlador.[55] En resumen, los dispositivos virtuales de MiniNet, conmutadores, enlaces y los controladores se crean utilizando software en lugar de hardware, en su mayor pate el comportamiento es similar a los elementos de hardware, por ser una plataforma de tipo virtual es muy eficiente al momento de ejecutar funciones de red tan solo con líneas de comandos. [59], como se muestra en la siguiente ilustración. Figura (005)

Figura 5. Comando para habilitar mininet

Muy ajustado a la tecnología de redes definidas por software SDN, mininet utiliza la virtualización ligera para hacer que un solo sistema parezca una red completa. Los programas que se ejecutan pueden enviar paquetes a través de lo que parece ser una interfaz de Ethernet real, con una velocidad de enlace y parámetros de retardo. [59], del mismo modo los hosts creados corren un software libre de red Linux y los switches soportan OpenFlow con gran flexibilidad para la configuración de ruteo y SDN. [53].

Atributos de Mininet

Los principales atributos y características en los que se basa MiniNet son los siguientes:

- Flexibilidad: Soporta diferentes tecnologías y funcionalidades de las nuevas tecnologías de red basadas en software.
- Desplegable: La implantación correcta de un prototipo SDN no debe exigir cambios en el código.
- Interactivo: La gestión y la operación de la red debe realizarse en tiempo real, como si no se tratase de una red virtualizada.
- Escalable y Realista: el comportamiento de la red debe coincidir con la real, así mismo soporta la creación de diferentes topologías totalmente personalizables.
- Compartible: Compartición de los diferentes prototipos para poder realizar pruebas de todos ellos en diferentes experimentos.

Es importante aclarar también que así como mininet posee bastantes ventajas también precisa limitaciones como la pérdida de fidelidad en su rendimiento especialmente ante grandes cargas debido al multiplexado en el tiempo de los recursos de la CPU planificado por defecto en Linux. [58], del mismo modo hace uso del kernel de Linux para todos los host virtuales, lo que significa que no se puede comunicar ni ejecutar software que depende de BSD, Windows, u otros núcleos del sistema operativo. (Aunque una opción es conectar máquinas virtuales a MiniNet.).[59]

Instalación de Mininet

La herramienta Mininet puede ser instalada de distintas maneras. Una de las más, recomendada para empezar a interactuar con la plataforma, consiste simplemente en descargar una imagen de una máquina virtual que puede ser ejecutada posteriormente en un software como VirtualBox [60]

La segunda forma, es en una instalación nativa en un sistema operativo de tipo Linux puede ser Ubuntu recomendablemente o Fedora. Para esto basta sólo con clonar el código fuente, abrir el directorio y ejecutar una instrucción específica para la instalación. Esta instrucción permite seleccionar, entre una serie de opciones, los elementos específicos a instalar. Por ejemplo, puede instalarse o no Wireshark, junto con los otros elementos del emulador. Las instrucciones para la instalación de Mininet por cualquiera de los dos métodos mencionados están disponibles en el siguiente capítulo. [56]

Topologías de Mininet

Mininet cuenta con un sinnúmero de configuraciones de red con las cuales Puede soportar diferentes tipos de topologías [61]. Las topologías disponibles son: minimal, single, linear, tree y personalizadas. En cualquiera de éstas existe un controlador, varían en el número de hosts, switches y los enlaces entre éstos, por ejemplo Minimal, está compuesta por dos hosts conectados a un switch; single también es una topología con un único switch pero la cantidad de host puede indicarse por parámetro. [56]

Una vez instalado el emulador, sólo con escribir la instrucción **$ sudo mn** en el Shell se ejecuta Mininet *Figura 005* y se crea la topología denominada mininal, el entorno cambiara y es donde se sabe que ya se está dentro del emulador.

La instrucción **sudo mn** puede venir sola o acompañada de diferentes parámetros, entre estos algunos de ayuda, para limpiar y reiniciar el emulador y otros para ejecutar topologías de otros tipos. [56]

Un ejemplo claro lo vemos al ejecutar el siguiente comando *Figura 6*, al hacerlo se genera un solo Switch con 3 hosts conectados

```
$ sudo mn --arp --topo simple, 3 --mac --switch ovsk --
controller remote
```

Figura 6. Comando para crear una topología de red

En la instrucción anterior, hay algunas palabras clave importantes a las que vale la pena prestarle atención: [61]

- - **mac** : brinda el ajuste automático de direcciones MAC

- - **arp** : rellena las entradas ARP estáticas de cada host
- - **switch** : ovsk se refiere al modo kernel OVS
- - **controller** : Es el controlador remoto puede tomar la dirección IP y el número de puerto como opciones

  El estilo de la topología se vería de la esta forma como lo muestra la siguiente ilustración *Figura 7*

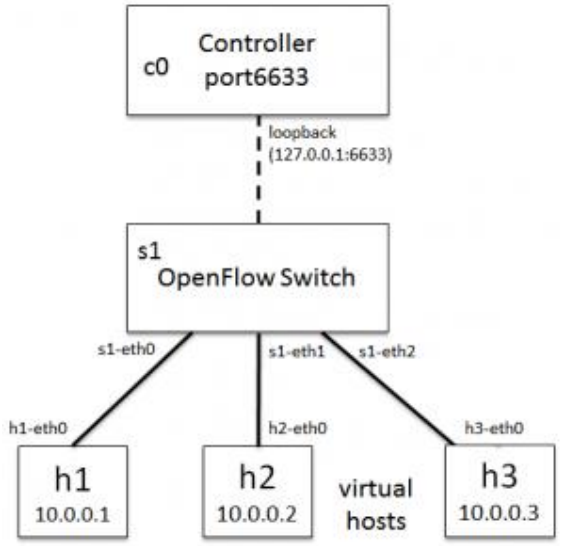


Figura 7. Arquitectura de topología básica en mininet

3.1.2. Xming

Xming es el servidor del sistema de ventanas X (X System Windows) líder de Microsoft, cuenta con todas las funciones, es gratuito, delgado, rápido, fácil de instalar y porque es Windows nativo independiente, fácilmente portátil no necesita una instalación específica de la máquina ni acceso al registro de Windows. [62]

El Sistema de Ventanas X es un sistema gestor de ventanas, común en sistemas operativos del tipo UNIX, que fue desarrollado para dotar de una interfaz gráfica a los sistemas Unix. Este protocolo permite la interacción gráfica en red entre un usuario y una o más computadoras haciendo transparente la red para éste. Generalmente se refiere a la versión 11 de este protocolo, X11, el que está en uso actualmente. X es el encargado de mostrar la información gráfica de forma totalmente independiente del sistema operativo.[64]

El servidor X Xming está basado en el servidor X.Org, cruzado en Linux con el compilador MinGW y Pthreads-Win32, se usa comúnmente en implementaciones de Secure Shell (SSH) para asegurar sesiones X11 en otras computadoras. Soporta PuTTY y ssh.exe, y tiene una versión del plink.exe de PuTTY.

Algunas de sus principales funciones son: [62]
Permitir que las máquinas Windows y Linux / Unix se integren como una nube privada / personal

- Permitir usar su computadora Microsoft como terminal de cliente ligero para máquinas Linux / Unix
- Poder visualizar clientes X remotos directamente en el escritorio de Windows sin ejecutar un Administrador de visualización remoto y así liberar los recursos utilizados por ese DM.
- Utilizar SSH y X-Forwarding en el modo *-multiwindow* para recuperar recursos valiosos y facilita las funciones de cortar y pegar entre ventanas.
- Permite acceder remotamente al servidor Linux y ejecutar algunas aplicaciones gráficas desde un equipo con Windows.[63]

3.1.3. PuTTY

PuTTY es una implementación gratuita o cliente de acceso remoto de SSH y telnet, a sistemas informáticos o plataformas Windows y unix. Es un cliente de poco tamaño y muy liviano que nos da acceso a una consola dentro del servidor para ejecutar en cualquier computadora con el sistema operativo Windows. Combinándolo con aplicaciones como Xming podremos iniciar aplicaciones gráficas de Linux en Windows XP de una manera bien sencilla. [65]

PuTTY utiliza el protocolo SSH para la comunicación segura de datos entre servidor y un cliente, además que se utiliza para generar un inicio de sesión de forma segura también ayuda a mantener la seguridad en la transferencia de archivos. [66]

SSH (Secure SHell): es un protocolo de red que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y 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 información y contraseñas no encriptadas. [58]

3.2. OpenFlow

Es claro que SDN cuenta con innumerables atributos que lo convierten en una de las tecnologías para revolucionar las redes de datos, entre muchos de sus características se encuentra la capacidad de controlar el nivel de red de reenvío de paquetes, sin embargo, para que suceda, necesita una interfaz que sea simple de administrar al momento de la comunicación entre los elementos de conmutación de la red y el controlador de red, de hecho, la interfaz que se ha asociado desde el principio al paradigma SDN fue OpenFlow, siendo incluso uno de los factores motivantes para su creación.[112]

En la arquitectura SDN descrita anteriormente, se evidencia una clara comunicación entre los controladores y los elementos de red, esta conexión se basa normalmente en el protocolo OpenFlow, siendo el protocolo de comunicación más avanzado entre un plano de control centralizado (controlador) y el plano de datos (conmutadores, enrutadores). [68]

OpenFlow[54], fue propuesto por la Universidad de Stanford, su desarrollo comenzó en el año 2007 como parte de la colaboración entre el mundo académico y el mundo industrial, está estandarizado por Open Networking Foundation (ONF) e implementado por muchos empresas fabricantes [68]. Su objetivo inicial es cumplir con la validación de demanda de la nueva arquitectura propuesta y protocolos de red en equipos comerciales. Por lo tanto, es posible implementar una tecnología capaz de promover la innovación en el núcleo de la red, mediante la ejecución de redes de prueba en paralelo con las redes de producción. La propuesta de OpenFlow promueve la creación de SDN, utilizando elementos comunes de la red como conmutadores, enrutadores, puntos de acceso o incluso computadoras personales. [112]

OpenFlow permite implementar fácilmente protocolos de enrutamiento y un cambio en la red sin la necesidad de cambiarla físicamente. Su uso está basado en aplicaciones como movilidad de máquinas virtuales, redes seguras y en la próxima generación de redes móviles IP [30].

Algunas de sus características principales son [68].

- Abierto: el hecho de utilizar un conjunto de instrucciones estándar en el proceso de comunicación entre el controlador y el conmutador proporciona una red abierta independiente de proveedores y fabricantes de dispositivos de red.

- Programable: OpenFlow se usa para crear conjuntos de reglas que funcionan en combinación con las opciones de configuración para un

proveedor de dispositivos, o independientemente de ellos. Además, es posible programar una red para los requisitos específicos de una aplicación determinada.

- Virtualizado: OpenFlow puede especificar diferentes reglas de reenvío para varios tipos de datos y permite crear varias rutas de comunicación lógica en la misma red física. En otros términos se refiere a la virtualización de la red

3.2.1. Arquitectura OpenFlow

Haciendo uso de OpenFlow, el controlador puede dictar normas específicas a los Switches habilitados para SDN, estas reglas definen si los flujos que se ajustan a las características específicas deberán ser remitidos, desviados, modificados o caídos según las políticas de servicio en forma QoS [54].Su funcionamiento esta dado a partir de una arquitectura muy versátil que conectando dinámicamente los elementos de la red SDN. [68], como se muestra en la *figura 009* a continuación se describen los elementos:

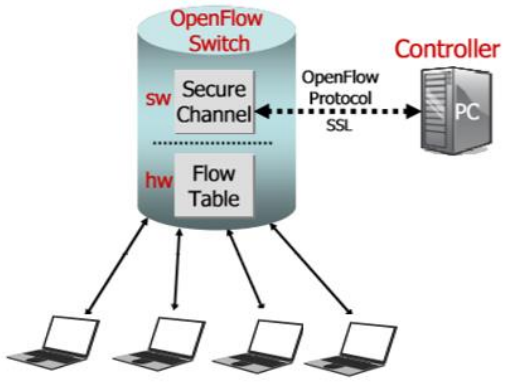


Figura 8. Arquitectura de red Openflow

- **Controlador OpenFlow:** es quién gestiona los switches, los puertos de cada switch, la calidad del servicio (QoS) y las colas asignadas al host físico, además procesa los paquetes informados en el switch OpenFlow.

- **Protocolo OpenFlow:** proporciona una interfaz de comunicación abierta y estandarizada entre el controlador OpenFlow y los switches OpenFlow. Por lo tanto, el buen funcionamiento de OpenFlow está relacionado en su diseño basado en el uso de la tabla de flujo

- **SwitchOpenFlow:** Consta de dos módulos

- ✓ La tabla de flujo que indica cómo cambiar el procesamiento de cada flujo de datos asociando una acción con cada entrada

- ✓ El canal seguro que asegura el paso seguro de las reglas entre el controlador OpenFlow y el switch OpenFlow. [54]

3.2.2. Switch OpenFlow

La función de un Switch OpenFlow, consiste en una o más tabla de reglas de manejo de paquetes (tablas de flujos), de manera que cada regla coincide con un subconjunto del tráfico y realiza ciertas acciones (p. ej.: reenvió, modificaciones) sobre el tráfico. Según su previa instalación dentro del controlador de la aplicación [54]Es de mucha importancia resaltar la capacidad de programabilidad de las SDN, que permite a un Switch OpenFlow, comportarse, como un router, switch, firewall, o llevar a cabo otras funciones como por ejemplo balanceador de carga, minería de datos, entre otras [112].

Proceso del Switch OpenFlow

Inicialmente parte de un paquete en entrada del conmutador Switch OpenFlow, examina directamente la primera tabla, buscando coincidencias. Si las hay, se apuntará la acción definida por la salida, en el orden determinado, en caso de no hallar coincidencias en la tabla de flujos, el paquete será devuelto al controlador, que indicará al Switch qué hacer con él, los diferentes estados de asignación de un paquete mostrados en la Figura 9. Según el Switch OpenFlow son: [54].

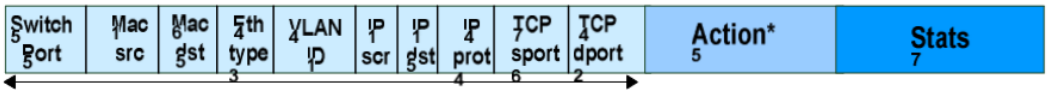


Figura 9. Composición y estados de la tabla de flujo

- Reenviar el paquete a través de un puerto. Esta acción permite enviar los paquetes por medio de la red.
- Encapsular y reenviar el paquete al controlador. Esto sucede cuando tenemos un paquete de un nuevo flujo. De esta manera, el controlador puede decidir si instaurar una nueva entrada en una tabla de flujos y una acción asociada, o si procesar de alguna manera el paquete.
- Eliminar el paquete. Dicho controlador puede actuar como firewall, y bloquear paquetes sospechosos, desconocidos por el mismo.

3.3. Controladores SDN

Dentro de las características principales que ofrece SDN es proporcionar funcionalidad de red mediante el empleo de un controlador lógicamente centralizado que se comunica con dispositivos de red programables a través del protocolo OpenFlow [54] por medio de su Api ((‘Application Programming Interface’) o conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas**))**en dirección norte-sur y con aplicaciones SDN a través de lo que comúnmente se refiere como una interfaz hacia el norte.[69]

Por diseño y desempeño, los controladores SDN tienen control absoluto sobre una red. Se comunican con sus Switches asignados a través de un canal de comando encriptado e implementan acciones de red tales como conmutación de paquetes y enrutamiento con la ayuda de aplicaciones SDN, esto se logra principalmente al proporcionar una vista global de la red y que con las aplicaciones SDN, se pueden tomar decisiones de red.[69]

Dentro de las ventajas que ofrecen los controladores SDN aparte de ser el cerebro y gobernar una red completa se incluyen: [69]

Control de flujo dinámico, lo que permite separar fácilmente los datos maliciosos de tráfico de red benigno.

