

Universidad Central de Venezuela  
Facultad de Ciencias  
Escuela de Computación  
Laboratorio de Comunicación y Redes



## Consideraciones de Diseño Para el Despliegue de Redes SDN en Campus Empresariales

Trabajo Especial de Grado  
presentado ante la  
Universidad Central de Venezuela  
por el Bachiller:

Gustavo J. Pereira S.  
V-10352539  
gustavo.pereira.salas@gmail.com

para optar al título de Licenciado en Computación

Tutores: Prof. Eric Gamess, Prof. Dedaniel Urribarri.

Caracas, Noviembre 2018



## Acta de Veredicto



**Universidad Central de Venezuela**  
Facultad de Ciencias  
Escuela de Computación  
Laboratorio de Comunicación y Redes



### ACTA DE VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el Bachiller Gustavo Pereira Salas (C.I. V-10352539), con el título **"Consideraciones de Diseño para el Despliegue de Redes SDN en Campus Empresariales"**, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el nombrado trabajo por cada uno de los miembros del jurado, éste fijó el día viernes 30 de noviembre de 2018, para que su autor lo defendiera en forma pública, en la Sala de Internet II, Laboratorio LACORE, de la Escuela de Computación, mediante una exposición oral de su contenido, luego de la cual respondió satisfactoriamente a las preguntas que le fueron formuladas por el jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela.

Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

Es de aclarar que el Profesor Eric Gamess se encuentra fuera del país. Por esta razón, el Profesor Robinson Rivas, director de la Escuela de Computación, firma el presente documento en su lugar.

En fe de lo cual se levanta la presente acta en Caracas a los 30 días del mes de noviembre del año 2018, dejando constancia que el Profesor Dedaniel Urbani actuó como coordinador del jurado.

por M.H.

por el Prof. Eric Gamess  
Tutor

Prof. Dedaniel Urbani  
Tutor

Prof. Antonio Russoniello  
Jurado Principal

Prof. Jaime Parada  
Jurado Principal



## **Agradecimientos**

Este Trabajo Especial de Grado y todo el esfuerzo que amerita culminar esta espléndida carrera se lo dedico especialmente a mi madre Chiquinquirá y a mi padre Nemesio.

También comparto este agradable mérito con las personas que siempre han confiado en mí y me han inspirado para alcanzar esta meta, mi hermano Horacio, mi hija Catherine, mi mujer Silvia, mi familia venezolana, y a todos mis familiares y amigos.

Agradezco a la U.C.V., a sus profesores y a la comunidad que la conforma.

A mis tutores Eric Gamess y Dedaniel Urribarri, por su exigencia, desmedida dedicación y calidad docente.

**Gustavo J. Pereira S.**





**Universidad Central de Venezuela**  
Facultad de Ciencias  
Escuela de Computación  
Laboratorio de Comunicación y Redes



## **Consideraciones de Diseño para el Despliegue de Redes SDN en Campus Empresariales**

**Autor:** Gustavo Pereira

**Tutores:** Prof. Eric Gamess y Prof. Dedaniel Urribarri.

### **Resumen**

En este Trabajo Especial de Grado se proponen criterios de diseño para implementar redes SDN (Software Defined Networking) en entornos de Campus Empresariales. SDN es una arquitectura de red emergente que separa el plano de control del plano de datos de los dispositivos de la red y coloca el plano de control en uno o varios servidores de control capaces de gestionar las reglas de reenvío de tráfico de todos los dispositivos de comunicación bajo su dominio. En la investigación se describen los principales componentes de SDN incluyendo hardware, software y protocolos y las consideraciones de diseño para el despliegue de redes SDN en el ámbito de campus empresariales. Las redes de campus empresariales están delimitadas a un conjunto de edificios o pisos de una edificación interconectados mediante redes Ethernet. El estudio comprende las raíces de SDN, las limitantes de las redes tradicionales, la arquitectura SDN, el protocolo OpenFlow, casos de uso SDN, el caso de estudio del despliegue SDN en el campus de la Universidad de Stanford y las consideraciones de diseño a seguir basado en las lecciones aprendidas en el caso de estudio y en los trabajos de investigación relacionados.

**Palabras Clave:** SDN, OpenFlow, Arquitectura SDN, Controladores SDN, Casos de Uso SDN, Caso de Estudio SDN, Criterios de Diseño.

## Tabla de Contenido

|                                                                      |    |
|----------------------------------------------------------------------|----|
| Índice de Figuras.....                                               | 13 |
| Índice de Tablas.....                                                | 15 |
| 1. Introducción.....                                                 | 17 |
| 2. El Problema.....                                                  | 19 |
| 2.1 Planteamiento del Problema .....                                 | 19 |
| 2.2 Justificación del Problema.....                                  | 19 |
| 2.2.1 Objetivo General.....                                          | 20 |
| 2.2.2 Objetivos Específicos .....                                    | 20 |
| 3. Antecedentes de SDN.....                                          | 21 |
| 3.1 Redes de Campus Empresariales Tradicionales .....                | 24 |
| 3.1.1 Características de las Redes de Campus Tradicionales .....     | 25 |
| 3.1.2 Tecnologías de Redes de Campus Tradicionales .....             | 25 |
| 3.1.3 Problemas de Desempeño en las Redes de Campus Tradicionales .. | 29 |
| 3.1.4 Limitaciones de las Redes de Campus Tradicionales .....        | 31 |
| 4. Redes SDN .....                                                   | 33 |
| 4.1 Arquitectura General de SDN.....                                 | 33 |
| 4.1.1 Pilares Fundamentales de una Arquitectura de Red SDN.....      | 34 |
| 4.1.2 Operación de SDN.....                                          | 35 |
| 4.2 Arquitectura de Capas SDN .....                                  | 36 |
| 4.2.1 Dispositivo SDN.....                                           | 37 |
| 4.2.2 Controlador SDN .....                                          | 37 |
| 4.2.3 Plano de Gestión .....                                         | 41 |
| 4.2.4 Arquitectura de Gestión Basada en Modelos .....                | 42 |
| 4.2.5 Protocolos de Gestión de Red .....                             | 42 |
| 4.2.6 Plano de Aplicaciones .....                                    | 43 |
| 4.2.7 Interfaces.....                                                | 43 |
| 4.2.8 Capas de Abstracción Adicionales .....                         | 44 |
| 4.2.9 Virtualización de Red (Slicing) .....                          | 44 |
| 4.2.10 Lenguajes de Programación SDN .....                           | 46 |
| 4.2.11 Aplicaciones .....                                            | 46 |
| 4.3 Comparativa entre Arquitectura de Red Tradicional y SDN .....    | 47 |
| 4.3.1 Análisis Funcional de una Arquitectura de Red Tradicional..... | 47 |
| 4.3.1 Análisis Funcional de la Arquitectura SDN .....                | 49 |

|       |                                                           |    |
|-------|-----------------------------------------------------------|----|
| 4.3.1 | Comparación entre Redes Tradicionales y Redes SDN .....   | 49 |
| 4.4   | Modelos de Despliegue SDN .....                           | 51 |
| 4.4.1 | Modelo Basado en Dispositivos SDN .....                   | 51 |
| 4.4.2 | Despliegue SDN Overlay .....                              | 52 |
| 4.4.3 | Despliegue SDN Híbrido.....                               | 55 |
| 5.    | OpenFlow.....                                             | 57 |
| 5.1   | Arquitectura OpenFlow.....                                | 58 |
| 5.1.1 | Switch OpenFlow.....                                      | 58 |
| 5.1.2 | Funcionamiento de Switches OpenFlow.....                  | 59 |
| 5.1.3 | Switch solo-OpenFlow (OpenFlow-only).....                 | 60 |
| 5.1.4 | Switch OpenFlow-híbrido (OpenFlow-hybrid) .....           | 61 |
| 5.1.5 | Controlador OpenFlow.....                                 | 62 |
| 5.2   | Tipos de Flujos.....                                      | 62 |
| 5.3   | Modos de Cargas de Flujos .....                           | 63 |
| 5.4   | Tablas OpenFlow .....                                     | 63 |
| 5.4.1 | Tablas de Flujos .....                                    | 63 |
| 5.4.2 | Ejemplo de Tabla de Flujos OpenFlow v1.0 .....            | 64 |
| 5.4.3 | Tablas de Grupos .....                                    | 65 |
| 5.4.4 | Contadores .....                                          | 66 |
| 5.5   | Protocolo OpenFlow.....                                   | 66 |
| 5.5.1 | Conexiones al Canal OpenFlow .....                        | 67 |
| 5.6   | Especificaciones OpenFlow .....                           | 68 |
| 5.6.1 | OpenFlow Versión 1.0.0 .....                              | 68 |
| 5.6.2 | OpenFlow Versión 1.1.0 .....                              | 71 |
| 5.6.3 | OpenFlow Versión 1.2 .....                                | 76 |
| 5.6.4 | OpenFlow Versión 1.3.0 .....                              | 77 |
| 5.6.5 | OpenFlow versión 1.4.0.....                               | 78 |
| 5.6.6 | OpenFlow versión 1.5.0.....                               | 78 |
| 5.6.7 | Cuadro Comparativo de las Especificaciones OpenFlow ..... | 80 |
| 6.    | Contexto Actual de SDN .....                              | 81 |
| 6.1   | Herramientas de Simulación SDN.....                       | 82 |
| 6.1.1 | ns-3.....                                                 | 82 |
| 6.1.2 | Mininet .....                                             | 83 |
| 6.1.3 | VT-Mininet (Virtual Time Mininet) .....                   | 85 |