Visibilidad en toda la red, lo que permite una supervisión simplificada en toda la red; fácil desarrollo de aplicaciones de seguridad de red a través de la programación.

Simplificación del plano de datos, haciendo la adición de módulos de seguridad livianos más fáciles en esta capa.

Como ya hemos comentado anteriormente el controlador centraliza todas las configuraciones de los elementos de la red, su comunicación mediante el protocolo Openflow con los elementos de la red incluyen la configuración de los flujos que pueden ser de dos maneras [58]

• **Proactiva:** Antes de que un paquete llegue al Switch, dando lugar a insignificantes retrasos.

• **Reactiva:** Cuando un Switch recibe un paquete que no coincide con ninguna entrada tiene que enviarlo el controlador y es este quien decide qué hacer con el mismo, sumando 3 retrasos, el de envio, el de procesamiento por el controlador, y el tiempo que se escribe una regla en la tabla de flujo del Switch Open Flow. [58]

En la actualidad existen muchos controladores de red SDN, a continuación se describen algunas de estas herramientas de software y sus principales características. *Tabla 1,* pero para el desarrollo de esta investigación y teniendo en cuenta el objetivo del proyecto se tomó en cuenta el controlador POX.

	BEACON	**FLOODLIGHT**	**NOX**	**POX**	**TREMA**	**RYU**	**ODL**
SOPORTE OPENFLOW	OF V 1.0	OF V 1.0	OF V 1.0	OF V 1.0	OF V 1.3	OF V 1.0, V 1.2, V1.3 Y EXTENSIONES NICIRA	OF V 1.0
VIRTUALIZACION	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Construcción de herramienta virtual de simulación	Mininet y Open vSwitch	Mininet y Open vSwitch
LENGUAJE	Java	Java	C++	Phyton	Rudy/C	Phyton	Java
PROVEE REST API	No	Si	No	No	Si (Basica)	Si(Basica)	Si
INTERFAZ GRAFICA	Web	Web	Phyton+ QT4	Phyton+ QT4 Web	No	Web	Web
SOPORTE DE PLATAFORMAS	Linux, Mac OS Windows y Android	Linux, Mac OS Windows	Linux	Linux, Mac OS Windows	Linux	Linux	Linux, Mac OS Windows
SOPORTE DE OPENSLACK	No	Si	No	No	Si	Si	Si
MULTIPROCESOS	Si	Si	Si	No	Si	No	Si
CODIGO ABIERETO	Si	Si	Si	Si	Si	Si	Si
TIEMPO EN EL MERCADO	4 Años	2 Años	6 Años	1 Año	2 Años	1 Año	5 Meses
DOCUMENTACION	Buena	Buena	Media	Baja	Media	Media	Media

Tabla 1. Controladores SDN y sus características basadas en tabla del artículo [51]

3.3.1. Controlador SDN POX

Es un controlador de red desarrollado a partir de NOX para cubrir las necesidades de las SDN usando Python es funcional en Windows, Mac o Linux. Es uno de los frameworks de desarrollo más creciente. [58]

POX es la primera plataforma pensada para construir aplicaciones que controlen la red en OpenFlow. Está creada para el desarrollo rápido y control de software de la red, a nivel básico, posee un punto clave en cuanto a escalabilidad.[70]

Dentro de sus principales características se tiene:

- Lenguaje de OpenFlow basado en Python
- Detección de topologías de red y selección de rutas de acceso
- Interfaz gráfica para el usuario y componentes de visualización
- Desarrollado para el manejo del protocolo OpenFlow en versión

Comprendiendo el concepto de POX y Mininet es posible desarrollar componentes en POX y usarlos en Mininet, pero para hacerlo, es necesario instalarlos en su máquina virtual Mininet o en una máquina que aloje la máquina virtual VM Mininet [50]. En la siguiente *Figura 10. S*e muestra un escenario de simulación que consiste en un solo Switch OpenFlow (S1) conectado a tres hosts (h2, h3, h4) y a un controlador POX que tiene creados dos componentes llamados Hub y Switch

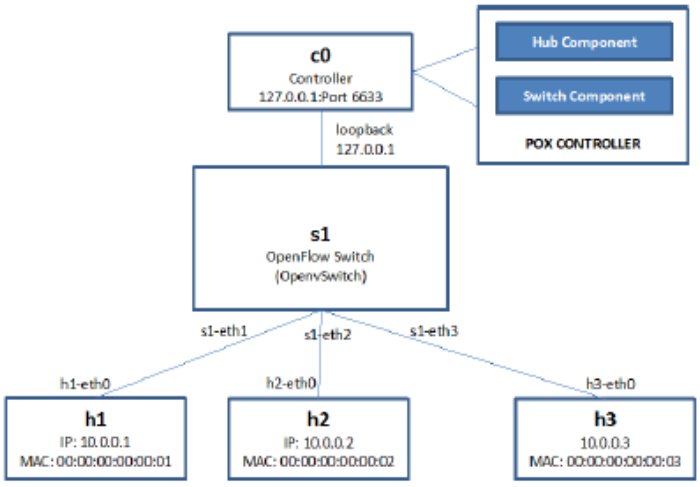


Figura 10. Escenario de simulación de red SDN con controlador POX

Componentes de POX

A continuación en la *Tabla 2.* Se describen algunos de los componentes que contiene POX

COMPONENTE	***DESCRIPCION***
Py	Es el encargado de arrancar el intérprete de Python para la depuración y ajuste interactivo.
Switch Component	Este componente tiene una acción de autoaprendizaje, aprende la ruta a medida que los paquetes llegan al Switch, en otras palabras, es un simple interruptor que aprende las formas de sus anfitriones
Hub Component	Componente que actúa como un centro, replicando paquetes recibidos y enviados a todos sus puertos de salida
forwarding.l2_learning:	Permite que los switches OpenFlow actúen como un switch de capa 2.
forwarding.l2_pairs	Este componente permite que los *switches* OpenFlow actúen como un tipo L2, con la diferencia de que instala reglas basadas exclusivamente en las direcciones MAC.
forwarding.l3_learning	Este componente no define el comportamiento de un *router*, pero tampoco es un *switch* de nivel 2. POX usa una biblioteca de paquetes para examinar y elaborar solicitudes y respuestas ARP.
forwarding.l2_multi	Este componente se puede ver como un switch de aprendizaje. El aprendizaje de los otros switches se realiza sobre una base de conexión de switch a switch, tomando las decisiones de conmutación, como si cada switch tuviera solo información local, para aprender la topología de toda la red.
openflow.spanning_tree	Este componente utiliza el componente de descubrimiento para construir una vista de la topología de la red, construye un árbol de expansión41, desactivando los *floodings* en los puertos del *switch* que no están en el árbol para que las topologías queden libres de bucles.
web.webcore	Este componente inicia un servidor web dentro del proceso POX. Otros componentes pueden interactuar con él para proporcionar su propio contenido web estático y dinámico.
messenger	Este componente proporciona una interfaz para interactuar con los procesos externos bidireccionales a través de mensajes basados en JSON42. Es una API para la comunicación a través de medios de transporte usando los *sockets* TCP y HTTP.
misc.arp_responder	Es un componente con el que se puede responder las peticiones ARP.
misc.dns_spy	Este componente supervisa las respuestas DNS y muestra sus resultados.
misc.mac_blocker	Este componente está destinado a ser utilizado junto con algunas aplicaciones de reenvío, tales como *l2_learning* y *l2_pairs*. Abre una interfaz gráfica de usuario que permite bloquear direcciones MAC.
openflow.of_01	Este componente se comunica con *switches* OpenFlow 1.0. Por lo general, se inicia de forma predeterminada (a menos que se especifique la opción *no-OpenFlow* en la línea de comandos).
openflow.discovery	Este componente envía mensajes LLDP de conmutadores OpenFlow para que pueda descubrir la topología de la red.
openflow.debug	Cargando este componente hará que POX pueda crear trazas con mensajes OpenFlow, que luego se pueden cargar en Wireshark para ser analizados.

openflow.keepalive	Este componente hace que POX pueda enviar solicitudes periódicas *echo* a los *switches* conectados

Tabla 2. Componentes del controlador POX basados en tablas de artículos [50] [58]

3.4. Entornos de análisis y monitoreo en redes SDN

Partiendo del hecho de que hoy en día los proveedores SDN ofrecen estándares con los cuales las soluciones calculan e implementan los respectivos planes de control de rendimiento de datos, no se puede dejar de lado a los proveedores de software de monitoreo convencionales que también pueden evolucionar, realizar recolección de datos e integrarlos centralizadamente, además en el mercado hay soluciones que cuentan con herramientas de análisis y gestión de tráfico, permitiendo que se tenga monitoreo y visibilidad total y los movimientos de la red.[71]

Actualmente la mayoría de soluciones de monitoreo que se ofertan son flexibles y modifican los estándares de manera amplia, además de integrar el ambiente virtual con la gestión global. Sin embargo, la transición SDN no es tan fácil como se piensa, primero que todo quien esté interesado en implementar SDN, debe tener muy claro cómo va a mantener el control, cuando eso pase y se tengan las estrategias de gestión bien establecidas se podrá asumir que se está preparado para las SDN. [77]

Para el desarrollo de la solución en esta investigación se manejaran los siguientes entornos de monitoreo.

3.4.1. OpenNetMon

Opennetmon es una sistema de monitoreo que funciona como un módulo para un controlador SDN de tipo OpenFlow. Su función es monitorear continuamente todos los flujos entre pares predefinidos de enlace-destino en el rendimiento, pérdida de paquetes y demora, se cree que una monitorización tan exhaustiva y en tiempo real de sistema es esencial para los propósitos de ingeniería y vigilancia de tráfico. [67]

Opennetmon está basado en el enfoque de monitoreo activo que proporciona la supervisión necesaria para determinar si los parámetros de calidad del servicio QoS de extremo a extremo en realidad se cumplen [67] su propósito es reducir el uso de la CPU, este sistema consulta las estadísticas de información para los Switches de destino con una tasa de sondeo adaptativa, aunque OpenNetMon

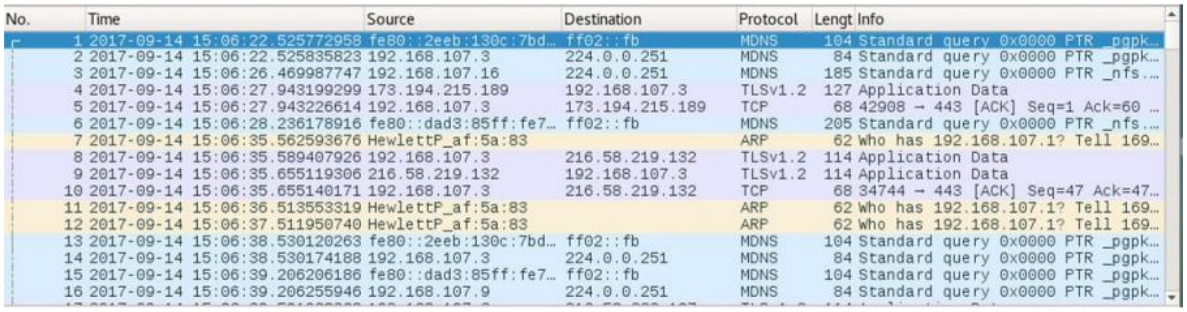
reduce el uso de la CPU, [72] este sistema solo se enfoca en los Switches de borde es decir conmutadores con puntos finales de flujo conectados, por lo tanto, es imposible para conocer la información estadística de los otros conmutadores, aunque no posee un enfoque heurístico la tasa de adaptación reduce la red y conmuta la sobrecarga de la CPU mientras sigue optimizando la precisión de la medición [67] de este modo pasivo no se provoca sobrecarga de monitoreo. [72]

Del mismo modo OpenNetMon tambien puede informar en tiempo real si el tiempo de recorrido del paquete de sonda sobre la ruta de extremo a extremo es menor o igual que tiempo de ráfaga de tráfico [73] La naturaleza adaptativa de OpenNetMon también podría ser eficiente para evitar el salto excesivo cuando los flujos son reasignados en base a una nueva vista detallada del estado de la red, por tal razón OpenNetMon no pre calcula las rutas, la calcula en línea cuando se necesitan en un camino de múltiple entorno.[67]

3.4.2. Wireshark

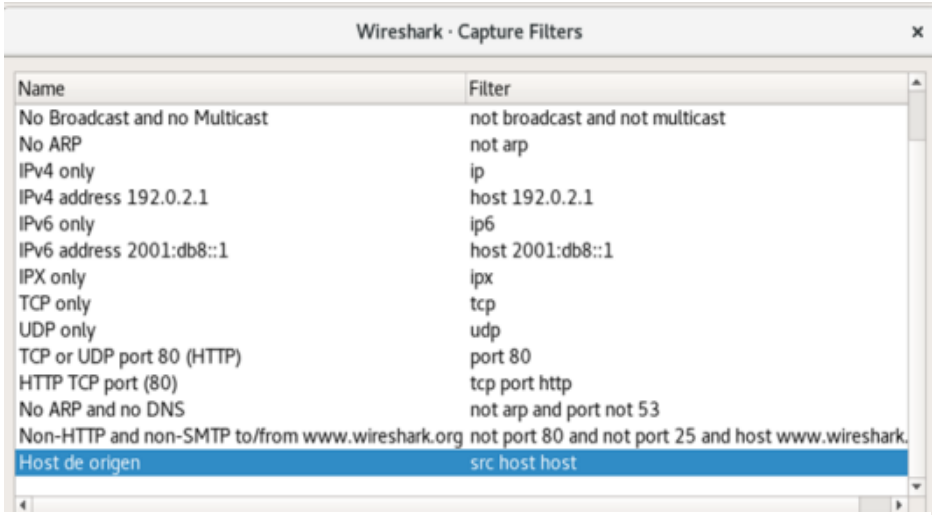
Wireshark es uno de los analizadores de protocolos de red más utilizado que hay actualmente [53], se podría pensar en un analizador de paquetes de red como un dispositivo de medición utilizado para examinar lo que está sucediendo dentro de un cable de red, al igual que un voltímetro es utilizado por un electricista para examinar lo que está sucediendo dentro de un cable eléctrico pero en un nivel más superior, [75] 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. [58]

El enfoque de esta herramienta está dirigido hacia el análisis del tráfico de las redes con herramientas que apoyen el trabajo del diseñador, analizador, ingeniero, consultor y administrador de la red, permite ver lo que está sucediendo en la red con un gran nivel de detalle. [53] es multiplataforma y se usa en distintos sistemas operativos como Windows, Linux, ios. [74]



No.	Time	Source	Destination	Protocol	Lengt	Info
1	2017-09-14 15:06:22.525772958	fe80::2eeb:130c:7bd…	ff02::fb	MDNS	104	Standard query 0x0000 PTR _pgpk…
2	2017-09-14 15:06:22.525835823	192.168.107.3	224.0.0.251	MDNS	84	Standard query 0x0000 PTR _pgpk…
3	2017-09-14 15:06:26.469987747	192.168.107.16	224.0.0.251	MDNS	185	Standard query 0x0000 PTR _nfs.…
4	2017-09-14 15:06:27.943199299	173.194.215.189	192.168.107.3	TLSv1.2	127	Application Data
5	2017-09-14 15:06:27.943226614	192.168.107.3	173.194.215.189	TCP	68	42908 → 443 [ACK] Seq=1 Ack=60 …
6	2017-09-14 15:06:28.236178916	fe80::dad3:85ff:fe7…	ff02::fb	MDNS	205	Standard query 0x0000 PTR _nfs.…
7	2017-09-14 15:06:35.562593676	HewlettP_af:5a:83		ARP	62	Who has 192.168.107.1? Tell 169…
8	2017-09-14 15:06:35.589407926	192.168.107.3	216.58.219.132	TLSv1.2	114	Application Data
9	2017-09-14 15:06:35.655119306	216.58.219.132	192.168.107.3	TLSv1.2	114	Application Data
10	2017-09-14 15:06:35.655140171	192.168.107.3	216.58.219.132	TCP	68	34744 → 443 [ACK] Seq=47 Ack=47…
11	2017-09-14 15:06:36.513553319	HewlettP_af:5a:83		ARP	62	Who has 192.168.107.1? Tell 169…
12	2017-09-14 15:06:37.511950740	HewlettP_af:5a:83		ARP	62	Who has 192.168.107.1? Tell 169…
13	2017-09-14 15:06:38.530120263	fe80::2eeb:130c:7bd…	ff02::fb	MDNS	104	Standard query 0x0000 PTR _pgpk…
14	2017-09-14 15:06:38.530174188	192.168.107.3	224.0.0.251	MDNS	84	Standard query 0x0000 PTR _pgpk…
15	2017-09-14 15:06:39.206206186	fe80::dad3:85ff:fe7…	ff02::fb	MDNS	104	Standard query 0x0000 PTR _pgpk…
16	2017-09-14 15:06:39.206255946	192.168.107.9	224.0.0.251	MDNS	84	Standard query 0x0000 PTR _pgpk…