|        |                                                         |     |
|--------|---------------------------------------------------------|-----|
| 6.1.4  | Estinet.....                                            | 86  |
| 6.1.5  | OFNet.....                                              | 87  |
| 6.1.6  | Integración OpenFlow en OMNeT++ .....                   | 88  |
| 6.2    | Switches OpenFlow basados en Software .....             | 88  |
| 6.2.1  | Open vSwitch (OVS).....                                 | 88  |
| 6.2.2  | Pantou/Open WRT .....                                   | 89  |
| 6.2.3  | ofsoftswitch14.....                                     | 90  |
| 6.2.4  | Indigo.....                                             | 90  |
| 6.3    | Switches OpenFlow Comerciales basados en Hardware.....  | 90  |
| 6.3.1  | Switches Bare Metal.....                                | 91  |
| 6.3.2  | Switches White Box.....                                 | 92  |
| 6.3.3  | Switches Brite Box.....                                 | 92  |
| 6.4    | Controladores SDN OpenFlow de Código Abierto .....      | 93  |
| 6.5    | Controladores SDN de Código Abierto Vigentes.....       | 93  |
| 6.5.1  | OpenDaylight.....                                       | 93  |
| 6.5.2  | Floodlight .....                                        | 98  |
| 6.5.3  | Ryu .....                                               | 103 |
| 6.6    | Controladores SDN de Propósito Especial.....            | 107 |
| 6.6.1  | FlowVisor.....                                          | 107 |
| 6.6.2  | Ovs-controller .....                                    | 107 |
| 6.6.3  | FlowN .....                                             | 107 |
| 6.6.4  | RouteFlow .....                                         | 107 |
| 6.7    | Herramientas de Gestión SDN.....                        | 108 |
| 6.8    | Herramientas de Monitoreo SDN .....                     | 108 |
| 6.8.1  | PayLess.....                                            | 108 |
| 6.8.2  | OpenTM.....                                             | 108 |
| 6.8.3  | FlowSense .....                                         | 109 |
| 6.9    | Herramientas de Depuración OpenFlow .....               | 109 |
| 6.10   | Casos de Uso de SDN.....                                | 110 |
| 6.10.1 | Virtualización de Red.....                              | 110 |
| 6.10.1 | Mejoras en la Seguridad y Aplicación de Políticas ..... | 112 |
| 6.10.2 | Movilidad Transparente .....                            | 113 |
| 6.10.3 | Redes con Conciencia de Aplicaciones.....               | 113 |
| 6.10.4 | Simplificación de la Gestión .....                      | 114 |

|        |                                                                     |     |
|--------|---------------------------------------------------------------------|-----|
| 6.10.5 | Video Streaming/Colaboración.....                                   | 115 |
| 6.10.6 | Virtualización de Funciones de Red (NFV).....                       | 115 |
| 7.     | Marco Metodológico.....                                             | 117 |
| 7.1    | Metodología.....                                                    | 117 |
| 7.1.1  | Estudio Teórico.....                                                | 117 |
| 7.1.2  | Propuesta.....                                                      | 117 |
| 7.1.3  | Diseño de la Investigación.....                                     | 117 |
| 7.1.4  | Alcance.....                                                        | 118 |
| 7.1.5  | Métodos y Técnicas de Recolección de Información.....               | 118 |
| 7.1.6  | Métodos y Técnicas de Análisis de la Información.....               | 120 |
| 7.1.7  | Procedimientos de la Investigación.....                             | 120 |
| 8.     | Marco Referencial.....                                              | 121 |
| 8.1    | Desafíos de SDN.....                                                | 121 |
| 8.1.1  | Desempeño.....                                                      | 121 |
| 8.1.2  | Escalabilidad.....                                                  | 121 |
| 8.1.3  | Disponibilidad y Sobrevivencia.....                                 | 121 |
| 8.1.4  | Costos.....                                                         | 121 |
| 8.2    | Caso de Estudio SDN en el Campus de la Universidad de Stanford..... | 122 |
| 8.2.1  | Objetivos Específicos.....                                          | 122 |
| 8.2.2  | Etapas de la Implementación SDN en Stanford.....                    | 122 |
| 8.2.3  | Fase 1: Prueba de Concepto.....                                     | 122 |
| 8.2.4  | Fase 2: Slicing y Escalabilidad de Despliegues SDN.....             | 127 |
| 8.2.5  | Fase 3: Implementación SDN Extremo-a-Extremo.....                   | 129 |
| 8.2.6  | Fase 4: Despliegue de Producción.....                               | 133 |
| 8.2.7  | Método de diagnóstico de desempeño de la red.....                   | 141 |
| 8.2.8  | Depuración de la red.....                                           | 142 |
| 8.2.9  | Análisis de Resultados.....                                         | 142 |
| 9.     | Consideraciones de Diseño SDN.....                                  | 145 |
| 9.1.1  | Consideración 1: Factibilidad Técnica y Comercial.....              | 145 |
| 9.1.2  | Consideración 2: Modelo de Despliegue.....                          | 146 |
| 9.1.3  | Consideración 3: Hoja de Ruta.....                                  | 148 |
| 9.1.4  | Consideración 4: Selección del Controlador.....                     | 150 |
| 9.1.5  | Consideración 5: Integración e Interoperabilidad.....               | 173 |
| 9.1.6  | Consideración 6: Gestión y Monitoreo.....                           | 173 |

|       |                                                |     |
|-------|------------------------------------------------|-----|
| 9.1.7 | Consideración 7: Soporte y Adiestramiento..... | 176 |
| 10.   | Conclusiones y Trabajos Futuros .....          | 177 |
|       | Referencias Bibliográficas .....               | 179 |

## Índice de Figuras

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| Figura 3.1: Antecedentes de SDN.....                                             | 24  |
| Figura 3.2: Red de Campus Empresarial .....                                      | 24  |
| Figura 3.3: Arquitectura de Red de Campus Empresarial Tradicional.....           | 26  |
| Figura 3.4: Plano de Control y Datos en el Hardware de Red.....                  | 27  |
| Figura 3.5: Modelo de Red Jerárquico de 3 Capas .....                            | 28  |
| Figura 3.6: Ejemplo de Red LAN con Jerarquía de 3 Capas.....                     | 29  |
| Figura 3.7: Ejemplo de Red Tradicional bajo la Regla 20/80 .....                 | 31  |
| Figura 4.1: Representación Lógica de una Arquitectura de Red SDN .....           | 34  |
| Figura 4.2: Topología SDN OpenFlow de Siete Switches SDN.....                    | 35  |
| Figura 4.3: Arquitectura de Capas SDN .....                                      | 36  |
| Figura 4.4: Arquitectura SDN basada en (a) Planos, (b) Capas y (c) Sistemas .... | 36  |
| Figura 4.5: Componentes de un Controlador SDN.....                               | 38  |
| Figura 4.6: APIs hacia el Norte de un Controlador SDN .....                      | 39  |
| Figura 4.7: Comunicación entre Controladores vía APIs Este/Oeste .....           | 40  |
| Figura 4.8: Arquitectura de Gestión Basada en Modelos .....                      | 42  |
| Figura 4.9: Topología de Red OpenFlow con FlowVisor .....                        | 45  |
| Figura 4.10: Representación Simplificada de un Router/Switch.....                | 48  |
| Figura 4.11: Red Tradicional de Routers/Switches .....                           | 48  |
| Figura 4.12: Red SDN de Routers/Switches .....                                   | 49  |
| Figura 4.13: Ejemplo del Modelo Basado en Dispositivos SDN .....                 | 52  |
| Figura 4.14: Redes Overlay.....                                                  | 53  |
| Figura 4.15: Modelo SDN Overlay.....                                             | 53  |
| Figura 4.16: Ejemplo del Modelo SDN Híbrido.....                                 | 55  |
| Figura 5.1: Arquitectura de Red OpenFlow .....                                   | 58  |
| Figura 5.2: Dispositivo SDN/OpenFlow .....                                       | 59  |
| Figura 5.3: Operación de Switches Solo-OpenFlow.....                             | 60  |
| Figura 5.4: Ejemplo de Red SDN con Switch OpenFlow-Híbrido .....                 | 61  |
| Figura 5.5: Ejemplo de una Tabla de Flujos OpenFlow 1.0.0.....                   | 65  |
| Figura 5.6: Procesamiento de Paquetes en un Switch OpenFlow 1.0.0.....           | 70  |
| Figura 5.7: Arquitectura de un Switch OpenFlow 1.1.0 .....                       | 72  |
| Figura 5.8: Procesamiento de Paquetes en un Switch OpenFlow 1.1.0.....           | 73  |
| Figura 5.9: Pipeline OpenFlow 1.1.0 .....                                        | 74  |
| Figura 5.10: Arquitectura de un Switch OpenFlow 1.2.0 .....                      | 77  |
| Figura 5.11: Arquitectura de un Switch OpenFlow 1.3.0 .....                      | 77  |
| Figura 5.12: Tablas de Egreso de un Switch OpenFlow 1.5.0 .....                  | 79  |
| Figura 6.1: Ejemplo de Creación de Red Mininet.....                              | 84  |
| Figura 6.2: Arquitectura General OpenDaylight.....                               | 95  |
| Figura 6.3: Arquitectura del Controlador Floodlight .....                        | 99  |
| Figura 6.4: Framework SDN Ryu .....                                              | 103 |
| Figura 6.5: Componentes e Interfaces del Framework SDN Ryu.....                  | 104 |
| Figura 6.6: Slicing en un Campus Universitario.....                              | 111 |
| Figura 6.7: Comparación Modelos de Red Clásicos y Equipos Virtuales. ....        | 115 |
| Figura 8.1: Infraestructura de Red Stanford Fase 1 Prueba de Concepto .....      | 123 |
| Figura 8.2: Infraestructura Etapa 1 Prueba de Concepto Extendida .....           | 124 |
| Figura 8.3: Infraestructura de Red Stanford Edificio William Gates Fase 2.....   | 128 |

|                                                                                 |     |
|---------------------------------------------------------------------------------|-----|
| Figura 8.4: Implementación de Capa de Virtualización Stanford Fase 2 .....      | 130 |
| Figura 8.5: Infraestructura de Red Stanford Fase 3 .....                        | 131 |
| Figura 8.6: Red WiFi OpenRoads en el edificio William Gates.....                | 134 |
| Figura 8.7: Despliegue OpenFlow del Edificio William Gates.....                 | 135 |
| Figura 8.8: Despliegue OpenFlow en el Edificio Paul Allen CIS/CISX .....        | 136 |
| Figura 8.9: Infraestructura de Monitoreo Migración a OpenFlow en Stanford ..... | 137 |
| Figura 8.10: Medición Tiempo de Establecimiento de Flujos. ....                 | 139 |
| Figura 8.11: Medición Retardo RTT. ....                                         | 139 |
| Figura 8.12: Medición Retardo wget.....                                         | 139 |
| Figura 8.13: Medición Uso de CPU. ....                                          | 140 |
| Figura 8.14: Medición Número de Flujos Activos. ....                            | 140 |
| Figura 8.15: Medición Tasa de Llegada de Flujos.....                            | 140 |
| Figura 8.16: Medición Número de Usuarios. ....                                  | 140 |
| Figura 9.1: Despliegue de Migración Greenfield .....                            | 147 |
| Figura 9.2: Despliegue de Migración Mixto .....                                 | 147 |
| Figura 9.3: Despliegue de Migración Híbrido .....                               | 147 |
| Figura 9.4: Diferentes Aspectos para Seleccionar Controladores SDN .....        | 150 |
| Figura 9.5: Arquitectura de un Controlador con Soporte Multihilo .....          | 154 |
| Figura 9.6: Pruebas Cbench - Variación Cantidad de Switches.....                | 157 |
| Figura 9.7: Pruebas Cbench - Variación Cantidad de Hilos .....                  | 158 |
| Figura 9.8: Pruebas Cbench - Variación Cantidad de Switches.....                | 158 |
| Figura 9.9: Esquema de Recuperación a Fallas en una Red Tradicional.....        | 160 |
| Figura 9.10: Esquema de Recuperación a Fallas en una Red SDN.....               | 161 |
| Figura 9.11: Condición Solo Punto de Falla de un Controlador.....               | 161 |
| Figura 9.12: Esquema de Alta Disponibilidad de Controladores SDN.....           | 161 |
| Figura 9.13: Jerarquía de Controladores.....                                    | 163 |
| Figura 9.14: Ejemplo de Opciones de Colocación de Controladores SDN.....        | 164 |
| Figura 9.15: Vectores de Seguridad SDN .....                                    | 166 |
| Figura 9.16: SDN Confiable y Segura .....                                       | 168 |
| Figura 9.17: Infraestructura de Monitoreo SDN - Recolección de Métricas.....    | 174 |

## Índice de Tablas

|                                                                              |     |
|------------------------------------------------------------------------------|-----|
| Tabla 4.1: Protocolos de Gestión SDN.....                                    | 43  |
| Tabla 4.2: Comparación Operacional entre Redes Tradicionales y Redes SDN...  | 50  |
| Tabla 4.3: Comparación Funcional entre Redes Tradicionales y Redes SDN ..... | 51  |
| Tabla 5.1: Campos de la Tabla de Flujos de un Switch OpenFlow 1.0.0 .....    | 63  |
| Tabla 5.2: Campos Header Fields OpenFlow Versión 1.0.0 .....                 | 64  |
| Tabla 5.3: Mensajes OpenFlow 1.0.0.....                                      | 67  |
| Tabla 5.4: Tabla de Flujos de un Switch OpenFlow 1.1.0 .....                 | 72  |
| Tabla 5.5: Lista de Instrucciones OpenFlow 1.1.0.....                        | 72  |
| Tabla 5.6: Campos Match de un Switch OpenFlow 1.1.0.....                     | 73  |
| Tabla 5.7: Tabla de Grupos de un Switch OpenFlow 1.1.0 .....                 | 75  |
| Tabla 5.8: Campos de la Tabla Meter de un Switch OpenFlow 1.3.0.....         | 78  |
| Tabla 5.9: Cuadro Comparativo de Especificaciones OpenFlow.....              | 80  |
| Tabla 6.1: Implementaciones de Switches SDN basados en Software .....        | 89  |
| Tabla 6.2: Lista de Dispositivos con Soporte Pantou .....                    | 90  |
| Tabla 6.3: Switches OpenFlow Comerciales Basados en Hardware.....            | 91  |
| Tabla 6.4: Herramientas de Depuración OpenFlow .....                         | 110 |
| Tabla 6.5: Dominios de Control en Procesa y Políticas de Alto Nivel .....    | 113 |
| Tabla 7.1: Diseño de la Investigación.....                                   | 118 |
| Tabla 8.1: Método de Diagnóstico de Desempeño de la Red .....                | 141 |
| Tabla 9.1: Plantilla Línea Base para Seleccionar Controladores .....         | 152 |
| Tabla 9.2: Escalabilidad de Flujos (en Millones de Flujos/Segundo).....      | 154 |
| Tabla 9.3: Casos de Uso SDN .....                                            | 171 |
| Tabla 9.4: Comparación de Controladores SDN Basado en Casos de Uso.....      | 172 |



# 1. Introducción

El estudio de las redes de campus empresariales ha ido creciendo en los últimos años debido a la creciente evolución en los servicios de red que se ofrecen a sus usuarios y a la incorporación de nuevas tecnologías. Las redes de campus son cada día más grandes y complejas y manejan grandes volúmenes y tipos de tráfico. Una red de campus empresarial está conformada por dispositivos de redes LAN (Local Area Networking) Ethernet que se conectan entre sí dentro de un mismo edificio y entre edificios cercanos pertenecientes a una empresa, organización o institución pública, como un campus universitario.

Las redes de campus empresariales han pasado de ofrecer servicios de conectividad a convertirse en proveedores de servicios de red inteligentes orientados a soportar de manera confiable, escalable y segura un gran número de aplicaciones, usuarios y dispositivos. En la actualidad los usuarios de la red solicitan servicios de colaboración de voz, video, datos y chat sobre dispositivos móviles multi-plataforma y acceden a recursos de IT hospedados en el campus o en alguna nube pública o privada. Estos servicios deben ser entregados de manera confiable y oportuna para ofrecer una buena calidad de experiencia al usuario independientemente del lugar, momento y dispositivo utilizado. Aunado a esta realidad las redes son cada día más complejas y dinámicas e incorporan nuevas tecnologías. IoT (Internet of Things), BigData, Virtualización de Cómputo, Virtualización de Redes, Virtualización de Funciones de Red NFV (Network Function Virtualization), Servicios de Nube, Movilidad y Transformación Digital son algunas de las tecnologías de red que han ido adquiriendo un espacio en los campus empresariales y que demandan un aprovisionamiento de recursos de red dinámico y flexible. En este nuevo contexto de mayor cantidad de aplicaciones y servicios, mayor volumen de tráfico, mayor demanda de servicios y mayor movilidad se requiere una arquitectura de red abierta, inteligente, ágil, escalable y flexible capaz de aprovisionar nuevos servicios y aplicaciones a sus usuarios de manera expedita.

Gracias a un esfuerzo de la industria liderado por la fundación ONF [1] (Open Network Foundation) se están redefiniendo las arquitecturas de redes a través del desarrollo de SDN (Software Defined Networking). SDN es una arquitectura de red emergente que desacopla el plano de control del plano de datos de los dispositivos de comunicación y centraliza el plano de control en una sola entidad lógica denominada controlador SDN que es capaz de proveer un control programático de la infraestructura de red. Esta separación de planos permite la programación de la red al permitir la interacción directa entre las aplicaciones y los controladores de la red mediante APIs (Application Programming Interface) de programación. Un Controlador SDN interactúa con aplicaciones externas mediante APIs Northbound e interactúa con dispositivos SDN a través de APIs Southbound.

Un controlador SDN centraliza la inteligencia y el estado de la red, abstrae la infraestructura de red a las aplicaciones y notifica las reglas de reenvío de tráfico a

los dispositivos SDN. Con esta nueva arquitectura se pueden obtener grandes beneficios en la operación de la red permitiendo programar la red, agilizar el aprovisionamiento de servicios, simplificar la administración de la red y automatizar el despliegue de servicios de red. Por estas razones las empresas, universidades e instituciones públicas están interesadas en incorporar SDN en su infraestructura de red y toma connotación la importancia de desarrollar un estudio que identifique las consideraciones de diseño claves para desplegar redes SDN en campus empresariales.

El resto del trabajo se encuentra organizado de la siguiente manera:

**Capítulo 2:** Presenta la descripción del problema.

**Capítulo 3:** Revisa los antecedentes a SDN y presenta las características y limitaciones de las redes de campus tradicionales.

**Capítulo 4:** Describe la arquitectura de red SDN, compara las arquitecturas de redes de campus tradicionales con la arquitectura de red SDN y explica los modelos de despliegue de SDN.

**Capítulo 5:** Explica el protocolo OpenFlow.

**Capítulo 6:** Reseña el contexto actual de SDN donde se muestran las herramientas de simulación para SDN disponibles, se menciona el estado actual de los componentes de hardware y software de la arquitectura, se identifican las potencialidades de OpenFlow, se describen los tipos de aplicaciones SDN y se mencionan algunos casos de uso.

**Capítulo 7:** Marco Metodológico.

**Capítulo 8:** Marco Referencial.

**Capítulo 9:** Enumera las consideraciones de diseño para el despliegue de Redes SDN en Campus empresariales.

**Capítulo 10:** Conclusiones obtenidas producto de la investigación realizada y trabajos futuros.

## **2. El Problema**

En este capítulo se exponen los argumentos que justifican el desarrollo de un estudio que identifique las principales consideraciones de diseño para desplegar redes SDN en campus empresariales.

### **2.1 Planteamiento del Problema**

El creciente volumen de servicios de movilidad: voz, chat y video en los campus empresariales y la incorporación de tecnologías de virtualización, movilidad, BigData, IoT y nube son algunos impulsores que están dirigiendo a las universidades, centros de investigación y fabricantes de equipos a reexaminar sus arquitecturas de red

Las arquitecturas de redes tradicionales de los campus empresariales están basadas en el modelo de plano de control distribuido y fueron concebidas cuando la computación cliente-servidor era dominante y el volumen de cambios en la red era controlado. En una red de campus empresarial basada en un plano de control distribuido los routers y switches de la red deben aprender de manera dinámica la alcanzabilidad de las redes de sus vecinos, sus capacidades y el estado de sus enlaces. Adicionalmente para que la red sea funcional los administradores de la red deben configurar los protocolos, servicios y políticas de red en cada dispositivo de red, uno a la vez, a través de interfaces de línea de comandos CLI (Command Line Interface) o a través de interfaces web GUI (Graphical Unit Interface). Esto se convierte en un proceso lento y complejo que involucra la configuración manual de parámetros de protocolos de red, activación de puertos, configuración de listas de control de acceso ACL (Access Control List), políticas QoS (Quality of Service) y políticas de seguridad a cientos de dispositivos.