Figura 12. Captura de paquetes Wireshark



Wireshark · Capture Filters

Name	Filter
No Broadcast and no Multicast	not broadcast and not multicast
No ARP	not arp
IPv4 only	ip
IPv4 address 192.0.2.1	host 192.0.2.1
IPv6 only	ip6
IPv6 address 2001:db8::1	host 2001:db8::1
IPX only	ipx
TCP only	tcp
UDP only	udp
TCP or UDP port 80 (HTTP)	port 80
HTTP TCP port (80)	tcp port http
No ARP and no DNS	not arp and port not 53
Non-HTTP and non-SMTP to/from www.wireshark.org	not port 80 and not port 25 and host www.wireshark.
Host de origen	src host host

Figura 11. Filtros de captura Wireshark

Wireshark cuenta con características muy importantes descritas en diversos paneles de vista y listados de paquetes como se muestra en la *figura 012*, que no son muy complicados de manejar y que hacen de las actividades del Snifer y la visualización de red una verdadera experiencia [74], el funcionamiento esta dado a partir de la visualización del tráfico de la red por medio de estadísticas, paneles y gráficos y su potencial se enfoca en la captura de datos de red que se obtiene a partir de filtros de captura, *figuras 11 y 12*

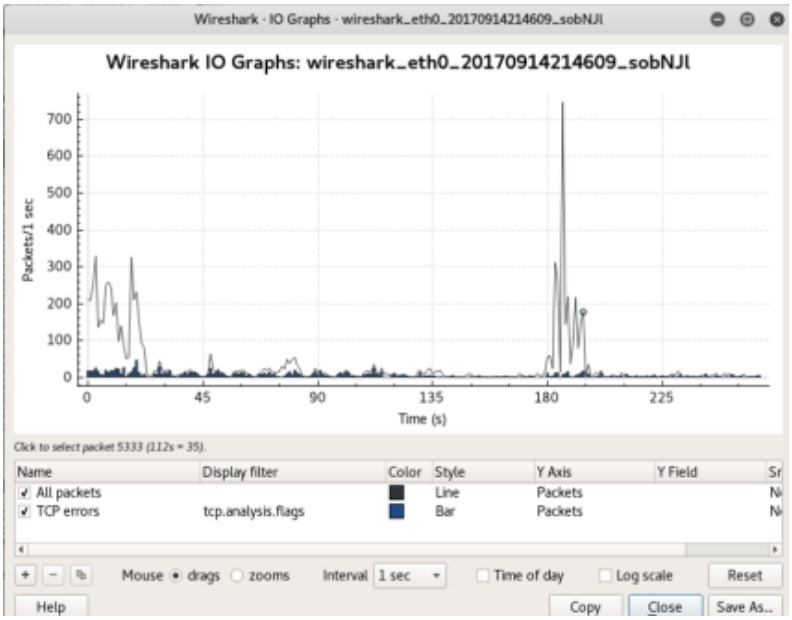


Figura 13. Gráfico de trafico opción I/O graphs Wireshark

Algunas de las importantes características a destacar de Wireshark son; su robustez, su capacidad para capturar datos de la red o leer datos almacenados en un archivo, basado en la librería pcap con una interfaz muy flexible, con gran capacidad de filtrado, admite entre otros el formato estándar de archivos tcpdump, permite la reconstrucción de sesiones TCP, se ejecuta en diferentes plataformas, es un opulento analizador VoIP, y los datos de captura pueden ser leídos en tiempo real, dependiendo de la plataforma, desde Ethernet [53], puede Importar paquetes de archivos de texto que contienen vuelcos hexadecimales de paquetes de datos, mostrar paquetes con información de protocolo muy detallada, guardar los datos del paquete capturados, exportar algunos o todos los paquetes en varios formatos de archivo de captura, colorizar la visualización de paquetes basada en filtros, crear varias estadísticas entre otras. [74]

El analizador de software Wireshark, se puede configurar para trabajar con distintas interfaces y se aplican el filtro de protocolo “openflow_v1” para que trabaje con OpenFlow [70].
Para el ver tráfico de control de OF ejecutamos “*$ sudo wireshark &”* [58].

3.5. Jitter

La inestabilidad o fluctuación de fase es un problema para las redes de datos a pequeña y gran escala que debe controlarse para garantizar que los sistemas funcionen sin problemas, si bien es casi imposible evitar esa inestabilidad, con soluciones adecuadas y un buen sistema de monitoreo, se puede reducir y las empresas podrían minimizar el impacto de las interrupciones en la red y reducir el desperdicio y el tiempo de inactividad [78]

En este escenario se tratara de presentar un componente inesperado de red llamado Jitter, que causa interrupciones en el proceso normal de una red de datos y para el cual se buscaran unas posibles soluciones.

El Jitter se define técnicamente como la variación o fluctuación en el tiempo de llegada o retardo de los paquetes en un cierto plazo de punto a punto o una desviación de fase respecto de la posición ideal en el tiempo de una señal digital que se propaga en un canal de transmisión.[79] causada por congestión de red, perdida de sincronización o por las diferentes rutas seguidas por los paquetes para llegar al destino [76].

El Jitter es un efecto completamente indeseable en cualquier sistema de comunicaciones y por ende introduce una serie de problemas al canal, que de no ser tratado adecuadamente puede degradar completamente la calidad y desempeño del enlace[79] En los sistemas de control, los paquetes se envían idealmente en una secuencia continua y espaciada uniformemente, sin embargo, la congestión de la red, los errores de configuración u otros problemas con un sistema como una cola incorrecta pueden causar inestabilidad como lo muestra la *figura 017* [78] si el retardo de las transmisiones varía demasiado extensamente en una llamada VoIP (Voice Over Internet Protocol o Protocolo de voz a través de internet.), la calidad de la trasmisión se degrada grandemente. [80]

El Jitter puede causar errores en la recepción de bits ya que si no es controlado confundirá al receptor y éste no podrá recobrar el reloj de sincronismo en el extremo receptor, además puede producir interferencia intersimbólica (ISI), entre los pulsos que se propagan por el canal, ya que el Jitter producirá un desplazamiento de las señales que componen el pulso *Figura 14*, y por ende se mezclarán, imposibilitando de esta manera el reconocimiento de los niveles respectivos de la señal en el receptor. [79]

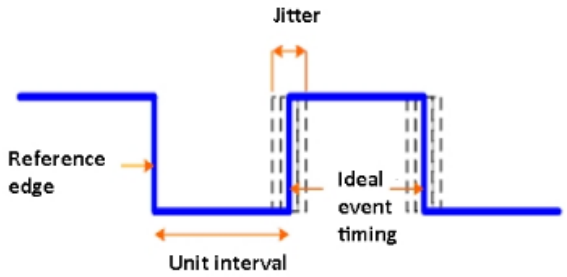


Figura 14. Diferencia absoluta posición del borde reloj o señales

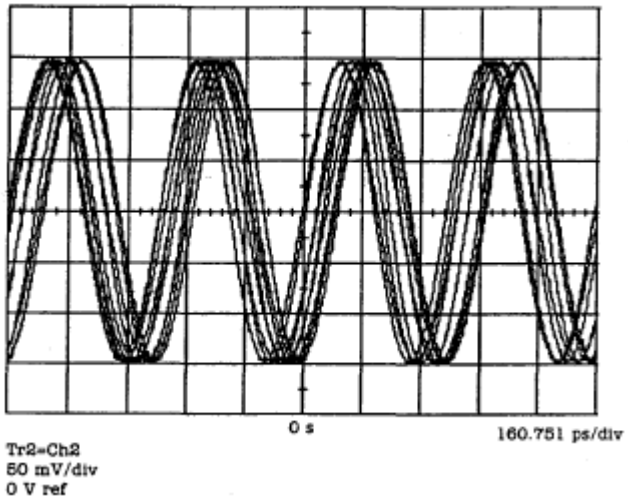


Figura 15. Efecto Jitter de 2.48832 GHz, en señal de reloj – Representación del dominio del tiempo (Hewlett Packard Journal – Copiright 1995)

En la *Figura 15*, Se puede evidenciar el efecto del Jitter en una señal sinusoidal que se propaga por un canal y experimenta sucesivos cambios en su fase, lo que genera un ensanchamiento de dicha señal. Obviamente este efecto es indeseable, ya que al aumentar imposibilita la identificación de la señal transmitida [79]

El Jitter es un efecto de las redes de datos no orientadas a conexión y basadas en conmutación de paquetes, como la información se discretiza en paquetes cada uno de los paquetes puede seguir una ruta distinta para llegar al destino [78], de este modo se toma el jitter como la vibración total en cada instante de muestreo, y puede ser la suma de varios tipos de fluctuación de tiempo [82], esta fluctuación es un factor crucial en redes de alta velocidad y aplicaciónes de rendimiento. [82]

En función, Jitter significa que la información puede llegar a su destino en diferentes momentos o incluso en un orden diferente al previsto, parte de la información llega más rápido, mientras que otros paquetes de datos llegan más despacio. [78] El jitter también puede causar la pérdida de paquetes, que es

cuando la información desaparece, esto sería como una llamada que cae de repente en mitad de una conversación o se escuchan problemas de audio,cla conexión pierde su señal. En una configuración industrial, las sobrecargas o interrupciones de la red pueden provocar la desconexión completa de un dispositivo del sistema. [78]

La inestabilidad es definitivamente un problema en el campo de las redes así como como en la industria, y es uno que debe y puede controlarse, un ejemplo claro está dado cuando nos imaginamos en un escenario un tren con sus respectivos vagones simulando cada paquete de red o mensaje, una red típica no puede procesar el próximo vagón de tren sin antes tratar con el que está frente a él *(Figura 16),* independientemente de cuán importante pueda ser el próximo en la

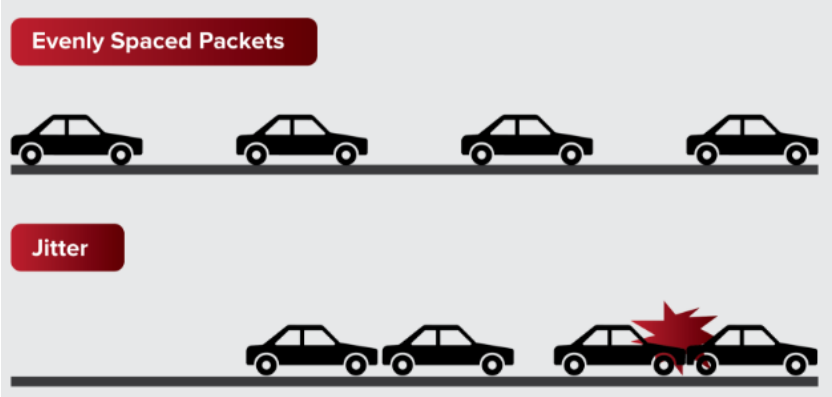


Figura 16. Comparación trafico uniforme de paquetes *con la fluctuación del retardo de llegada (Jitter)*

cola, no habría como continuar con el proceso; Sin embargo podría existir un canal de prioridad donde se puede interrumpir ese vagon típico e identificar el próximo como de mayor prioridad, enviándolo a través del sistema antes de procesar la información menos crucial.
Para resolver el inconveniente al ejemplo propuesto, la solución más ampliamente adoptada es la utilización del **Jitter buffer** que consiste básicamente en asignar una pequeña cola o almacén para ir recibiendo los paquetes y sirviéndolos con un pequeño retraso. Si algún paquete no está en el buffer (se perdió o no ha llegado todavía) cuando sea necesario se descarta. Normalmente en los teléfonos IP (hardware y software) se pueden modificar los buffers. Un aumento del buffer implica menos perdida de paquetes pero más retraso. Una disminución implica menos retardo pero más pérdida de paquetes. [76] El búfer de retardo también se denomina a veces memoria intermedia de fluctuación de fase (Jitter) [81]

El retardo, el jitter, y las medidas de la pérdida del paquete pueden entonces ayudar en el diseño y la configuración correctos de la priorización del tráfico, así

como mitigar los parámetros en el equipo de red de datos para que se mantenga estable la conexión el jitter entre el punto inicial y final de la comunicación **debiera ser inferior a 100 ms**. si el valor es menor a 100 ms el jitter puede ser compensado de manera apropiada. En caso contrario debiera ser minimizado. [80]

Antes de las aplicaciones VoIP que despliegan, es importante evaluar el retardo, el jitter, y la pérdida del paquete en la red de datos para determinar si las Aplicaciones de voz trabajan. La cantidad de jitter tolerable en la red es afectada por la profundidad del buffer del jitter en el equipo de red en el trayecto de la voz. Más el buffer del jitter disponible, más la red puede reducir los efectos del jitter.
La pérdida del paquete está perdiendo los paquetes a lo largo del trayecto de datos, que degrada seriamente la aplicación de voz. [80]

Tipos de Jitter

El Jitter puede ser limitado o no limitado. El Jitter limitado está relacionado con la frecuencia y magnitud de los eventos del sistema; por lo tanto, este es determinístico. Esto significa que al deshabilitarse la fuente se detendrá el jitter limitado también, por otra parte el jitter no limitado no depende de los eventos. Puede ser causado por los componentes del sistema o influencias externas [83]

Existen varios tipos de tipos de Jitter que pueden influir en sistemas de datos de alta velocidad, en la siguiente *Figura 17*, se evidencia el árbol de ascendencia y como están distribuidos según su criterio.

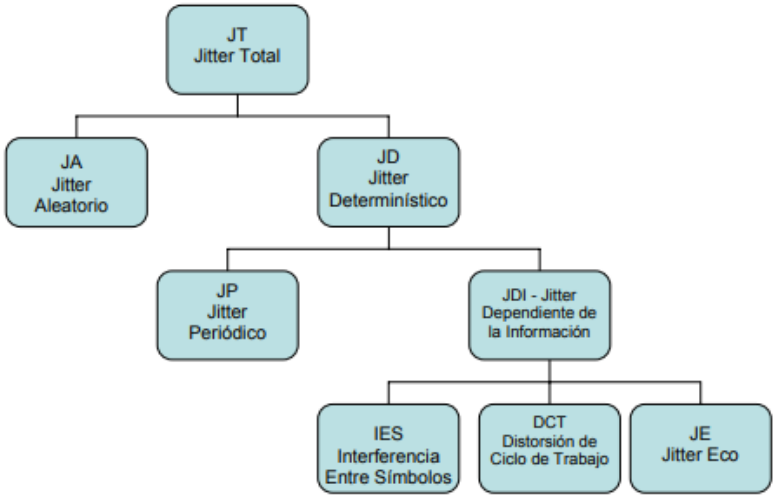


Figura 17. Árbol de tipos de Jitter

En la siguiente *Tabla 3,* se describen uno a uno los tipos de Jitter que influyen en los distintos sistemas.