Junto a esta realidad existe una marcada dependencia con los fabricantes de los equipos. Los equipos de red son adquiridos con hardware y software integrado verticalmente, cerrado y propietario. Esta condición impide que se pueda modificar o mejorar el software de los equipos. Bajo este escenario un investigador interesado en implementar una nueva funcionalidad o mejorar un protocolo de red debe esperar el ciclo completo de desarrollo de la nueva funcionalidad por parte del fabricante que incluye las fases de diseño, pruebas y estandarización, pudiendo llevar meses e inclusive años en materializarse.

Por estas razones expuestas se requiere la implementación de arquitecturas de redes abiertas e interoperables que ofrezcan mayor flexibilidad y agilidad para aprovisionar servicios e innovar en nuevas áreas de investigación.

### **2.2 Justificación del Problema**

SDN es una arquitectura de red abierta, flexible y programable que centraliza el plano de control de la red en entidades de control encargadas de publicar las

reglas de reenvío de tráfico a los dispositivos SDN bajo su dominio. SDN permite aprovisionar servicios y políticas de red de forma rápida y confiable, a la vez, que facilita la innovación, simplifica los ciclos de desarrollo de nuevas tecnologías y disminuye el grado de dependencia de la infraestructura con los fabricantes de equipos de red. Por estas razones resulta interesante desarrollar un estudio que revele las consideraciones de diseño y las opciones de despliegue para implementar una arquitectura SDN en ambientes de campus empresariales.

### **2.2.1 Objetivo General**

El objetivo general de esta investigación es proponer las consideraciones de diseño para desplegar redes SDN en campus empresariales de manera confiable, escalable y segura.

### **2.2.2 Objetivos Específicos**

- Hacer un análisis general de la forma en que operan las redes tradicionales.
- Analizar el modo de operación de un plano de control centralizado en ambientes SDN e identificar los componentes necesarios para implementar una arquitectura de red SDN.
- Describir el funcionamiento y el estado actual de las especificaciones del protocolo de comunicación estándar OpenFlow [2] que se utiliza para la comunicación entre el controlador y los dispositivos de una red SDN.
- Investigar el estado actual de los componentes de la arquitectura SDN.
- Describir los tipos de aplicaciones SDN existentes.
- Identificar los principales casos de uso SDN.
- Analizar casos de estudio de despliegue SDN en campus empresariales.
- Determinar los aspectos claves y las principales consideraciones para implementar SDN en campus empresariales.

### 3. Antecedentes de SDN

Las redes de datos tuvieron un gran impulso en la década de 1960, momento en el cual un gran número de investigadores crearon los estándares que han hecho posible la comunicación sobre las redes de datos. Estándares como paquetes, conmutación de paquetes e Interface Message Processor (versión previa de un switch o router) permitieron que el primer mensaje de ARPANET (Advanced Research Projects Agency Network) fuera transmitido entre la Universidad de California y el Instituto de Investigación de la Universidad de Stanford el 29 de octubre de 1969<sup>1</sup>.

En la década de 1970 surgen los protocolos Ethernet y la pila de protocolos TCP/IP, seguida por la estandarización del protocolo LAN Ethernet para permitir la comunicación de routers y bridges.

En 1989 la compañía Kapanan introduce el sucesor a los bridges en la forma del primer switch multi-puerto, como una alternativa económica y de alto rendimiento a los routers utilizados en las redes de *backbone* y aparece el fabricante Cisco en el mercado de los switches.

A finales de 1980 se crea el protocolo SNMP [3] (Simple Network Management Protocol) para gestionar y monitorear dispositivos de red y gana aceptación en las implementaciones de gestión de redes. SNMP utiliza una aplicación de gestión centralizada y un conjunto de agentes distribuidos en los dispositivos de red que informan capacidades y estado de variables de utilización y rendimiento.

A mediados de la década de 1990 se plantea el framework de Redes Activas [4] en el cual los usuarios pueden inyectar programas a los mensajes que viajan por una red de nodos con capacidad de extraer y ejecutar programas. Se plantean 2 enfoques para la implementación de Redes Activas: (1) el enfoque discreto de switches programables, en el cual el código a ejecutar en los nodos se establece por mecanismos fuera de banda y (2) el enfoque Capsula, donde el código viaja con los paquetes en banda. La tecnología de Redes Activas puede ser aprovechada para agilizar y flexibilizar la gestión y el monitoreo de la red como se describe en la propuesta *Smart Packets* [5]. La propuesta persigue utilizar la tecnología de Redes Activas para gestionar la red mediante la programación de nodos gestionados. De esta manera un centro de gestión puede enviar programas a un conjunto de nodos gestionados. El enfoque brinda los siguientes beneficios: (1) el contenido de la información retornada al centro de gestión puede ser personalizada limitando la cantidad de datos requiriendo examinación, (2) se pueden enviar reglas a los nodos gestionados en los programas para identificar y corregir problemas directamente en los nodos gestionados y (3) se reducen significativamente las operaciones de medición y monitoreo a un solo paquete desde un centro de gestión.

---

<sup>1</sup> [http://www.lk.cs.ucla.edu/internet\\_first\\_words.html](http://www.lk.cs.ucla.edu/internet_first_words.html)

Otro hito importante en la programación de las redes fue la creación del grupo COMET<sup>2</sup> en la Universidad de Columbia con el objetivo de ampliar el conocimiento de las arquitecturas de software de las redes. El grupo COMET ha jugado un rol importante en el establecimiento de nuevos foros internacionales, como el grupo de trabajo OPENSIG [6] (Open Signalling Community), cuyas metas fueron la creación de redes programables y abiertas que pudieran aceptar nuevos servicios y funcionalidades y promover la investigación en arquitecturas y señalizaciones abiertas y en la programación de redes para ATM, Internet y redes móviles. OPENSIG vio esencial la separación del software de control del hardware de comunicaciones y persiguió esta meta por una década, con talleres anuales hasta el año 2003.

A través de talleres en OPENSIG en el año 2011 se obtuvieron un número de artículos basados en la programación de las redes, incluyendo la propuesta ForCES [7] (Forwarding and Control Element Separation) del grupo IETF (Internet Engineering Task Force). El grupo de trabajo OPENSIG consideró la necesidad de separar los planos de control y reenvío en los dispositivos de una red IP<sup>3</sup> para facilitar la innovación.

Las metas de OPENSIG fueron retomadas por otros grupos de investigación. En mayo del 2003 el IETF convoca una reunión para discutir las tecnologías de gestión de redes con los principales operadores de red y desarrolladores de protocolos. De esta reunión se determina que la configuración de la gestión de la red mediante SNMP es compleja y se propone la configuración de dispositivos mediante interfaces CLI [8] .

En agosto del 2003 se publica un borrador (Internet-draft) para NETCONF<sup>4</sup> en el cual se detalla un mecanismo para gestionar dispositivos de red utilizando mensajes codificados en XML [9] (eXtensible Markup Language) entre un cliente y un dispositivo de red. El protocolo fue estandarizado en el año 2006 y fue actualizado en el año 2011.

En noviembre del 2003 el IETF publicó un memorándum<sup>5</sup> clarificando su trabajo previo en el modelo ForCES. El memorándum describe como se puede avanzar el desarrollo de los planos de control y datos si estos se separan y sugieren la necesidad de un protocolo estándar para la comunicación entre los dispositivos de red y el controlador de la red.

En agosto de 2004 se describe una plataforma RCP [10] (Routing Control Platform) que guarda similitud con la propuesta ForCES. Los autores consideran que los routers deben ser optimizados para consultar y reenviar paquetes tan

---

<sup>2</sup> <http://comet.columbia.edu/about.htm>

<sup>3</sup> <http://tools.ietf.org/html/draft-anderson-forces-req-02>

<sup>4</sup> <http://tools.ietf.org/html/draft-ietf-netconf-prot-00>

<sup>5</sup> <http://tools.ietf.org/html/rfc3654>

rápido como sea posible, mientras que el procesamiento de los protocolos de enrutamiento debe ser tomado por un sistema de control centralizado. Este modelo busca disponer de una visión global de la red desde un punto central, facilitar la gestión, agilizar las actualizaciones de los protocolos de enrutamiento y minimizar las inconsistencias en la red.

En noviembre del 2004 en la conferencia HotNets-III Sigcomm<sup>6</sup> se presenta el artículo *Network-Wide Decision Making: Toward A Wafer-Thin Control Plane* [11] que mueve la toma de decisiones de routers individuales a servidores separados. Este artículo guarda relación con el proyecto *A Clean Slate 4D Approach to Network Control and Management* [12] de la Universidad Carnegie Mellon en el cual se describe la separación de los planos de decisión y datos, equivalente al modelo ForCES. El enfoque 4D describe el reemplazo de la funcionalidad de control propietaria por controladores de software que corren en servidores separados. El modelo 4D fue demostrado con éxito en el año 2007 utilizando el software Tesseract [13].

En el año 2007 la Universidad de Stanford inicia el programa Clean Slate con el objetivo de rediseñar el Internet con el beneficio de los últimos avances y retrospectivas<sup>7</sup>. El proyecto se desarrolló en cuatro áreas principales de investigación, una de las cuales se llamó OpenFlow and SDN. Los aspectos fundamentales de SDN, como se describe en el sitio web Clean Slate Program<sup>8</sup> contemplan la separación de los planos de datos y control, la disposición de una interfaz uniforme agnóstica de vendedor que la conexión de los planos de control y datos y la disposición de un plano de control lógico centralizado lo cual se encuentra alineado al enfoque desarrollado en los modelos ForCES y 4D.

En octubre del 2007, la Universidad de Stanford en colaboración con la Universidad de California publican el artículo *Ethane: Taking Control of the Enterprise* [14]. Ethane propone utilizar switches basados en flujos controlados por un computador centralizado.

A inicios del año 2008 se publica el artículo OpenFlow: Enabling Innovation in Campus Networks [2]. El artículo describe un método para controlar tablas de flujos en switches de red desde un controlador central al igual que Ethane.

En el año 2008 en un boletín del ACM SIGCOMM se describe una arquitectura SDN con un controlador NOX [15].

En la Figura 3.1 se muestra un diagrama resumen de los principales eventos que anteceden a SDN desde la aparición de las arquitecturas de Redes Activas hasta la creación del primer controlador SDN basado en OpenFlow.

---

<sup>6</sup> <http://conferences.sigcomm.org/hotnets/2004/hotrev.pdf>

<sup>7</sup> <http://cleanslate.stanford.edu/index.php>

<sup>8</sup> <http://cleanslate.stanford.edu>

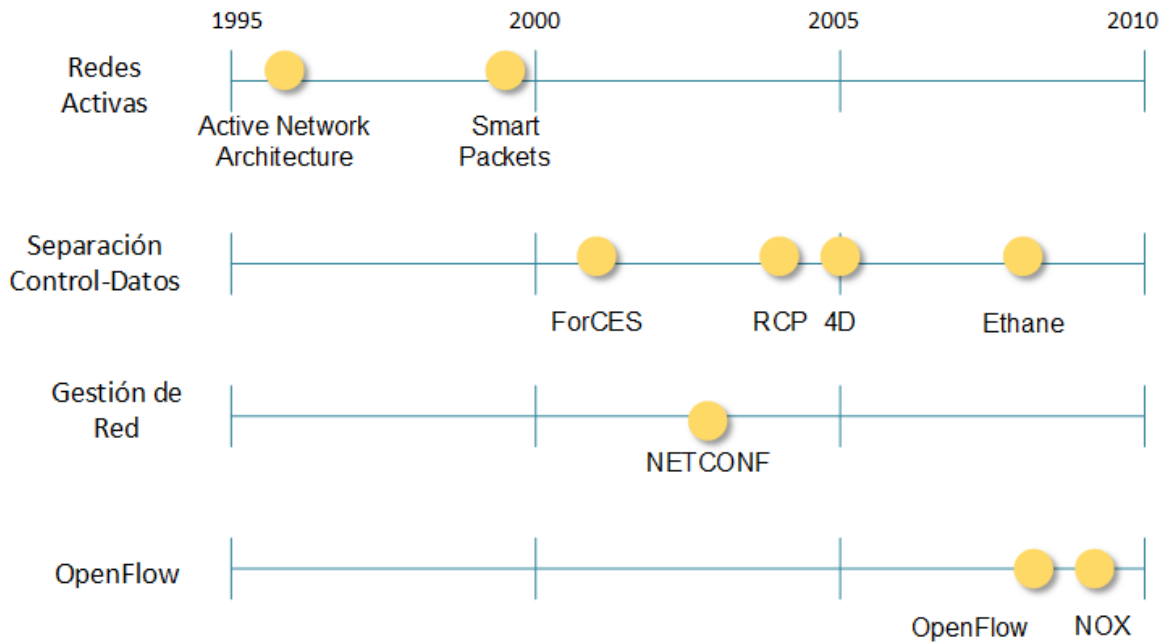


Figura 3.1: Antecedentes de SDN

### 3.1 Redes de Campus Empresariales Tradicionales

Una red de campus empresarial está conformada por un edificio o un grupo de edificios conectados entre sí a una red empresarial que consta de muchas LANs. Generalmente, un campus se limita a un área geográfica fija, que abarca varios edificios vecinos, por ejemplo, un complejo industrial o un campus universitario (ver Figura 3.2).

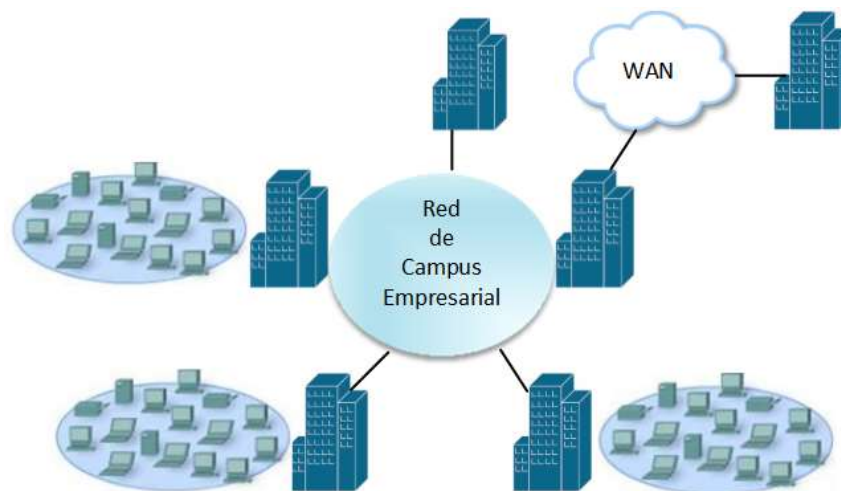


Figura 3.2: Red de Campus Empresarial

En estos entornos las oficinas regionales y los trabajadores móviles se conectan al campus central para acceder a los datos, servicios y aplicaciones de la empresa. Generalmente, los nodos de la red del campus se encuentran interconectados por

medios de transmisión fijos de fibra óptica 1/10 GE (Gigabit Ethernet) y por medios de transmisión inalámbricos WLAN (Wireless Local Area Network).

El principal reto en estos ambientes es lograr que la red se adapte a las demandas de tráfico de los usuarios y de las aplicaciones de red. Para lograr éste objetivo es necesario comprender el funcionamiento de las redes tradicionales. En éste capítulo se revisan las características, los requerimientos y las limitaciones de las redes de campus tradicionales.

### **3.1.1 Características de las Redes de Campus Tradicionales**

Las redes de campus empresariales de escala mediana y grande están conformadas por cuatro bloques funcionales (ver Figura 3.3 tomada de [16]): (1) campus empresarial, (2) borde de la empresa (Enterprise Edge), (3) borde del Proveedor de Servicios de Comunicaciones y (4) sucursales remotas. El campus empresarial está constituido por el módulo de infraestructura del campus y el módulo del Centro de Datos. La infraestructura de red del campus está compuesta por los equipos LAN fijos e inalámbricos. La granja de servidores está formada por la red, los servidores y el centro de almacenamiento. El resto de los bloques y módulos se conectan al campus empresarial a través de tecnologías de red tradicional LAN y WAN (Wide Area Network) [17]. Una WAN no tiene restricciones geográficas y puede interconectarse a otras redes o puede interconectar muchas LANs entre sí<sup>9</sup>.

### **3.1.2 Tecnologías de Redes de Campus Tradicionales**

Las redes de campus empresariales están conformadas por dispositivos de red que interconectan los edificios de una organización a través de tecnologías LAN Ethernet fija e inalámbrica y tecnologías WAN incluyendo switches, routers y firewalls pertenecientes a un área geográfica específica, generalmente dentro de un mismo edificio o grupo de edificios adyacentes. En esta tesis el alcance de las tecnologías de redes de campus empresariales está limitado a redes LAN Ethernet. En la siguiente sección se describen los principales tipos de dispositivos de red presentes en los campus empresariales.

#### **Switch**

Un switch es un dispositivo de red que opera en la capa 2 del modelo de referencia OSI (Open System Interconnection) y es capaz de reenviar tráfico de un puerto de entrada a un puerto de salida al cual se encuentran conectados dos dispositivos finales, o entre un puerto de entrada y un puerto de salida que se conecta a otro dispositivo de red como un router u otro switch en el mismo segmento de red. El switch registra las direcciones MAC (Medium Access Control) de todos los dispositivos conectados a un segmento de red en una tabla de switching, incluyendo las direcciones MAC de computadoras, laptops, servidores de red, teléfonos IP y puertos de un router entre otros. Cuando un paquete llega al

---

<sup>9</sup> <https://kb.iu.edu/d/agki>

switch, este consulta la dirección MAC de destino en la tabla de switching y reenvía el paquete al puerto del switch donde se encuentra conectado el nodo destino [17] .

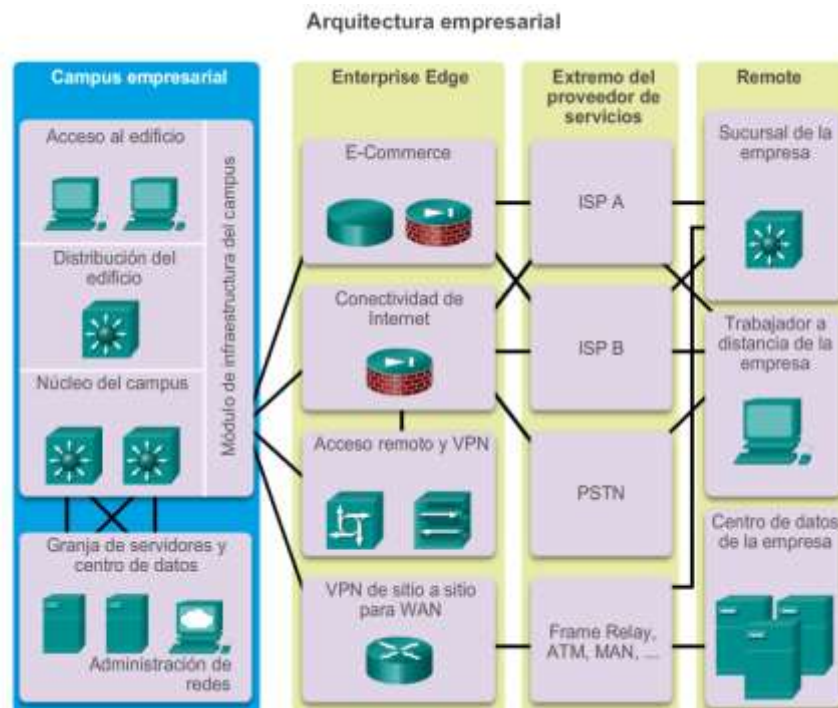


**Figura 3.3: Arquitectura de Red de Campus Empresarial Tradicional**

Los switches operan a nivel de la capa 2 del modelo OSI y cualquier frame cuyo destino pertenezca a otra subred distinta del segmento IP de origen debe ser procesada por un dispositivo Capa 3 OSI, como un router o un switch multicapa capa2/capa3. Una red LAN puede segmentarse en subredes lógicas mediante el establecimiento de VLANs (Virtual LAN) para simplificar la red, aislar tipos de tráfico y disminuir el tamaño de los dominios de colisión de la tecnología Ethernet. Un dominio de colisión se puede definir como el conjunto de dispositivos que pertenecen a un mismo segmento de red o a una VLAN en particular y que pueden recibir mensajes de broadcast en el evento que la dirección MAC de destino de un paquete no se encuentre en la tabla de switching o en el evento que se utilice una aplicación que requiera enviar un mismo paquete a un conjunto de nodos miembros de una dirección multicast.