TIPO DE JITTER	*DESCRIPCION*
JT **Jitter Total**	El Jitter total es el valor pico a pico obtenido. JT = JD + n × JA donde n = número de desviaciones estándar correspondientes al BER (Tasa de error de bit) requerido, La suma del Jitter determinístico y el aleatorio.
JA **Jitter Aleatorio**	La fuente principal el ruido (blanco) Gaussiano dentro de los componentes del sistema. Interactúa con la velocidad muerta de las señales y produce errores de temporización en los puntos de conmutación.
JD **Jitter Determinístico**	Jitter con función de densidad de probabilidad no Gaussiana. Está siempre limitada en amplitud y con causas específicas. Las fuentes son imperfecciones de los dispositivos, crosstalk, IEM, problemas de aterramiento.
JP **Jitter Periódico**	También llamado Jitter Senoidal debido a su forma senoidal. La fuente es generalmente señales en forma de interferencia relacionadas con el patrón de datos, variaciones de aterramiento o variaciones en la fuente de alimentación.
JDI **Jitter Dependiente de la Información**	Consiste en la Interferencia Entre Símbolos (IES), Distorsión del Ciclo de Trabajo (DCT), y el Jitter Eco (JEC). Los errores de tiempo varían con el patrón de datos. La fuente primaria son los componentes y las limitaciones del ancho de banda.
IES **Interferencia Entre Símbolos**	La Interferencia Entre Símbolos es la forma más común de JDI. Es causada generalmente por limitaciones del ancho de banda en las líneas de transmisión. Afecta a bits individuales rodeados por bits del estado opuesto.
DCT **Distorsión de Ciclo de Trabajo**	El Jitter por Distorsión del Ciclo de Trabajo es causado cuando ciertos estados de bit tienen diferente duración. “1” siempre es más largo que “0” o viceversa. Causado por la configuración del bias, e insuficiente alimentación continúa de un componente.
JEC **Jitter Eco**	El Jitter Eco es causado por incompatibilidad entre el componente y la línea, depende del patrón de datos. La longitud de línea influye también en la magnitud del JEC.

Tabla 3. Tipos de Jitter

3.5.1. Como Calcular el JItter

La medición del Jitter se determina a partir de la variación de la demora *(Delay).* Dependiente en la aplicación, un gran valor de jitter es un problema que compromete en gran medida el rendimiento de la aplicación o en la trasmisión.

Como tenemos la Ecuación $(Ds1\text{-}s2 = TravelTime\ \text{-}Dc\text{-}s1\text{-}Dc\text{-}s2)$, para calcular el retraso, necesitamos almacenar tanto la demora anterior como la actual para calcular su variación en un intervalo de sondeo $[i, i\text{-}1]$. La siguiente ecuación calcula el tiempo actualizado, considerando el instante i y el instante anterior $(i\text{-}1)$. El retraso en este intervalo es $D\ (i\text{-}1, i)$. Que sería igual al Jitter. [39]

$$J(i) = j(i-1) + (|D\ (i-1, i)\ | - J\ (i-1))\ /\ 16$$

Frecuencia del Jitter:

La frecuencia de Jitter (Fj) de la oscilación básica se calcula del siguiente modo:

$$Fj = 1/Th$$

Costo del Jitter:

Según [39], un cambio en el retraso debe repetirse varias veces para influir en el tiempo estimar significativamente, por esta razón debemos tener en cuenta que para el cálculo del costo de Jitter no podemos solo sumar los valores de retraso cuando el total de una ruta no se puede definir exactamente como suma del valor de cada enlace, porque al hacerlo asume que cada enlace tiene una variación de retraso del mismo valor y no es así, Sin embargo, la suma de los valores de Jitter es un buen enfoque para evitar caminos con volúmenes potencialmente grandes, de este modo se asume que el costo del parámetro Jitter está dado a partir de la siguiente *ecuación [39].*

$$JitterCost(j) = Jj\ /\ i)$$

Calculo del Jitter Periódico:

Donde ***TP (1)*** es el periodo de oscilación de la primera oscilación después del evento iniciador y ***T0*** el periodo de oscilación ideal.

$$JP = TP\ (1) - T0$$

Calculo del Jitter Total:

El Jitter total es el valor pico a pico obtenido, la suma del Jitter determinístico y el aleatorio. Donde n = número de desviaciones estándar correspondientes al BER (Tasa de error de bit) requerido [83]

$$JT = JD + n \times JA$$

CAPÍTULO 4. INSTALACION DE LA RED SDN

Las tareas de simulación, instalación y pruebas de la red SDN, se realizaron en una red de datos virtualizada bajo la tecnología de MiniNet que ofrece esta posibilidad como se ha descrito en el Capítulo 3.

Dentro del plan de pruebas, inicialmente se realizaron simulaciones de tráfico de red a través de conexión ssh con la máquina virtual que contiene MiniNet con la red simulada, que a su vez aloja el controlador POX

4.1. Descripción de la red

4.1.1. Mapa de la red

En la figura se muestra el mapa general de la red. La cual describe una topologia de estrella, donde se pueden apreciar:

- 2 hosts virtuales, cada uno con una dirección IP separada.
- 1 solo switch de software OpenFlow en el kernel con 2 puertos.
- Cada host virtual está conectado al switch con un cable de ethernet virtual.

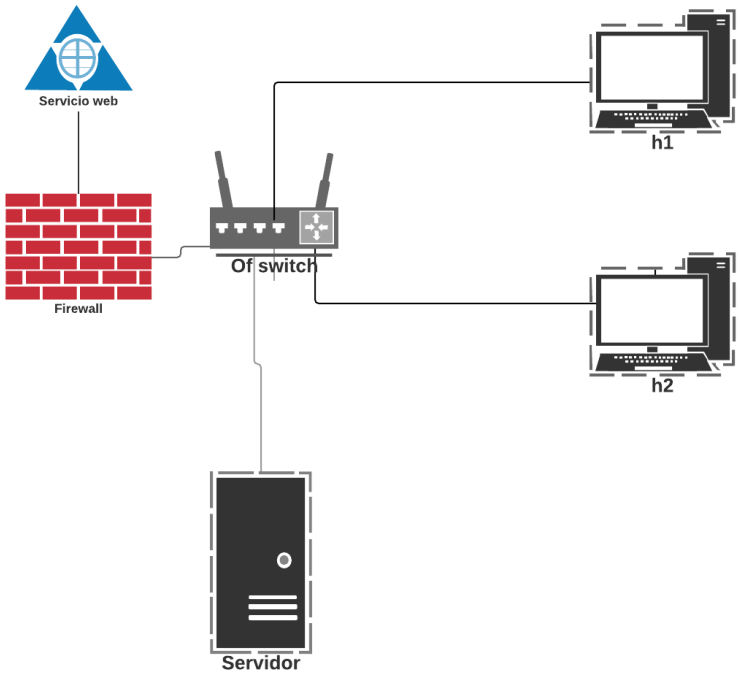


Figura 18. Diagrama de red

4.1.2. Configuración de la red

La red es una red privada de clase a, por lo cual está configurada de la siguiente manera:

Red Privada - Red de Área Local

- ✓ Dirección de red 10.0.0.0/8
- ✓ Mascara de red 255.0.0.0
- ✓ Puerta de enlace 10.0.0.0

4.1.3. Esquema de direcciones ip

Las direcciones IP están configuradas se encuentran distribuidas entre los distintos host con las siguientes configuraciones:

Dirección ip	*Host*
10.0.0.1	Host 1 (h1)
10.0.0.2	Host 2 (h2)

Tabla 4. Esquema de direcciones ip

4.2. Configuración de entornos SDN

En el momento de realizar la simulación del comportamiento de la red, tanto de los hosts como del switch of bajo la tecnología SDN, se realizó con virtual box la cual tiene una licencia pública (GPL) versión 2. El cual es un producto de virtualización, tanto para uso empresarial como domestico permitiendo instalar sistemas operativos adicionales (sistemas invitados) dentro de otro sistema operativo (anfitrión). Además permite emular las características necesarias que son requeridas para el servidor virtual donde se aloja MiniNet quien permite la creación y virtualización de la red anteriormente mencionada.

4.2.1. Software utilizado

Nombre	*Versión*	*Descripción*
Virtual box	5.2	Software de virtualización para arquitecturas x86/amd64
MiniNet	2.2.2	MiniNet permite crear una red virtual realista, ejecutando código real de kernel, switch y aplicación, en

		una sola máquina
Wireshark	2.6.2	Es una analizador de trafico
Pox		Es una plataforma de software de red escrita en Python
Xming	7.5.0.6	Es una implementación portátil del sistema de ventanas X para sistemas operativos Microsoft Windows
Putty	0.70	Es un cliente de SSH y telnet
OpenNetMon		Ofrece un módulo para Pox, que permite la monitorización.

Tabla 5. Software utilizado

4.2.2. Configuración máquina virtual

Para la creación de la red se hace necesario la instalación de un sistema operativo "invitado" que permita alojar mininet, por lo cual se utiliza la herramienta: virtual box. Esta herramienta facilita la instalación de sistemas operativos (SO), ya sea en Windows, Linux o Mac, en este caso Ubuntu 14.04-LTS – 64bits un SO con kernel en linux. Es necesario ya que MiniNet corre sobre arquitectura Linux.

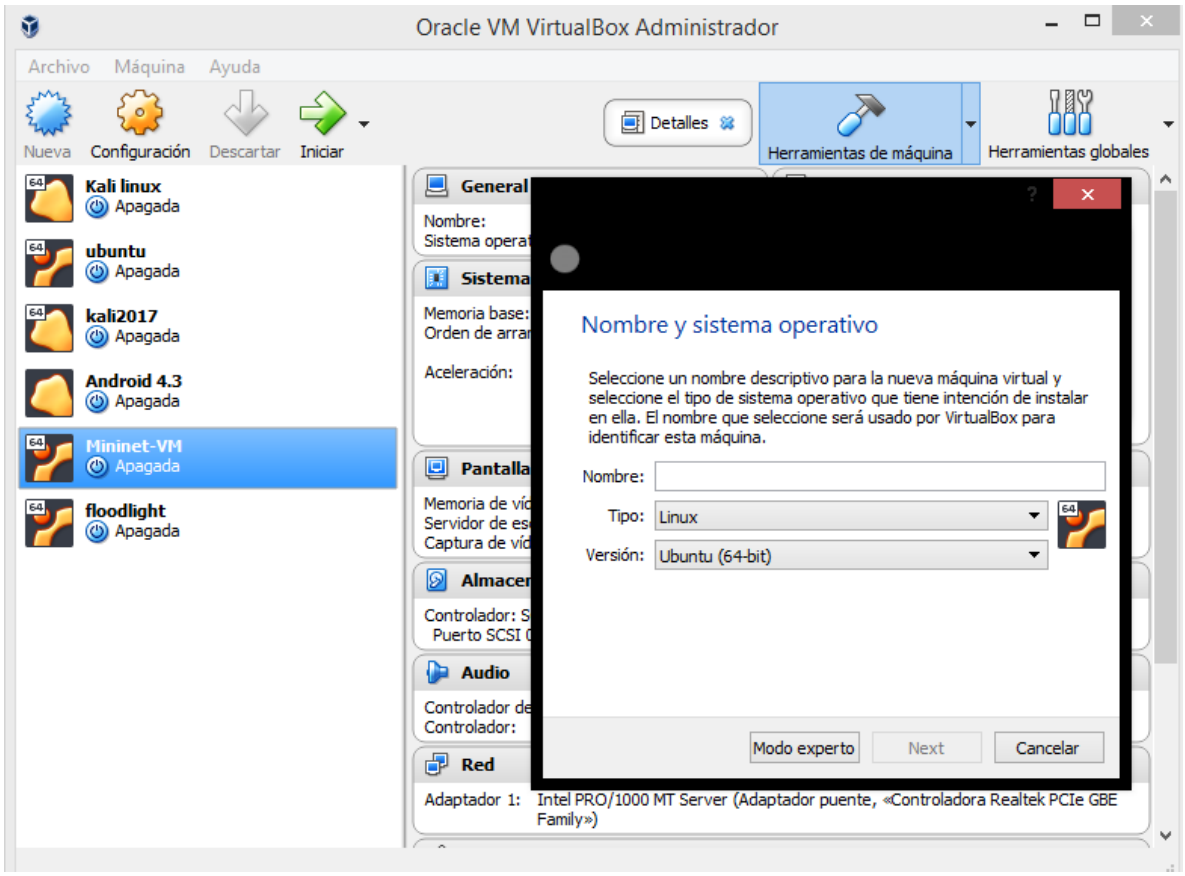


Figura 19. Máquina virtual con mininet

En la *Figura 19,* se puede apreciar cómo se ha creado con el nombre de mininet el sistema operativo requerido para el desarrollo del actual proyecto, al cual se le ha asignado una memoria RAM de 1024 Mb y un espacio de 8gb. El sistema operativo instalado (Ubuntu 14.04) carece de interfaz gráfica por lo cual se maneja solamente por terminal de comandos *Figura 20*.

Figura 20.Inicio de sesión Ubuntu 14.04

Además es necesario configurar el tipo red que manejara virtual box para este SO, para ello virtual Box nos permite escoger entre los siguientes modos de conexión *Figura 21*

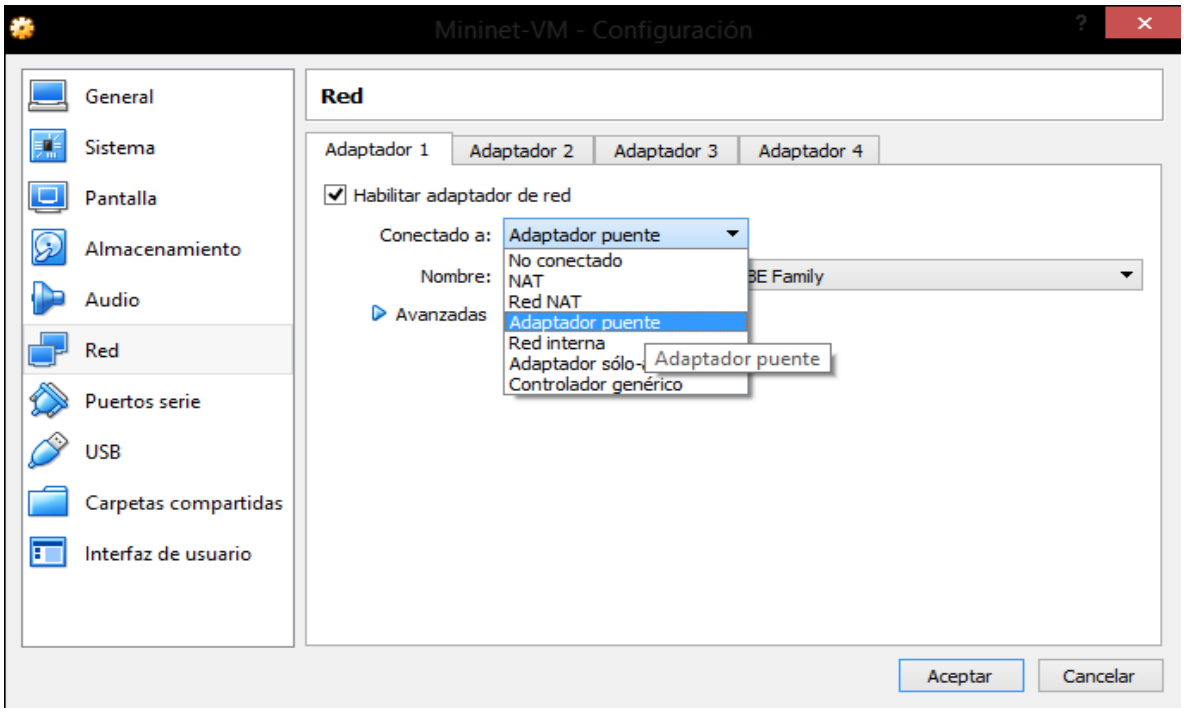


Figura 21. Configuración de red Virtual Box

Cada modo de conexión permite hacer lo siguiente:

- **No conectado:** Muestra un adaptador de red pero sin conexión. (cable desconectado)

- **Network Address Translation (NAT):** Fue desarrollado para solucionar el problema de la escasez de direcciones IP de forma que redes de ordenadores utilicen un rango de direcciones especiales (IP privadas) y se conecten a Internet usando una única dirección IP (IP pública), de esta forma varios equipos se conectan a internet con una única IP pública.