## Router

Un router es un dispositivo de red que opera en la capa 3 del modelo OSI y permite conectar segmentos que corren protocolos de red diferentes como redes LANs y WANs. La función principal del router es enviar un paquete de un puerto origen a un puerto destino basado en la dirección IP de destino de un paquete y en la selección de la mejor ruta para alcanzar el destino del paquete. La selección de la mejor ruta se hace tomando en consideración una métrica como el ancho de banda del enlace que conecta a un puerto, el retardo, el número de saltos u alguna otra métrica presente en los caminos que unen un nodo origen y un nodo

destino. Los routers toman sus decisiones de reenvío apoyándose en la información de alcanzabilidad de subredes IP presentes en una tabla de enrutamiento que puede ser alimentada de manera dinámica a través de un protocolo de enrutamiento o de manera manual por un administrador de la red. El router también puede basar sus decisiones de reenvío basados en políticas de seguridad, QoS o de negocio que indiquen a cual puerto se deben enviar los paquetes dependiendo de la información IP presente en el paquete.

### Planos de Datos, Control y Gestión

Un dispositivo de red consta de tres planos: (1) Plano de datos, (2) plano de control y (3) plano de gestión. El plano de datos se refiere a la parte del hardware encargada de las siguientes funciones: (1) reenviar paquetes de un puerto de comunicación a otro, (2) modificar campos de la cabecera de un paquete y (3) filtrar paquetes basado en reglas de configuración definidas por un plano de control. Generalmente, el plano de control consiste de un firmware desarrollado y mantenido por un fabricante de red. El plano de gestión es la parte del plano de control utilizada para propósitos de monitoreo y administración de los dispositivos.

En la Figura 3.4 se muestran los planos de control y datos de un dispositivo de red. En la parte superior se encuentra el plano de control que ejecuta las funciones de enrutamiento, selección de mejor ruta y definición de reglas de filtrado de datos.

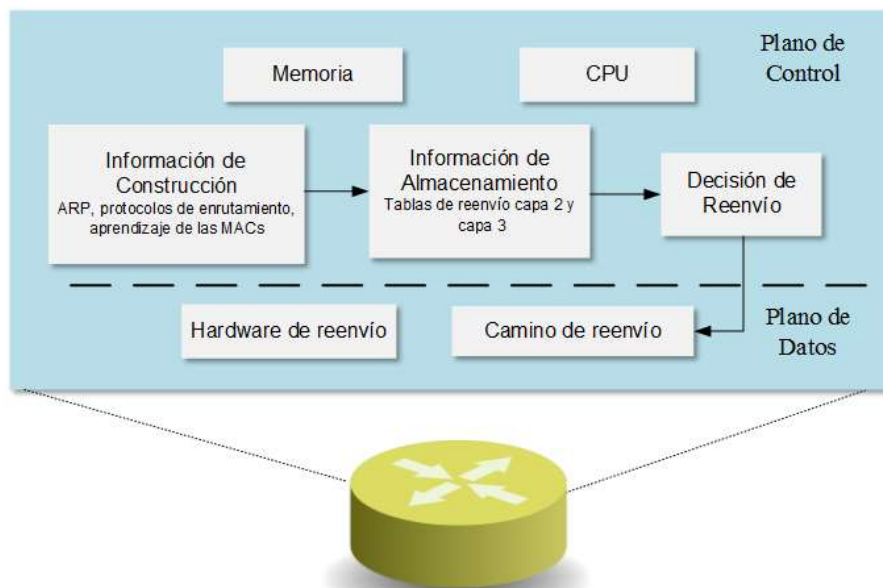


Figura 3.4: Plano de Control y Datos en el Hardware de Red

En la parte inferior se encuentra el plano de datos que ejecuta las funciones de reenvío de datos, en la cual se extraen los campos de cabecera de los paquetes entrantes y se consulta el puerto de salida en la tabla de reenvío del dispositivo.

Una vez determinado el puerto saliente, el paquete es reenviado al puerto de salida correspondiente a través de la fábrica de conmutación del dispositivo.

### Modelo de Redes Jerárquicas

Las redes de campus empresariales tradicionales se dividen en capas independientes que conforman un modelo jerárquico. La división de una red en capas ofrece beneficios tanto a nivel operacional, como a nivel de diseño, ya que se facilita la escalabilidad, se pueden aislar y restringir las fallas dentro de una capa y se pueden seleccionar los dispositivos de red adecuados en base a las características de desempeño y funcionalidades requeridas en cada capa.

En una red LAN jerárquica de campus empresarial tradicional se incluyen tres capas como se indica en la Figura 3.5.

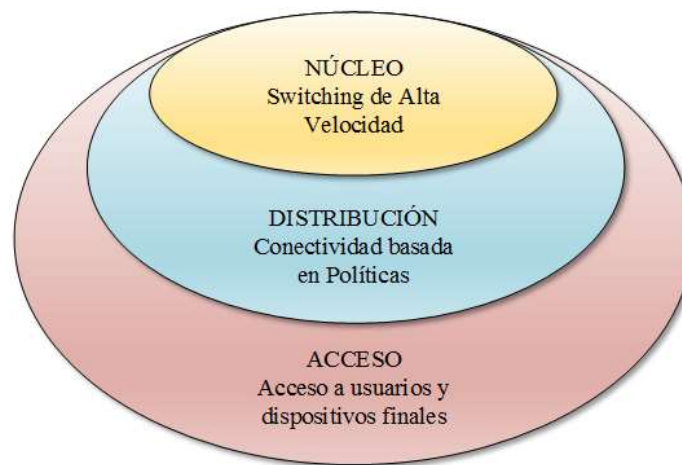


Figura 3.5: Modelo de Red Jerárquico de 3 Capas

- **Capa de Acceso (Access Layer):** Proporciona acceso a los recursos de red a los usuarios y grupos de trabajo a través de switches de acceso.
- **Capa de Distribución (Distribution Layer):** Proporciona servicios de conectividad basados en políticas y define el borde entre las capas de acceso y núcleo. Esta capa sirve de agregación de tráfico de los switches de acceso para permitir la comunicación entre los departamentos y usuarios de una empresa. Entre sus funciones se encuentran: (1) enrutar tráfico entre departamentos, (2) definir dominios de broadcast y multicast, (3) proporcionar servicios de seguridad y filtrado de paquetes y (4) proveer la comunicación entre las capas de núcleo y acceso.
- **Capa de Núcleo (Core Layer):** Proporciona un transporte rápido de paquetes entre los switches de distribución dentro del campus empresarial y se encarga de transportar grandes volúmenes de tráfico de manera confiable basado en principios de baja latencia, alta velocidad y resiliencia.

En la Figura 3.6 se muestra un despliegue de una red de campus tradicional basada en el modelo jerárquico de 3 capas: núcleo, distribución y acceso.

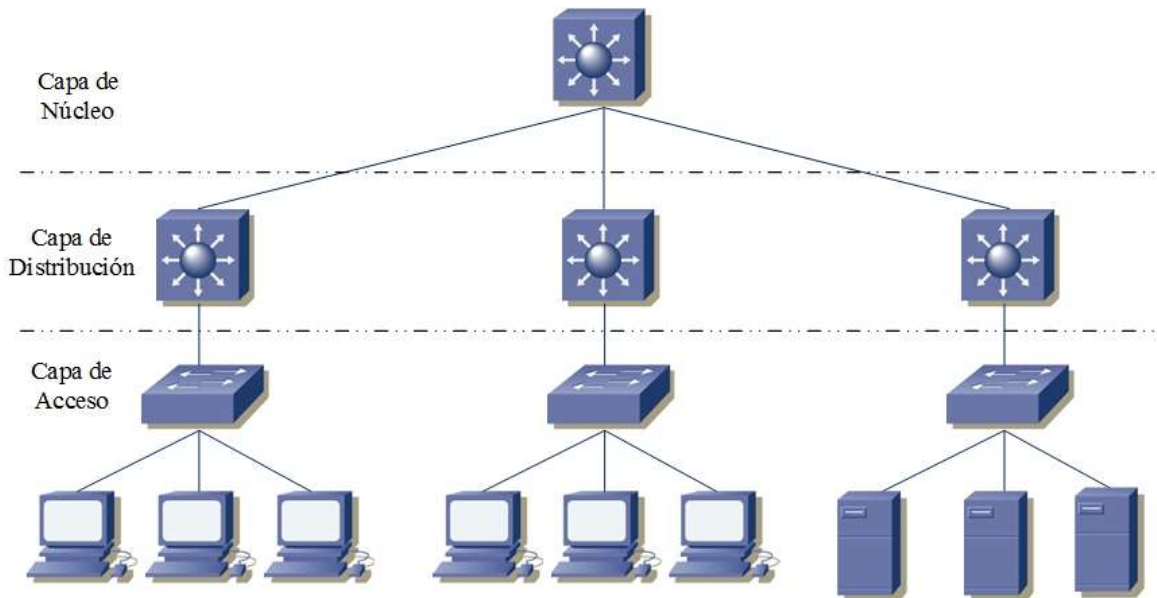


Figura 3.6: Ejemplo de Red LAN con Jerarquía de 3 Capas

### VLANS (Virtual LANs)

Una VLAN es una red lógica construida en el tope de una red LAN física. Los miembros de una VLAN pueden pertenecer a segmentos de red físicos diferentes y recibir tráfico unicast, multicast o broadcast de otros miembros pertenecientes a la misma VLAN lógica. Bajo un esquema de VLANs se puede conectar un equipo a cualquier puerto del switch y un administrador establece la asociación entre el puerto y la VLAN correspondiente. Para permitir la comunicación de nodos entre VLANs diferentes se debe contar con un dispositivo de red capa 3 como un router o un switch multicapa capa2/capa3 que permita comunicar las VLANs entre sí.

### 3.1.3 Problemas de Desempeño en las Redes de Campus Tradicionales

Los problemas de desempeño en las redes de campus tradicionales están asociados a las colisiones, el tráfico de broadcast y el ancho de banda.

#### Colisiones

Un dominio de colisión es un segmento de red físico de una red LAN Ethernet donde las tramas de dos o más hosts conectados pueden interferir entre sí. A medida que se incorporan más hosts en un mismo segmento aumentan las posibilidades de que 2 hosts intenten transmitir al mismo tiempo, generando colisiones y disminuyendo el rendimiento de la red. En una red Ethernet los hosts que participan en una colisión deben esperar un intervalo de tiempo aleatorio antes de intentar retransmitir tráfico al medio de transmisión. Para resolver el problema de las colisiones se utilizan switches capa 2 que separan el dominio de

colisión en múltiples dominios de colisión. En un switch capa 2 cada puerto forma un dominio de colisión aislado.

### **Ancho de Banda**

El ancho de banda de un segmento se mide por la cantidad de datos que puede transmitir el segmento en un momento dado. Un problema común asociado al ancho de banda es la distancia entre dos nodos de comunicación debido a la degradación que sufre la señal a medida que se aleja de un emisor en un medio de transmisión. Una opción para superar esta situación es mantener las distancias entre los nodos de comunicaciones dentro de las recomendaciones del protocolo de comunicación y diseñar la red con una segmentación adecuada mediante routers y switches. En una red Ethernet la distancia máxima permitida entre dos nodos es de 100 mts.

### **Broadcast y Multicast**

Un dominio de broadcast es una red lógica donde todos los hosts pertenecen a una misma subred o VLAN y pueden recibir simultáneamente un mismo mensaje de difusión (broadcast) de un nodo emisor. El tráfico de broadcast es un elemento de control necesario en el funcionamiento de las redes TCP/IP a la vez que puede ser considerado perjudicial por el consumo excesivo de recursos de ancho de banda. Una red con tráfico excesivo de broadcast puede verse afectada en su rendimiento. El tráfico de multicast también puede causar problemas si no se configura de manera adecuada. Un tráfico de multicast es un caso especial de tráfico de broadcast en el cual un nodo emisor envía un mensaje a un subconjunto de hosts destino. Si se tienen grupos grandes de multicast o aplicaciones multicast intensivas en ancho de banda como IPTV (Internet Protocol Television) se puede consumir la mayoría del ancho de banda de la red en detrimento del resto de las aplicaciones. Para resolver los problemas de broadcast se recomienda segmentar la red con routers o switches multicapa.

### **La Regla 80/20**

En una red tradicional se colocan los usuarios y grupos de trabajo en una misma localidad física. En este tipo de redes se sigue la regla 80/20, la cual supone que el 80% del tráfico de los usuarios permanece en el mismo segmento de red local y solamente el 20% restante cruza los dispositivos de borde a otros segmentos. En este escenario el diseño de la red debe procurar que todos los recursos de red requeridos por los usuarios sean contenidos dentro de su propio segmento de red. Entre los recursos de red se incluyen: servidores, impresoras, directorios y aplicaciones.

### **La Nueva Regla 20/80**

Las organizaciones están desplazando los servidores de las sucursales hacia centros de datos en las sedes principales por razones de seguridad, costo y administración, lo cual hace que la regla 80/20 se haga obsoleta. Bajo este nuevo escenario todo el tráfico de datos debe atravesar la capa de núcleo del campus, ocasionando un desplazamiento de la regla 80/20 a una regla 20/80. La Figura 3.7 muestra un despliegue de red bajo la regla 20/80. Bajo este escenario 20% del

tráfico de los usuarios permanece local, mientras que el 80% del tráfico cruza los puntos de segmentación de la red para acceder a los servicios.

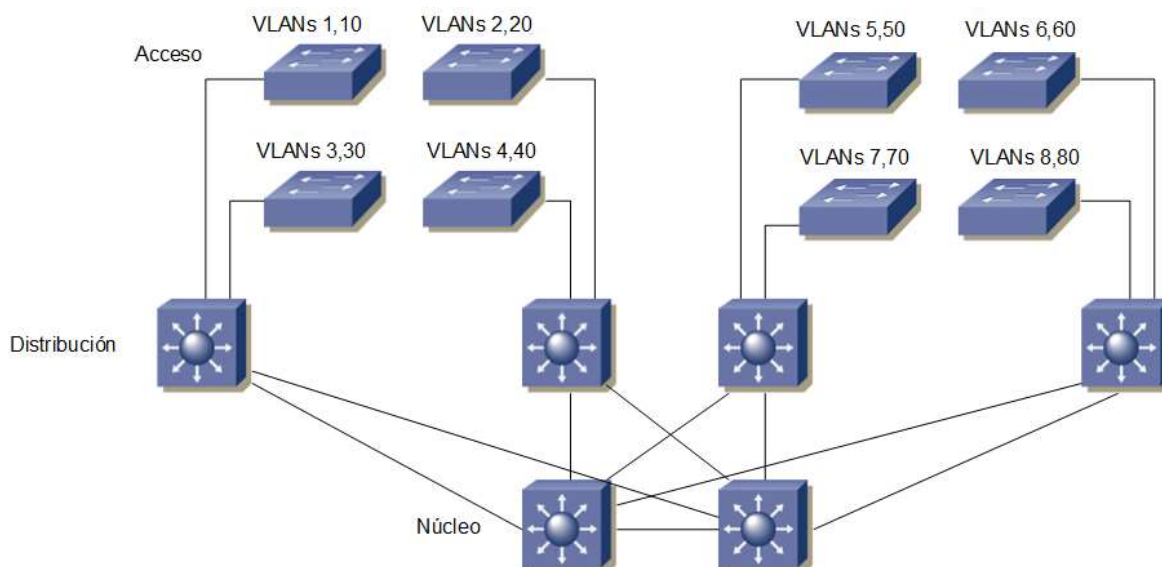


Figura 3.7: Ejemplo de Red Tradicional bajo la Regla 20/80

Con la nueva regla 20/80, la mayoría de los usuarios necesitan cruzar dominios de broadcast, lo cual impone una carga en los routers o switches capa 3 de la red. Al implementar VLANs dentro de éste modelo, se puede reducir el tamaño del dominio de broadcast global de la red en dominios de broadcast más pequeños, disminuyendo el volumen del tráfico de broadcast a nivel general. Las VLANs rompen los dominios de broadcast utilizando un router o switch multicapa.

### 3.1.4 Limitaciones de las Redes de Campus Tradicionales

Los factores que limitan el aprovisionamiento de nuevos servicios y tecnologías en las redes tradicionales se describen a continuación:

#### Complejidad en el Ciclo de Vida de la Red

El ciclo de vida de las redes tradicionales que incluye las fases de planificación, diseño, implementación, operación y optimización es sumamente complejo:

- **Planificación:** Se requiere una inversión de tiempo considerable para planificar nuevos despliegues, actualizaciones u optimizaciones. Los costos de planificación son altos porque se deben considerar muchas ventanas de mantenimiento y se deben considerar muchos dispositivos.
- **Diseño:** Se requiere mucho tiempo para las actividades de diseño de bajo nivel que involucran la elaboración de las plantillas de configuraciones asociadas a las políticas QoS y de seguridad en ambientes de redes heterogéneas con equipos de múltiples fabricantes y múltiples sistemas operativos de red.

- **Implementación:** La implementación de la red es compleja porque se deben configurar todos los dispositivos de la red uno a la vez y se deben resolver aspectos de compatibilidad en ambientes de múltiples fabricantes. También se requiere implementar mecanismos QoS y de optimización para satisfacer la calidad de experiencia de los usuarios fijos y móviles. Adicionalmente se deben implementar mecanismos QoS y de seguridad para incorporar nuevas tecnologías como IoT, Virtualización, BigData y Nube.
- **Operación:** El mantenimiento de la red es complejo porque se deben administrar equipos de diferentes fabricantes y se deben gestionar equipos con múltiples versiones de un mismo sistema operativo. La incorporación de nuevos servicios o aplicaciones es un proceso lento y complejo porque se deben configurar de manera manual todos los dispositivos de la red involucrados en los nuevos requerimientos de servicios.
- **Optimización:** La optimización de la red es compleja porque se deben configurar todos los dispositivos de red involucrados en la mejora de un servicio.

### **Políticas Inconsistentes**

La implementación de políticas en la red es un proceso largo y complejo en el cual los administradores de la red deben configurar de manera manual cientos de dispositivos y mecanismos, lo cual es propenso a cometer errores y dificulta la implementación de políticas consistentes en la red.

### **Limitación para escalar**

Las redes tradicionales son difíciles de escalar por contar con tráfico dinámico e impredecible [1] y por carecer de analíticas de red que permitan estimar y planificar crecimiento de tráfico futuro.

### **Dependencia del fabricante**

Existe un grado de dependencia muy alto con los fabricantes de los equipos de red. Existe mucho hardware y software propietario y cerrado que imposibilita el desarrollo de nuevos protocolos, funcionalidades y servicios.