- **Adaptador puente:** Simula una conexión física real a la red, asignando una IP al sistema operativo invitado. Esta IP se puede obtener por DHCP o directamente configurándola en el Sistema Operativo invitado.

- **Red interna:** Similar al Adaptador puente, se puede comunicar directamente con el mundo exterior con la salvedad de que ese mundo exterior está restringido a las máquinas virtuales conectadas en la misma red interna. Esta limitación viene justificada por seguridad y velocidad.

- **Adaptador sólo-anfitrión:** Es una mezcla entre los tipos “Adaptador

puente” e “interna”.

Se escoge adaptador puente porque al tener una ip propia se independiza del equipo anfitrión dando la facilidad de acceder mediante una conexión ssh, como se apreciara más adelante.

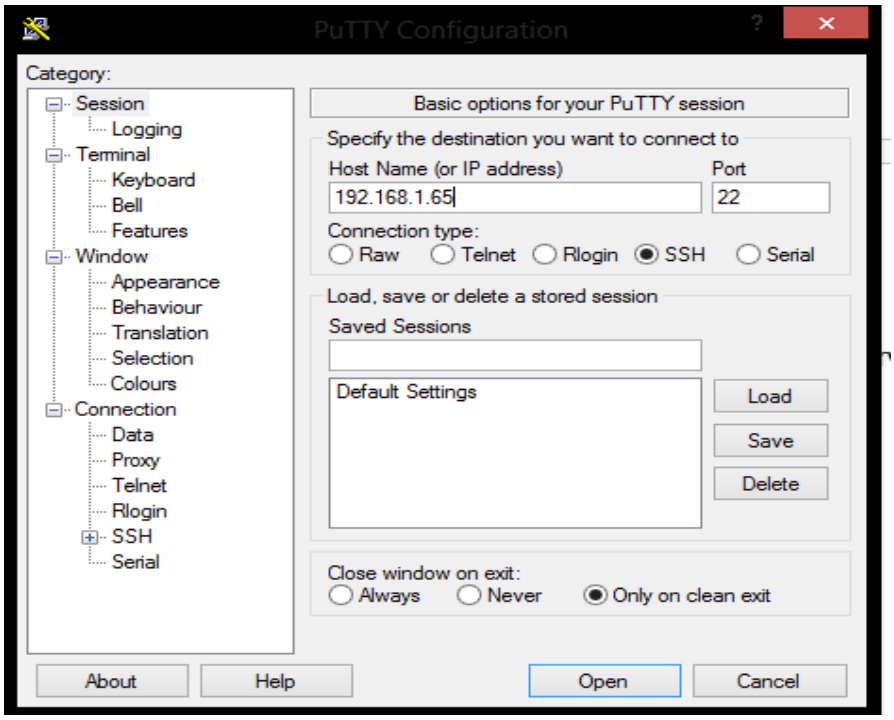
4.2.3. Entorno de simulación

Como se mencionaba anteriormente, para la simulación se hace necesario el uso de un cliente ssh, desde el cual podamos acceder a servidor donde está alojado MiniNet y aplicar los comandos requeridos, que permitan hacer un buen trabajo con la simulación. Para este caso se usara PuTTY; esta herramienta permite la conexión ssh desde un sistema operativo Windows.

```
mininet@mininet-vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:37:b2:d6
          inet addr:192.168.1.65  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:217 errors:0 dropped:0 overruns:0 frame:0
          TX packets:144 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24319 (24.3 KB)  TX bytes:12824 (12.8 KB)
```

Figura 22. Dirección ip SO Ubuntu (cliente mininet)

En la *Figura 22* se evidencia la configuración de red que tiene el servidor sobre el



cual esta mininet donde a su vez se aloja el controlador POX.

Figura 23. Configuración PuTTY

La *Figura 23,* muestra cómo se debe configurar el cliente SSH, donde se agrega la ip del servidor, posteriormente se configura el reenvió x11 donde solo debe habilitarse la casilla subrayada *Figura 24.* Y será posible conectar con el servidor.

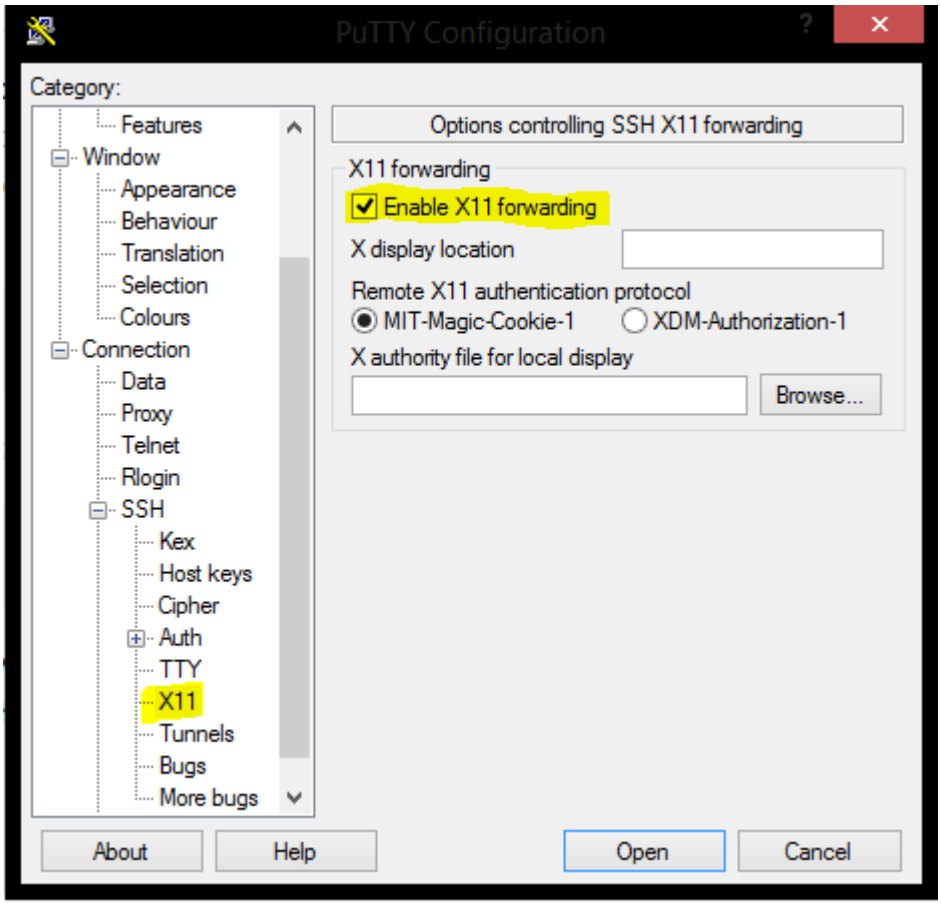


Figura 24. Cliente PuTTY X11

Como se puede apreciar en la *Figura 25*. El servidor cuenta con MiniNet, openflow y el controlador POX. Sin embargo, MiniNet cuenta con un repositorio en GitHub esto hace posible instalar en cualquier momento la herramienta en caso de no contar con ella. También el controlador Pox, cuenta con un repositorio en Github, que permite tanto descargarlo como actualizarlo si es necesario. Los controladores SDN por lo general hacen uso del puerto 6633.

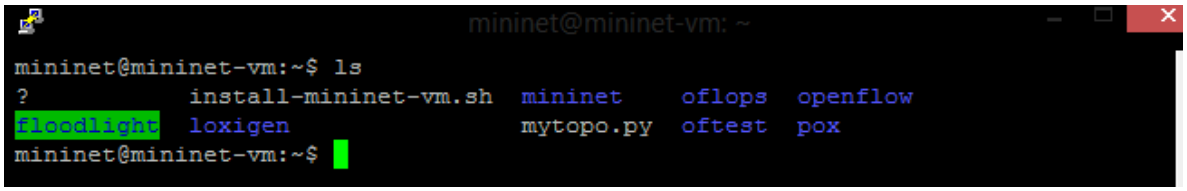


Figura 25. Entorno de simulación

Otra herramienta que es imprescindible y se ha instalado desde su repositorio es OpenNetMon *Figura 26 y 27*. Esta herramienta permite monitorizar el flujo de forma precisa, además cuenta con una licencia GNU General Public License (GPL), Lo cual permite modificar su núcleo que está desarrollado en Python y adaptarlo para satisfacer las necesidades en la monitorización de la red

Figura 26. OpenNetMon

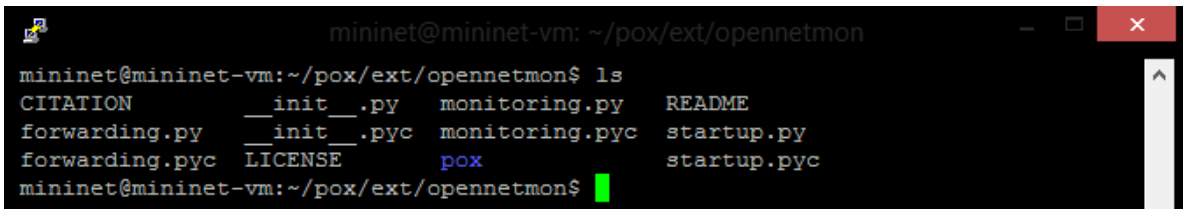


Figura 27. OpenNetmon núcleo

4.3. Topología de red

MiniNet permite la simulación de diferentes topologías de red. Esto hace fácil la adaptación, aplicando los parámetros de configuración en un entorno físico. Los switches solo deben permitir la integración con el protocolo openflow para que la red sea totalmente funcional.

MiniNet soporta cuatro tipos de topología:

TOPOLOGÍA	DESCRIPCIÓN
Single	Genera una topología simple de 1 switch y N hosts.
Linear	Genera una topología en serie de k switches con N hosts conectados a cada switch.
Tree	Genera una topología de árbol compuesta de N niveles, N ramas, y 2 switch conectados a cada switch.
Custom	Lee archivos de configuración en Python, con extensión .py, principalmente para crear redes personalizadas.

Tabla 6. Topologias

Debido a que la monitorización se realizó en 2 host y con un solo switch OF. Se implementa una topología de tipo single, en la *Figura 28*, se puede evidenciar como se crea la topología, además se simula la red con una conexión donde cada host está conectado al switch con un cable Ethernet, además se configura una dirección MAC igual a la dirección ip para cada host. La configuración se realiza con el siguiente comando: `$ sudo mn --topo single,3 --mac --switch ovsk --controller remote`

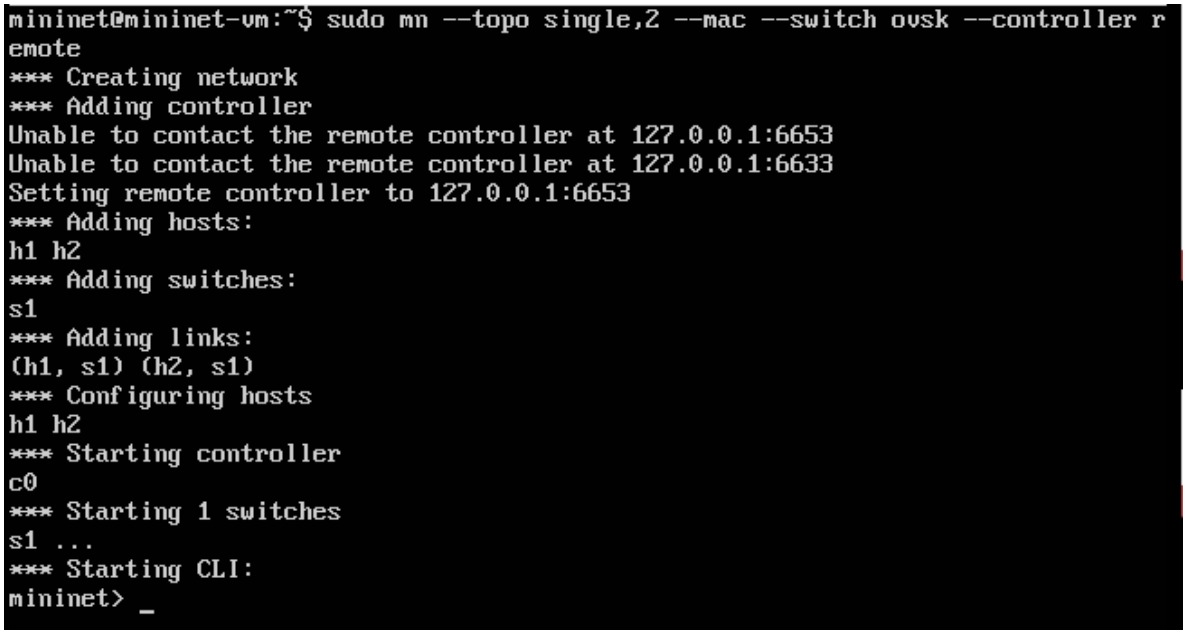


Figura 28. Topología implementada

Al crear la red se puede comprobar los nodos habilitados, solo basta con digitar el comando: `mininet> nodes`, en la consola de mininet *Figura 29.*

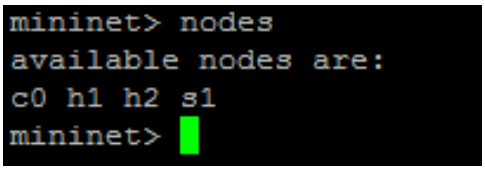


Figura 29. Nodos creados

Es posible también mirar la configuración de cada host con el siguiente comando: `mininet> hn ifconfig`, Donde n es el número del host que se desea inspeccionar *Figura 30.*

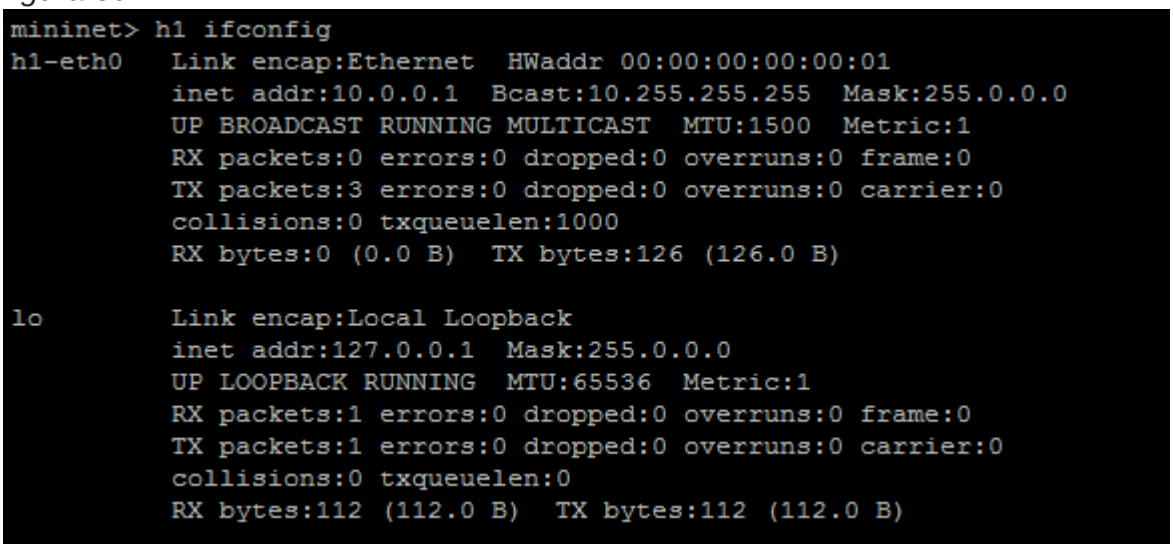


Figura 30.Configuración h1, ifconfig

Para comprobar la conexión y el envío de paquetes como se muestra en la *Figura 31*. Muestra claramente como al hacer ping enviamos un paquete desde el host 1 hacia el host 2, dice que el destino es inaccesible.

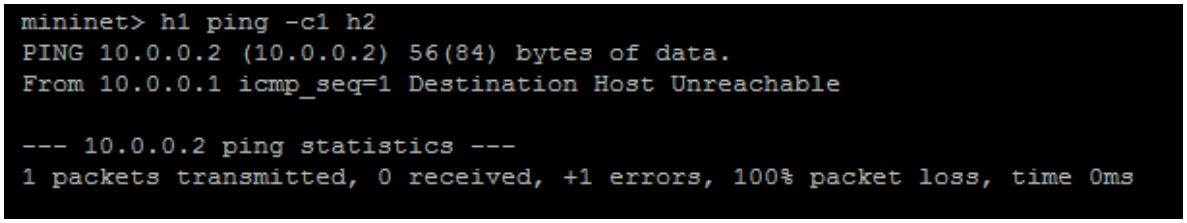


Figura 31. Enviando paquetes desde h1 a h2

Se evidencia de como un switch OF tiene los puertos cerrados y no permite que la información fluya a través de ambos hosts, necesita de un “cerebro” que le permita decidir en base a la información recibida actuar y darle una salida. Si se mira la tabla de flujo estará vacía *Figura 32.*

```
mininet@mininet-vm:~$ sudo ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_VLAN SET_DL_SRC SET_DL_DST SET_N
W_SRC SET_NW_DST SET_NW_TOS SET_TP_SRC SET_TP_DST ENQUEUE
 1(s1-eth1): addr:7e:42:63:03:8c:a5
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:02:b6:c9:35:87:5a
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:5e:d9:da:f1:92:49
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
mininet@mininet-vm:~$
```

Figura 32. Tabla de flujo vacía

Esto pasa porque aún no se han abierto los puertos del switch, para esto hay dos formas de hacerlo: Manualmente ingresando una sentencia que permita el flujo entre ambos hosts *Figura 034* o iniciando un controlador Figura 1

```
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2
mininet@mininet-vm:~$ sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1
```

Figura 33.Habilitando flujo entre h1 y h2

Se puede apreciar como en la *Figura 33*, se han abierto los puertos, y se le ha dicho al switch que los paquetes que lleguen por el puerto 1 sean enviados por el puerto 2 y viceversa. Ahora al hacer ping se permitirá la transmisión de paquetes entre ambos host *Figura 34.*

```
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.725 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.050 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.037/0.270/0.725/0.321 ms
```

Figura 34. Haciendo ping h1 a h2 con puertos abiertos

Ahora la tabla de flujo al ser consultada estará con la información de la transmisión *Figura 35.*

```
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=756.416s, table=0, n_packets=5, n_bytes=378, idle_age=474,
 in_port=1 actions=output:2
 cookie=0x0, duration=740.468s, table=0, n_packets=5, n_bytes=378, idle_age=474,
 in_port=2 actions=output:1
mininet@mininet-vm:~$
```

Figura 35. Tabla de flujo

Otra forma de analizar como fluye los paquetes a través de ambos hosts, es mediante Wireshark, para ello debe estar inicializado Xming y digitar el siguiente comando: `$ sudo wireshark &`, automáticamente abrirá desde el cliente ssh la herramienta Wireshark, y se podrá analizar el tráfico de la red creada *Figura 36.*

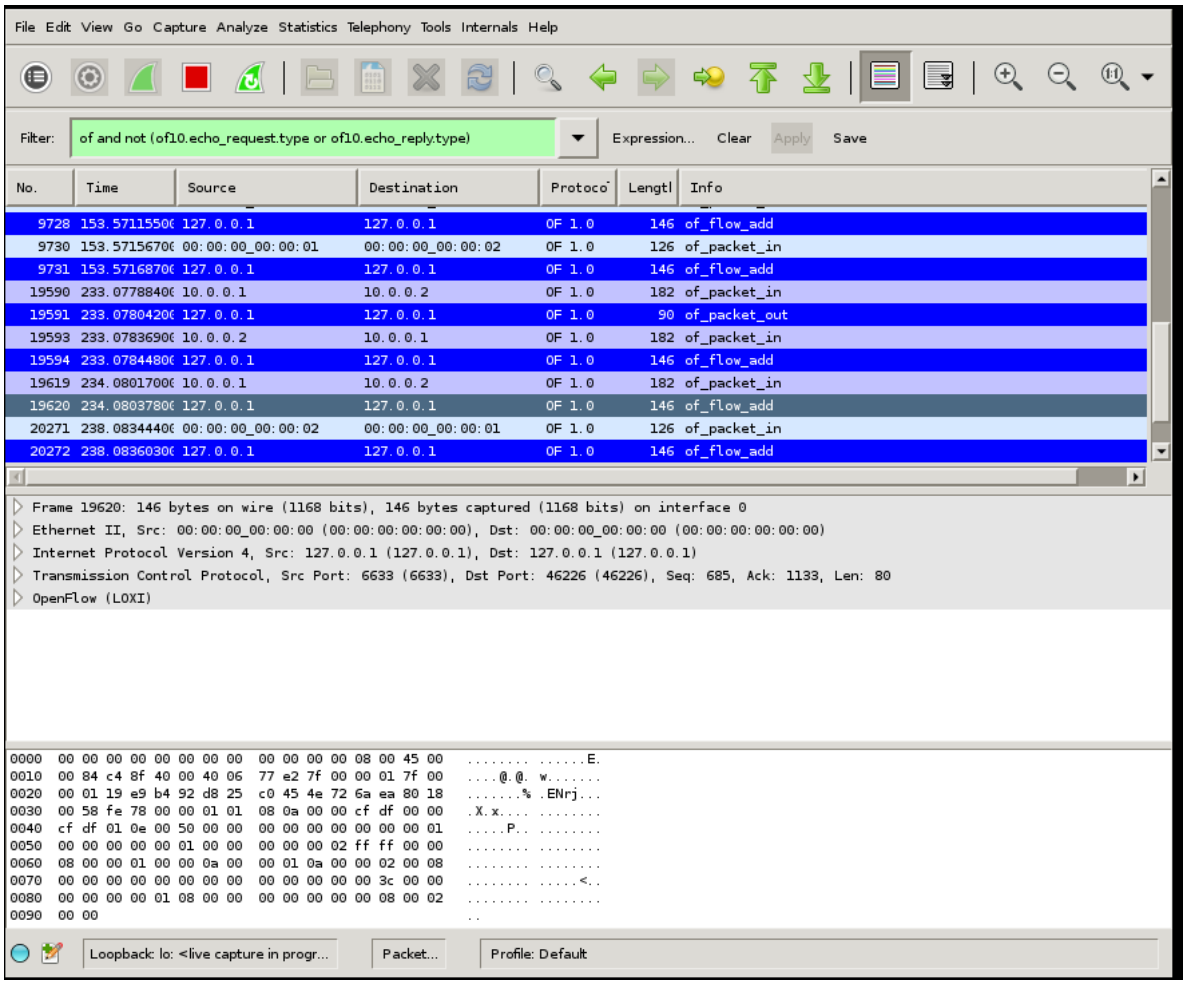


Figura 36. Analizando tráfico a través de wireshark

En la *Figura 38,* se puede apreciar como al aplicar el filtro of que permite ver el tráfico de la red corriendo a través del protocolo OF, al hacer ping desde el host 1 al host 2, captura la información del paquete como la ip de origen.

CAPÍTULO 5. MONITOREO DE UNA RED SDN

Una red puede incluir una variedad de dispositivos que transfieren datos, esta información es típicamente contenida en paquetes que se transfieren mediante dispositivos de red como conmutadores, enrutadores u otros, para determinados casos esta información contenida es requerida para realizar monitoreo de los distintos componentes de la red que nos ayudan a observar el tránsito, estado de esta y así mismo su rendimiento y funcionalidad [71]

En cuanto a las redes definidas por software, es indispensable que los componentes del hardware funcionen de manera adecuada. El buen control puede ayudar a resolver situaciones adversas, y no provoca ninguna desventaja, pero el rendimiento de toda la red es influenciado directamente a la medición de datos y detección temprana de anomalías.

El monitoreo en la administración de la red es crucial. Las aplicaciones de administración requieren estadísticas precisas y oportunas sobre los recursos de red en diferentes niveles de agregación. Sin embargo, la sobrecarga de la red para la recopilación de estadísticas debe ser mínima. Las estadísticas precisas y oportunas son esenciales para muchas tareas de gestión de red, como el equilibrio de carga, la ingeniería de tráfico, la aplicación del acuerdo de nivel de servicio (SLA), la contabilidad y la detección de intrusiones. [51]

La tarea de monitorización en una red SDN comprende varios actores que conforman una arquitectura estándar, El controlador SDN contiene tablas de flujo para administrar todo los flujos transportados a través de la red. En algunos casos, específico mediante funciones de monitoreo el controlador SDN da las órdenes al switch SDN para monitorear los eventos específicos, y el SDN el interruptor sigue monitoreándolos para detectar los eventos especificados. Este tipo de actividades de monitoreo pueden darle a SDN cambiar el carga de todos modos. Si se monitorea un flujo para un evento específico y el evento ocurre con frecuencia o periódicamente, luego el anterior método sería considerado como eficiente. Sin embargo, si el evento ocurre raramente, entonces puede no ser el método correcto. [84]

5.1. Monitorización de la red

Hasta el momento se ha evidenciado como a partir de una red simulada con un switch SDN y dos host, ha sido posible programar manualmente el comportamiento para el reenvío de paquetes. Y como ha sido posible analizar el tráfico de estos al hacer pruebas de ping. Pero, para monitorizar una red, es necesario desplegar un controlador que permita monitorizar en tiempo real el

estado de la red, ya que un administrador de red debe disponer de la información necesaria para tomar una decisión

5.1.1. Desplegando controlador SDN POX

Hasta el momento se ha evidenciado como a partir de una red virtual, con un switch SDN y dos host se ha programado para que manualmente haga el reenvío de paquetes desde un host al otro. También se ha hecho evidencia de cómo ha sido posible analizar el tráfico al hacer pruebas de ping entre ambos hosts. Pero, para monitorizar una red, es necesario desplegar un controlador que permita monitorizar en tiempo real el estado de esta, ya que un administrador de red debe disponer de la información necesaria para gestionar los recursos.
En el proyecto como se menciona en el capítulo 4, se hizo uso del controlador POX, el cual contiene un núcleo desarrollado en el lenguaje de programación: Python. Así mismo permite desarrollar aplicaciones en este lenguaje que contribuyan con las necesidades de la red.

Para poder desplegar un controlador primero se debe consultar si el proceso está corriendo *Figura 37*.

```
mininet@mininet-vm:~$ ps -A | grep controller
mininet@mininet-vm:~$
```

Figura 37. Verificación de procesos

Como se puede observar, no arroja resultados por lo cual no hay controladores desplegados aun. *Figura 38*, muestra que está instalado el controlador POX para poder desplegarlo.

```
mininet@mininet-vm:~$ ls -l
total 32
-rw-rw-r--  1 mininet mininet    0 Jul 26 09:50 ?
-rw-rw-r--  1 mininet mininet 1630 Mar 21  2017 install-mininet-vm.sh
drwxrwxr-x 17 mininet mininet 4096 Mar 21  2017 loxigen
drwxrwxr-x 13 mininet mininet 4096 Mar 21  2017 mininet
-rw-rw-r--  1 mininet mininet  894 Aug  8 10:10 mytopo.py
drwxrwxr-x 14 mininet mininet 4096 Mar 21  2017 oflops
drwxrwxr-x 11 mininet mininet 4096 Mar 21  2017 oftest
drwxrwxr-x 19 mininet mininet 4096 Mar 21  2017 openflow
drwxrwxr-x  7 mininet mininet 4096 Sep 19 20:08 pox
```

Figura 38. Verificando instalación de POX

Al verificar que esté instalado se ingresa a su carpeta Figura 39. Y se despliega el servicio *Figura 40.*

```
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$
```

Figura 39. Ingresando carpeta POX

```
floodlight@floodlight:~/pox$ ./pox.py log.level --DEBUG misc.of_tutorial
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.1.0 (betta) going up...
DEBUG:core:Running on CPython (2.7.6/Nov 23 2017 15:49:48)
DEBUG:core:Platform is Linux-3.13.0-32-generic-x86_64-with-Ubuntu-14.04-trusty
INFO:core:POX 0.1.0 (betta) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 1]
```

Figura 40. Desplegando controlador POX

Esto le dice a POX que habilite el registro detallado y que comience el componente of_tutorial que usará (Que actualmente actúa como hub). Para verificar esto se abren dos ventanas (Una para cada host) Figura 41.

```
mininet> xterm h1 h2
mininet>
```

Figura 41. Abriendo ventanas para cada host

En la ventana del nodo 1 (host 1) se hace ping hacia al nodo 2 (host 2) Figura 42. Y en la ventana del nodo 2 se captura la información que comprueba el éxito del ping entre ambos hosts Figura 43.

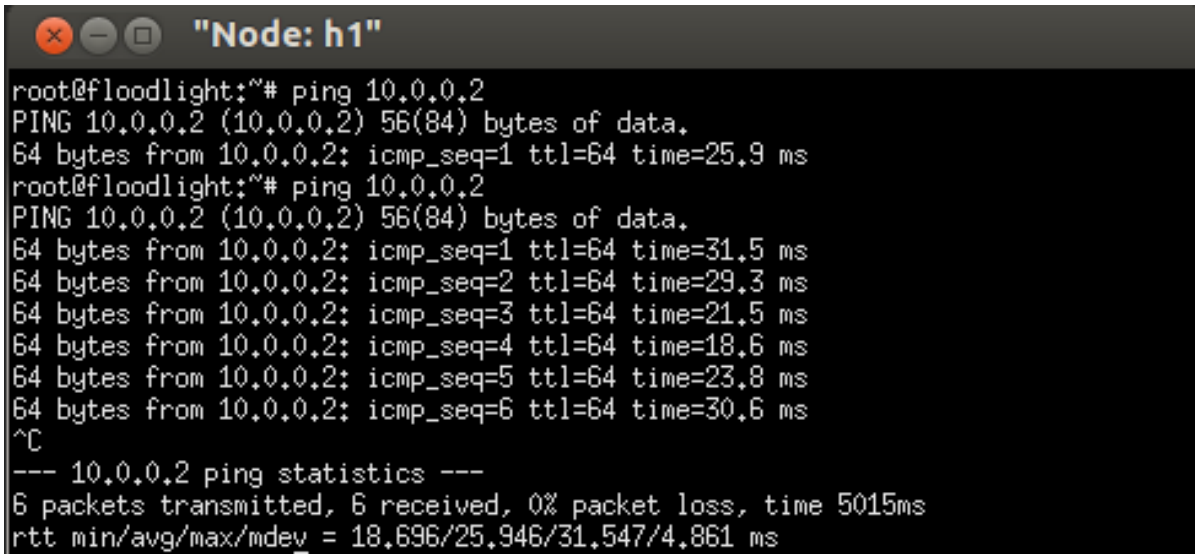


Figura 42. Haciendo ping desde host 1 a host 2

En resumen, no fue necesario abrir puertos manualmente. Solo se requirió tener instalado un controlador que esté atento en el momento en el cual se monte una red para que re direccione el envío de paquetes. Hasta aquí el switch funciona como si fuera un hub. Pero, para poder monitorizar la red es necesario configurarlo como switch para que agilice el intercambio de paquetes entre ambos hosts y hacer “fácil” la monitorización de estos.

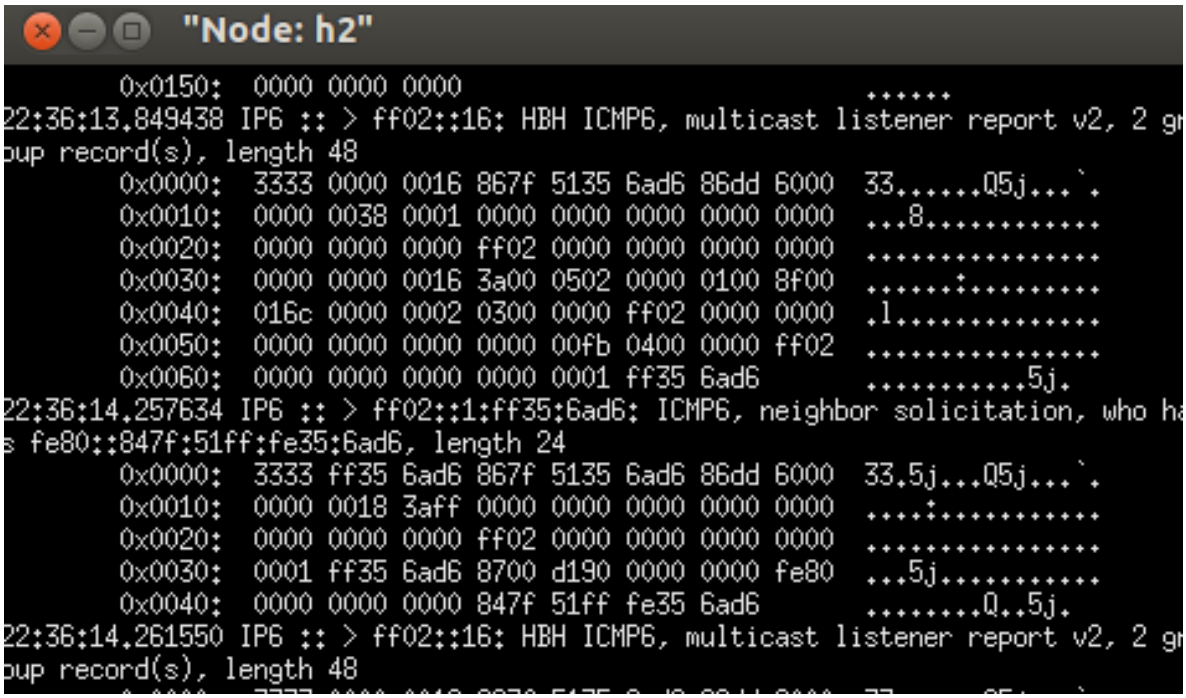


Figura 43. Comprobando ping en el host 2

5.2. Desplegando OpenNetMon

Dada la necesidad de configurar el switch para que funcione de manera adecuada es necesaria la instalación y despliegue de OpenNetMon. Herramienta totalmente libre que corre bajo el lenguaje de programación Python. Su principal contribución es monitorizar el Delay entre todos los flujos y rutas en uso en el controlador POX, haciendo posible la reutilización de código para la construcción de la función que permita la medición del jitter, ya que está programado bajo el paradigma orientado a objetos (POO).

5.2.1. Instalación de OpenNetMont

Para la instalación primero se crea un directorio dentro de la carpeta POX con el nombre de “opennemont” *Figura 44*. Y se verifica que haya sido creado *Figura 46*.

```
floodlight@floodlight:~$ mkdir ~/pox/ext/opennetmon
floodlight@floodlight:~$ ls
```

Figura 44. Creando directorio OpenNetmon

```
floodlight@floodlight:~/pox/ext$ ls
README  opennetmon
```

Figura 45. Verificación de carpeta

Después de la creación de la carpeta, se procede a clonar el proyecto desde su repositorio: [86] *Figura 46*.

```
floodlight@floodlight:~/pox$ git clone https://github.com/TUDelftNAS/SDN-OpenNet
Mon ~/pox/ext/opennetmon
Cloning into '/home/floodlight/pox/ext/opennetmon'...
remote: Enumerating objects: 29, done.
remote: Total 29 (delta 0), reused 0 (delta 0), pack-reused 29
Unpacking objects: 100% (29/29), done.
Checking connectivity... done.
```

Figura 46. Clonando desde repositorio

5.2.2. Corriendo OpenNetMon

Al haberse instalado correctamente *Figura 46*. Solo queda desplegar OpenNetMon *Figura 47*. Donde empieza está ocupando el puerto 6633 esperando que se envíen paquetes para su respectivo monitoreo.

```
floodlight@floodlight:~/pox$ ./pox.py opennetmon.startup
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
[opennetmon.forwarding  ] Forwarding coming up
[opennetmon.forwarding  ] Forwarding started
[opennetmon.monitoring  ] Monitoring coming up
[opennetmon.monitoring  ] Monitoring started
[core                   ] POX 0.1.0 (betta) going up...
[core                   ] Running on CPython (2.7.6/Nov 23 2017 15:49:48)
[core                   ] Platform is Linux-3.13.0-32-generic-x86_64-with-Ubuntu
-14.04-trusty
[core                   ] POX 0.1.0 (betta) is up.
[openflow.of_01         ] Listening on 0.0.0.0:6633
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:44.108636
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:45.118342
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:46.288257
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:47.557822
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:49.008297
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:50.631812
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:52.461814
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:54.493880
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:56.820007
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:38:59.388455
[opennetmon.monitoring  ] Monitoring paths 2018-09-25 23:39:02.290466
```

Figura 47. OpenNetMon desplegado

5.3. Monitorización del Jitter

Una vez instalado OpenNetMon, se desarrolló la función que permite monitorizar el jitter para esto fue necesario investigar sobre programación en Python a nivel de red, documentación sobre POX y la documentación de OpenNetmon.

5.3.1. Diagrama de flujo

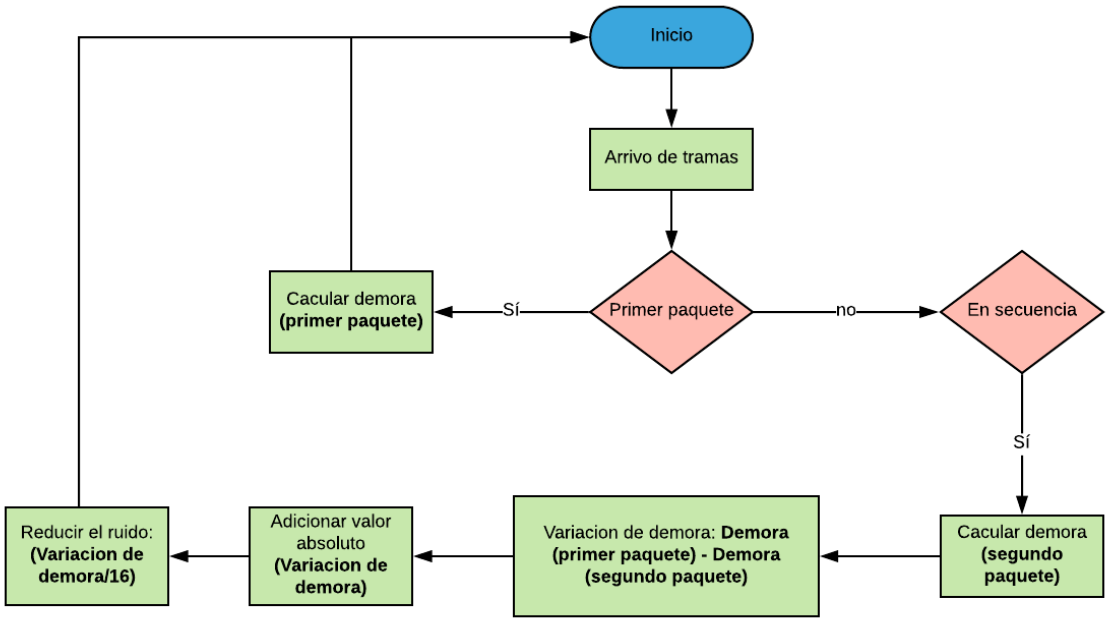


Figura 48. Diagrama de flujo para medida del jitter basado en diagrama de tesis [85]

Si el paquete es el primero recibido en la secuencia, entonces se calcula la demora de transferencia y se almacena. Si no es el primero en la secuencia, entonces se realiza un chequeo para asegurarse que está en la secuencia correcta. Si el paquete no está en secuencia, se eliminan los resultados del valor anterior y se toma este paquete como el primero. Si el paquete recibido no es el primero se calcula el retraso del segundo paquete. Se calcula la variación de demora (Jitter) que es la diferencia entre el primer paquete y el segundo paquete. Posteriormente se le saca el valor absoluto y se reduce el ruido para mejorar la precisión de los datos [85]

5.3.2. Medición del Jitter

Después de haber inicializado OpenNetMon *Figura 48*. Empezamos a enviar paquetes desde el host 1 al host 2, es ahí donde la función desarrollada en Python sobre OpenNetmon y basada en el diagrama de flujo empieza a monitorizar el Jitter entre paquetes *Figura 49*.

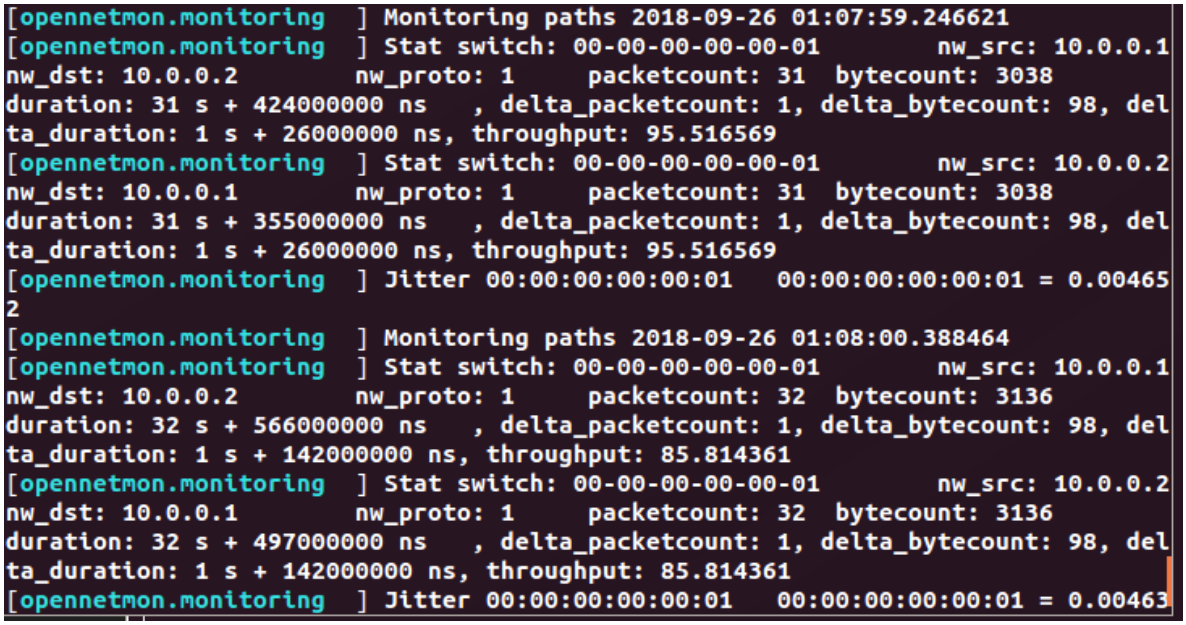


Figura 49. Monitorización de Jitter

5.3.3. Evaluación de la función

El retraso entre llegada de paquetes o fluctuación de fase (Jitter), en nuestra red virtualizada se mide en distintas secuencias, la función de monitoreo brinda un muestreo básico del tiempo de llegada entre paquetes en ms y las medidas de la pérdida de tiempo entre estos, para este caso de validación tal vez porque se trata de una red a pequeña escala con pocos componentes, es posible afirmar que la conexión se mantiene estable, la función de monitoreo nos arroja en la *Figura 49*, dos mediciones de Jitter y se puede notar que en los dos registros de medida el rango de Jitter está por debajo de los 100 ms, (**0.00465 ms** y **0.00463 ms** respectivamente) que es un rango tolerable y estable, en caso contrario al notar que la estadística de medida sobrepasa el límite en ms tolerado, debiera tomarse medidas para minimizar esta fluctuación y de no hacerse se presentaría un grave

problema de retardo, perdida de paquetes y hasta interrupciones con la caída de conexión. En la *Figura 50* se hace un paralelo entre los datos reales recogidos por la función de monitoreo y un hipotético caso de ejemplo en donde se presentaría un desmedido retraso entre paquetes que genera una variación exagerada de Jitter y por ende se deberían tomar alternativas para solucionar el evento inesperado.

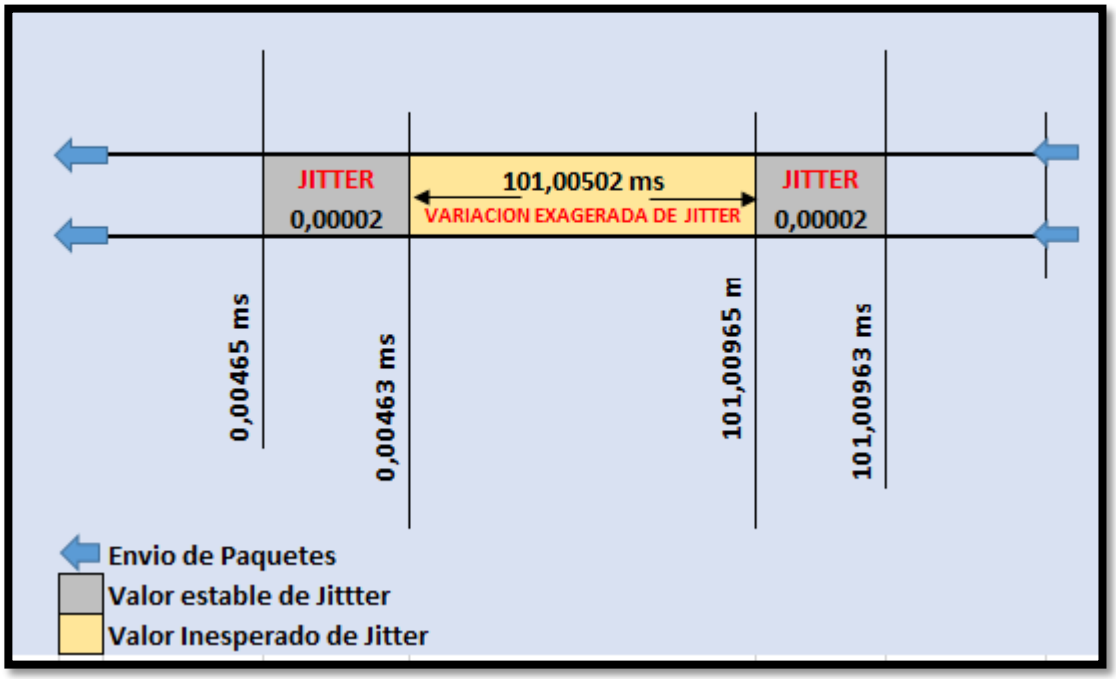


Figura 50. Diagrama de medición y validación de datos Jitter

CAPÍTULO 6. CONCLUSIONES

En este documento, hemos diseñado e implementado por primera vez una función de monitoreo de red, desarrollada en un entorno SDN totalmente virtualizado y bajo parámetros del protocolo OpenFlow, Dado que la función de monitoreo está programada dentro del controlador POX y corre a través de OpenNetMon, se consigue una vista plena de monitoreo que registra el intercambio de resultados y una medición en tiempo real del evento que ayudan a la supervisión y mecanismos de control para la red.

La solución propuesta como método de monitoreo para un evento especifico JITTER que se presenta en este documento demuestra que si se pueden proporcionar aplicaciones de monitoreo y gestión de red capaces de seguir los estados cambiantes de la SDN, de este modo al proveer datos concretos y estadísticas permite reconfiguraciones rápidas y efectivas que contribuyen a la corrección significativamente de los retrasos, reducción de costos y consumo de recursos

En términos generales se puede deducir que la solución propuesta ayuda a conocer más de cerca los inconvenientes que provocan los eventos específicos en una red, se demuestra que con el uso de las herramientas adecuadas, un entorno de red versátil y estable, así como con buenas técnicas de programación SDN, es posible monitorizar estos parámetros (JITTER) de modo que se obtengan datos reales de las actividades en la red y por tanto si se encuentran inconsistencias el administrador tendrá la opción de tomar acciones preventivas o dado el caso corregir dichas anomalías.

Por otro lado, fue muy interesante conocer que las redes SDN han cambiado la forma en cómo se veían las redes tradicionales, el hecho de inventarse una manera de administrar la red de forma programable y administrarla centralizadamente evoca un sentido de grandeza, es un gran paso que revoluciona los sistemas de información. Ahora bien, el hecho de adoptar estas tecnologías en nuestro proyecto de investigación abre las puertas para el conocimiento y aporta tareas y retos grandes a futuro como el objetivo de seguir monitorizando eventos en la red y supervisar el estado de esta. El siguiente paso que se pretende dar con el desarrollo de esta investigación, es en un mañana llegar a programar un sistema completo de monitoreo bajo OpenFlow en un entorno autónomo como Onnos que supervise y recopile información en conjunto de parámetros como (tráfico, métricas, kpi, alertas, retardos, etc), y en contexto mejore la forma en cómo se gestiona y se controla la red.

Referencias

[1] D. Martín, M. Marrero, J. Urbano, E. Barra, y J.-A. Moreiro, «Virtualización, una solución para la Eficiencia, Seguridad y Administración de Intranets», *El Prof. Inf.*, vol. 20, n.º 3, pp. 348-355, may 2011.

[2] «Redes de computadoras.pdf». .

[3] J. D. Britos, L. Vargas, S. Arias, N. Giraudo, y G. Veneranda, «Laboratorio virtual y remoto para la enseñanza de diseño y administración de redes de computadoras», en *XV Workshop de Investigadores en Ciencias de la Computación*, 2013.

[4] G. Tangari, D. Tuncer, M. Charalambides, y G. Pavlou, «Decentralized monitoring for large-scale Software-Defined Networks», en *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 289-297.

[5] «Informe Anual La Sociedad en red 2015 (Edición 2016).pdf». .

[6] G. Liu y D. Jiang, «5G: Vision and Requirements for Mobile Communication System towards Year 2020», *Chinese Journal of Engineering*, 2016. [En línea]. Disponible en: https://www.hindawi.com/journals/cje/2016/5974586/. [Accedido: 14-nov-2017].

[7] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, y S. Uhlig, «Software-Defined Networking: A Comprehensive Survey», *Proc. IEEE*, vol. 103, n.º 1, pp. 14-76, ene. 2015.

[8] H. Alkhatib *et al.*, «What Will 2022 Look Like? The IEEE CS 2022 Report», *Computer*, vol. 48, n.º 3, pp. 68-76, mar. 2015.

[9] J. A. Wickboldt, W. P. D. Jesus, P. H. Isolani, C. B. Both, J. Rochol, y L. Z. Granville, «Software-defined networking: management requirements and challenges», *IEEE Commun. Mag.*, vol. 53, n.º 1, pp. 278-285, ene. 2015.

[10] N. Samaan y A. Karmouch, «Towards Autonomic Network Management: an Analysis of Current and Future Research Directions», *IEEE Commun. Surv. Tutor.*, vol. 11, n.º 3, pp. 22-36, rd 2009.

[11] F. Estrada-Solano, A. Ordonez, L. Z. Granville, y O. M. Caicedo Rendon, «A framework for SDN integrated management based on a CIM model and a vertical management plane», *Comput. Commun.*, vol. 102, n.º Supplement C, pp. 150-164, abr. 2017.

[12] E. E. Haleplidis, E. K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, y O. Koufopavlou, «Software-Defined Networking (SDN): Layers and Architecture Terminology», 2015.

[13] «TR_SDN_ARCH_1.0_06062014.pdf». .

[14] «T-REC-Y.3300-201406-I!!PDF-E.pdf». .

[15] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, y R. Boutaba, «Network Function Virtualization: State-of-the-art and Research Challenges», *IEEE Commun. Surv. Tutor.*, vol. 18, n.º 1, pp. 236-262, 2016.

[16] S. Kuklinski y P. Chemouil, «Network Management Challenges in Software-Defined Networks», *IEICE Trans. Commun.*, vol. E97-B, n.º 1, pp. 2-9, ene. 2014.
[17] Y. Wang y I. Matta, «SDN Management Layer: Design Requirements and Future Direction», en *2014 IEEE 22nd International Conference on Network Protocols*, 2014, pp. 555-562.
[18] O. M. Caicedo Rendon, F. Estrada-Solano, V. Guimarães, L. M. Rockenbach Tarouco, y L. Z. Granville, «Rich dynamic mashments: An approach for network management based on mashups and situation management», *Comput. Netw.*, vol. 94, n.º Supplement C, pp. 285-306, ene. 2016.
[19] X. T. Phan y K. Fukuda, «Toward a Flexible and Scalable Monitoring Framework in Software-Defined Networks», en *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2017, pp. 403-408.
[20] G. Tangari, D. Tuncer, M. Charalambides, y G. Pavlou, «Decentralized monitoring for large-scale Software-Defined Networks», en *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 289-297.
[21] «Cisco Express Forwarding (CEF), CAM, TCAM Y FIB – Como conmutan los switchs y routers de Cisco», *Sea CCNA*. .
[22] D. Levin, A. Wundsam, B. Heller, N. Handigol, y A. Feldmann, «Logically Centralized?: State Distribution Trade-offs in Software Defined Networks», en *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, New York, NY, USA, 2012, pp. 1–6.
[23] «cyb29_computer_int_sp.pdf». .
[24] «ISI Prof. CM Mansilla - Informática Básica FCA - UNL -Redes de computadorass.pdf». .
[25] «rfc7426.pdf». .
[26] Dr. Víctor J. Sosa Sos, «GestionRedes.pdf». 2012.
[27] T. Tanaka, «Flexible and robust optical network technologies for SDN and network virtualization», en *2014 12th International Conference on Optical Internet 2014 (COIN)*, 2014, pp. 1-2.
[28] J. D. Britos, L. Vargas, S. Arias, N. Giraudo, y G. Veneranda, «Laboratorio virtual y remoto para la enseñanza de diseño y administración de redes de computadoras», en *XV Workshop de Investigadores en Ciencias de la Computación*, 2013.
[29] F. Galán y D. Fernández, «VNUML: Una Herramienta de Virtualización de Redes Basada en Software Libre», en *Proc. Open Source International Conference*, 2004, pp. 35–41.
[30] E. J. Kitindi, S. Fu, Y. Jia, A. Kabir, y Y. Wang, «Wireless Network Virtualization With SDN and C-RAN for 5G Networks: Requirements, Opportunities, and Challenges», *IEEE Access*, vol. 5, pp. 19099-19115, 2017.
[31] T. D. P. C. D. Souza, C. E. Rothenberg, M. A. S. Santos, y L. B. D. Paula, «Towards Semantic Network Models via Graph Databases for SDN

Applications», en *2015 Fourth European Workshop on Software Defined Networks*, 2015, pp. 49-54.
[32] N. L. M. van Adrichem, C. Doerr, y F. A. Kuipers, «OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks», en *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1-8.
[33] Z. Yang y K. L. Yeung, «An efficient flow monitoring scheme for SDN networks», en *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2017, pp. 1-4.
[34] A. G. Centeno, C. M. R. Vergel, C. A. Calderón, y F. C. C. Bondarenko, «Controladores SDN, elementos para su selección y evaluación.», *Rev. Telemtica*, vol. 13, n.º 3, pp. 10-20, nov. 2014.
[35] «Redes_Pop_TM». [En línea]. Disponible en: https://1386288.netacad.com/courses/484013. [Accedido: 26-sep-2017].
[36] P. Tee, G. Parisis, y I. Wakeman, «Vertex Entropy As a Critical Node Measure in Network Monitoring», *IEEE Trans. Netw. Serv. Manag.*, vol. 14, n.º 3, pp. 646-660, sep. 2017.
[37] M. Mehta, R. Rao, y E. R. Swierk, «Systems and methods for controlling switches to monitor network traffic», US9008080 B1, 14-abr-2015.
[38] I. Ahmad, S. Namal, M. Ylianttila, y A. Gurtov, «Security in Software Defined Networks: A Survey», *IEEE Commun. Surv. Tutor.*, vol. 17, n.º 4, pp. 2317-2346, Fourthquarter 2015.
[39] R. B. Santos, T. R. Ribeiro, y C. d A. C. César, «A network monitor and controller using only OpenFlow», en *2015 Latin American Network Operations and Management Symposium (LANOMS)*, 2015, pp. 9-16.
[40] R. Pacífico, P. Goulart, A. B. Vieira, M. A. M. Vieira, y J. A. M. Nacif, «Hardware Modules for Packet Interarrival Time Monitoring for Software Defined Measurements», en *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, 2016, pp. 188-191.
[41] Z. Yang y K. L. Yeung, «An efficient flow monitoring algorithm using a flexible match structure», en *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, 2016, pp. 176-181.
[42] I. Angelopoulos, E. Trouva, y G. Xilouris, «A monitoring framework for 5G service deployments», en *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2017, pp. 1-6.
[43] «4G_Americas_5G_Spectrum_Recommendations_White_Paper_-_Spanish.pdf». .
[44] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, y L. Z. Granville, «Interactive monitoring, visualization, and configuration of OpenFlow-based SDN», en *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 207-215.
[45] S. Baik, Y. Lim, J. Kim, y Y. Lee, «Adaptive flow monitoring in SDN architecture», en *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2015, pp. 468-470.

[46] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, y S. Uhlig, “Software-Defined Networking: A Comprehensive Survey”, Proceedings of the IEEE, vol. 103, núm. 1, pp. 14–76, Enero 2015.

[47] «SDN: el futuro de las redes inteligentes». [En línea]. Disponible en: https://www.ramonmillan.com/tutoriales/sdnredesinteligentes.php. [Accedido: 27-ago-2018].

[48] D. Martín, M. Marrero, J. Urbano, E. Barra, y J.-A. Moreiro, «Virtualización, una solución para la Eficiencia, Seguridad y Administración de Intranets», *El Prof. Inf.*, vol. 20, n.º 3, pp. 348-355, may 2011.

[49] V. Kotronis y B. Ager, «Outsourcing the Routing Control Logic: Better Internet Routing Based on SDN Principles», p. 6.

[50] Ligia Rodrigues Prete, A. A. Shinoda, C. M. Schweitzer, y R. L. S. de Oliveira, «Simulation in an SDN network scenario using the POX Controller», en *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, Bogota, Colombia, 2014, pp. 1-6.

[51] S. R. Chowdhury, M. F. Bari, R. Ahmed, y R. Boutaba, «PayLess: A low cost network monitoring framework for Software Defined Networks», en *2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, 2014, pp. 1-9.

[52] «Open Networking Foundation is an operator led consortium leveraging SDN, NFV and Cloud technologies to transform operator networks and business models», *Open Networking Foundation*. [En línea]. Disponible en: https://www.opennetworking.org/. [Accedido: 27-ago-2018].

[53] D. Bolatti *et al.*, «Estudio de Herramientas de Simulación en Redes Definidas por Software», p. 5.

[54] N. McKeown *et al.*, «OpenFlow: enabling innovation in campus networks», *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, n.º 2, p. 69, mar. 2008.

[55] B. Lantz y B. O’Connor, «Mininet: An instant Virtual Network on your Laptop», p. 43.

[56] B. Valencia y S. Santacruz, «Mininet: una herramienta versátil para emulación y prototipado de Redes Definidas por Software», p. 9.

[57] Shie-Yuan Wang, Chih-Liang Chou, y Chun-Ming Yang, «EstiNet openflow network simulator and emulator», *IEEE Commun. Mag.*, vol. 51, n.º 9, pp. 110-117.

[58] P. Y. Fernádez y J. M. M. Sanahuja, «Programación de redes SDN mediante el controlador POX», p. 85.

[59] «Vargas - Emulación de una red definida por software utiliza.pdf». .

[60] S.-Y. Wang, «Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet», en *2014 IEEE Symposium on Computers and Communications (ISCC)*, Funchal, Madeira, Portugal, 2014, pp. 1-6.

[61] http://sdnhub.org/resources/useful-mininet-setups/, «Recursos | SDN Hub». .

[62] «Xming X Server for Windows - Official Website». [En línea]. Disponible en: http://www.straightrunning.com/XmingNotes/. [Accedido: 27-ago-2018].
[63] «Xming+PuTTY = administración gráfica y remota de un servidor Linux», *MicroTeknologias*, 14-abr-2008. .
[64] «Programar en hendrix-ssh desde un equipo con sistema operativo Windows», p. 4.
[65] « Wilberto Vega, Oficina de Sistemas de Información. SSH PUTTY[110].pdf». .
[67] N. L. M. van Adrichem, C. Doerr, y F. A. Kuipers, «OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks», en *2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, 2014, pp. 1-8.
[68] R. Jmal y L. Chaari Fourati, «Implementing shortest path routing mechanism using Openflow POX controller», en *The 2014 International Symposium on Networks, Computers and Communications*, Hammamet, Tunisia, 2014, pp. 1-6.
[69] D. Tatang, F. Quinkert, J. Frank, C. Ropke, y T. Holz, «SDN-Guard: Protecting SDN controllers against SDN rootkits», en *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Berlin, 2017, pp. 297-302.
[70] «SERRANO - Redes Definidas por Software (SDN)_ OpenFlow (120).pdf». .
[71] H. Packard, «(54) MONITORING NETWORK TRAFFIC», p. 11.
[72] W. Kim, J. Li, J. W.-K. Hong, y Y.-J. Suh, «OFMon: OpenFlow Monitoring System in ONOS Controllers», p. 6.
[73] *Debanshu Sinha, K Haribabu, Sundar Balasubramaniam* Computer Science & Information Systems Birla Institute of Technology & Science, Real-Time Monitoring of Network Latency in Software Defined Networks«latencia en sdn(123).pdf». .
[74] «Wireshark User's Guide: Version 2.9.0», p. 292.

[75] E. A. G. Garzón y C. Y. Solarte, «Seguridad En Redes, Monitoreo y Captura de Información», p. 7

[76] VoIPForo, «VoIP Foro - QoS - jitter - Causas, soluciones y valores recomendados». [En línea]. Disponible en: http://www.voipforo.com/QoS/QoS_Jitter.php. [Accedido: 28-ago-2018].
[77] «¿Es posible monitorear la SDN?» [En línea]. Disponible en: http://www.junipernetworksblog.com/es-posible-monitorear-la-sdn. [Accedido: 28-ago-2018].
[78] «What is jitter? And how it affects your operation», *Innovasic*, 13-ene-2014. [En línea]. Disponible en: http://www.innovasic.com/news/industrial-ethernet/what-is-jitter-and-how-it-affects-your-operation/. [Accedido: 28-ago-2018].
[79] «analisis de diagrama de ojo_Jitter(128).pdf». .

[80] «Medición de demora, fluctuación y pérdida de paquetes con SAA y RTTMON del IOS de Cisco», *Cisco*. [En línea]. Disponible en: https://www.cisco.com/c/es_mx/support/docs/availability/high-availability/24121-saa.html.
[81] «What is Jitter. cisco jitter_Meraki (127).pdf». .
[82] L. Xu, Y. Duan, y D. Chen, «A low cost jitter separation and characterization method», en *2015 IEEE 33rd VLSI Test Symposium (VTS)*, Napa, CA, USA, 2015, pp. 1-5.
[83] «wireless telecom group _201_WTG_Intro_to_ Jitter_SP.pdf». .
[84] R. Izquierdo, «Monitoreo de Red. Qué debemos saber», 25-sep-2017. .
[85] «Osniel Pozo Mederos.pdf_ Madicion y analisis del Jitter en Redes_ La Habana Cuba, 2008».

[86] *Accurate and precise OpenFlow Monitoring POX module: TUDelftNAS/SDN-OpenNetMon*. Delft University of Technology - Network Architectures and Services, 2018.