## 4. Redes SDN

La última década ha sido testigo de una extraordinaria revolución en los dispositivos de usuarios finales tanto a nivel de la capacidad de cómputo como a nivel de inteligencia. La presencia de un gran número de dispositivos inteligentes, tabletas, teléfonos y servidores de alto desempeño, son una muestra evidente del poder de cómputo que se ha brindado a los usuarios en sus equipos terminales. La infraestructura de red de hoy en día transporta mucho más datos sobre redes más grandes y complejas que conectan personas, aplicaciones y cosas, lo cual demanda mejoras en la capacidad de procesamiento del hardware de la red y un mejor control sobre el tráfico de datos. Los procesadores existentes no pueden ser reemplazados con procesadores más rápidos debido al fuerte lazo que existe entre la arquitectura de los procesadores y el software que opera sobre estos. Reemplazar el software tampoco es una opción viable ya que las redes están conformadas por equipos de diferentes fabricantes que deben interoperar y se requieren grandes esfuerzos para lograr esta compatibilidad. Esta condición limita la innovación en el plano de control de la red y se plantean las siguientes interrogantes: ¿Se pueden mover los algoritmos y la computación del hardware de los equipos de red?, ¿se puede crear una red más abierta, flexible e inteligente?, ¿se puede mover el control de la red a una entidad central que se pueda programar? Si se puede responder afirmativamente a estas interrogantes, se puede decir que estamos en la presencia de un desplazamiento del modelo de red tradicional a un modelo de redes definidas por software, donde el plano de control de los equipos de la red es movido a un punto lógico central capaz de controlar de manera programática el reenvío de tráfico de muchos dispositivos [18].

### 4.1 Arquitectura General de SDN

SDN [19] es un nuevo enfoque en la programación de la red que consiste en la capacidad de inicializar, controlar, cambiar y gestionar el comportamiento de reenvío del tráfico de una red mediante APIs abiertas. En una red SDN se separan los planos de control y datos de los dispositivos de red y se desplaza el plano de control a una unidad central de controladores SDN. Un controlador SDN se encarga de definir y comunicar las reglas de reenvío de tráfico a los dispositivos SDN y abstraer la infraestructura de red y su topología a las aplicaciones. Bajo este modelo las aplicaciones consideran a la red como un solo switch lógico central que provee servicios de conectividad a los usuarios y a las aplicaciones. Al separar los planos de datos y control, los switches de la red se convierten en dispositivos de reenvío simples y la lógica de control se implementa en un sistema operativo de red centralizado. Con este nuevo enfoque se obtienen grandes beneficios: Se simplifica el aprovisionamiento de políticas, se agiliza la reconfiguración de servicios y se acelera la innovación de las redes [20].

Una red SDN está conformada generalmente por tres grandes capas: (1) Capa de Aplicación (Application Layer), (2) Capa del Plano de Control (Control-Plane Layer) y (3) Capa del Plano de Datos (Data-Plane Layer) (ver Figura 4.1 tomada de [21]).

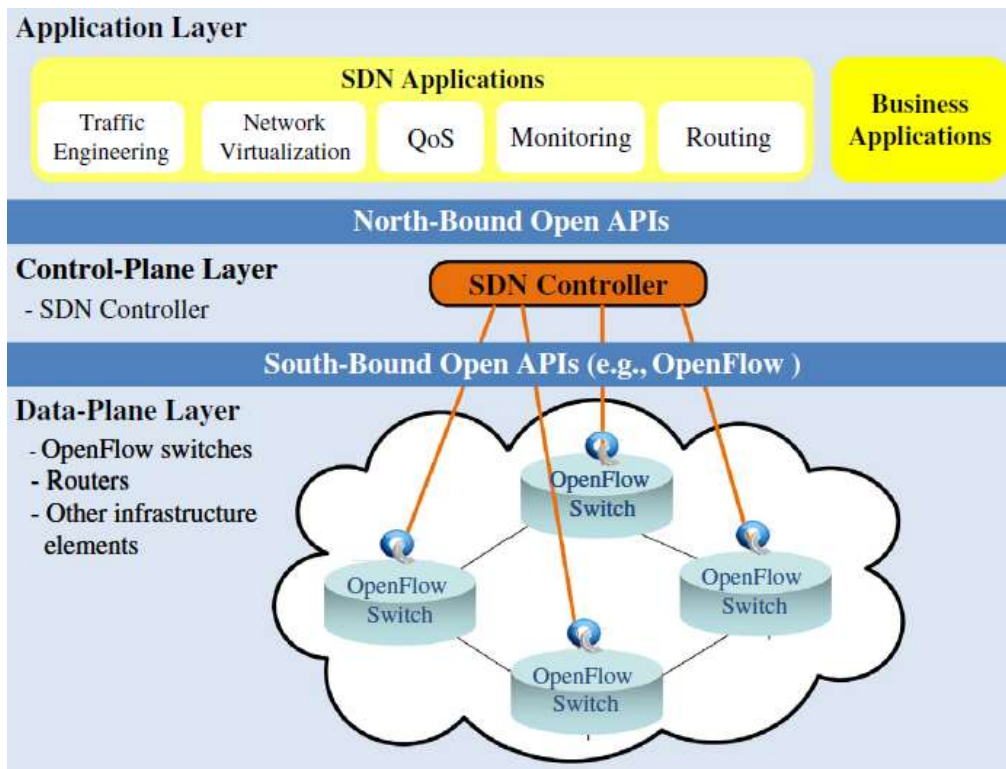


Figura 4.1: Representación Lógica de una Arquitectura de Red SDN

**Capa de Aplicación:** Está conformada por las aplicaciones del negocio y por las aplicaciones de servicios de red.

**Capa del Plano de Control:** Contiene los controladores SDN encargados de gobernar y dirigir la manera en que se transportan los datos en los dispositivos SDN.

**Capa del Plano de Datos:** Está conformada por dispositivos SDN físicos y virtuales encargados de transportar datos en base a instrucciones recibidas por los controladores SDN de la red.

En una red SDN los controladores se comunican con las aplicaciones externas mediante APIs Northbound abiertas y con los dispositivos SDN mediante APIs Southbound abiertas.

#### 4.1.1 Pilares Fundamentales de una Arquitectura de Red SDN

Una arquitectura de red SDN sigue 4 pilares fundamentales [22]:

1. **Los planos de control y datos son desacoplados.** La funcionalidad de control es removida del hardware de los dispositivos de red, quienes ahora juegan un rol más simple de transportar datos en la red.
2. **Las decisiones de reenvío de tráfico están basadas en flujos.** Se establece la definición de flujos, como instancias de campos de cabeceras de paquetes que actúan como un criterio (*filtro*) y un conjunto de acciones

(*instrucciones*) a ejecutar sobre el tráfico de datos de la red. En el contexto de SDN OpenFlow un flujo es una secuencia de paquetes entre un origen y un destino que comparten características comunes y reciben el mismo tratamiento de políticas en los dispositivos de red. La abstracción de flujos permite unificar el comportamiento de diferentes dispositivos de red, incluyendo switches, routers, firewalls, optimizadores de tráfico y balanceadores de carga.

3. **La lógica de control se mueve a una entidad lógica central de controladores SDN** o sistemas operativos de red NOS (Network Operating System). Un NOS es una plataforma de software que corre sobre un servidor general que provee los recursos y las abstracciones esenciales que facilitan la programación de los dispositivos SDN.
4. **La red puede ser programada** por aplicaciones de software que corren en el tope de los controladores de la red.

#### 4.1.2 Operación de SDN

En una red SDN un plano de control centralizado ejecuta todas las funciones complejas de enrutamiento, manejo de políticas y chequeos de seguridad de la red, define las reglas de reenvío y comunica las reglas a los dispositivos SDN a través de una API Southbound. En la Figura 4.2 se muestra una topología de red SDN OpenFlow conformada por un controlador SDN y siete switches operados mediante una API OpenFlow.

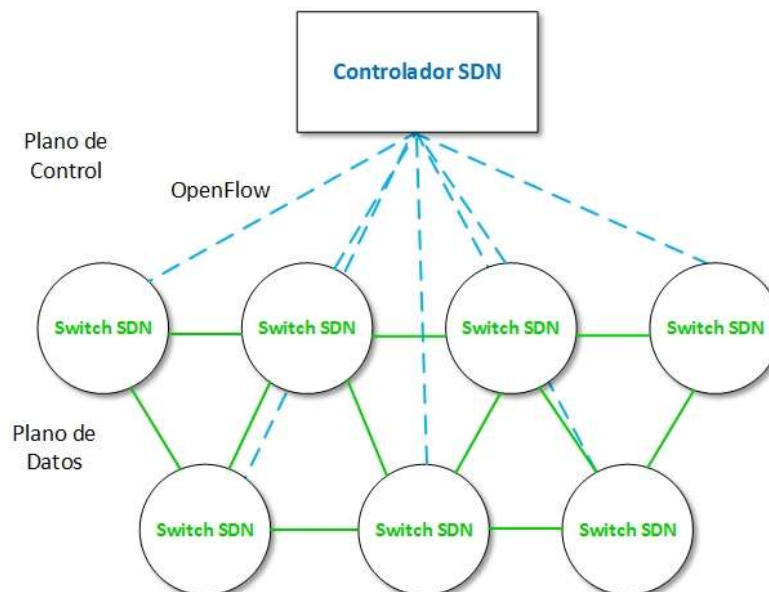


Figura 4.2: Topología SDN OpenFlow de Siete Switches SDN

El plano de control SDN se utiliza para el intercambio de información de control entre el controlador y los dispositivos SDN y el plano de datos se utiliza para el transportar los datos en la red. El controlador SDN OpenFlow define las reglas de reenvío de datos y propaga esas reglas a los dispositivos bajo su dominio en

forma de entradas de flujos que son almacenadas en las tablas de flujos del plano de datos de los switches SDN.

## 4.2 Arquitectura de Capas SDN

SDN puede ser representada de manera abstracta por una composición de planos y capas [19] conectados mediante interfaces (ver Figura 4.3 tomada de [19]).

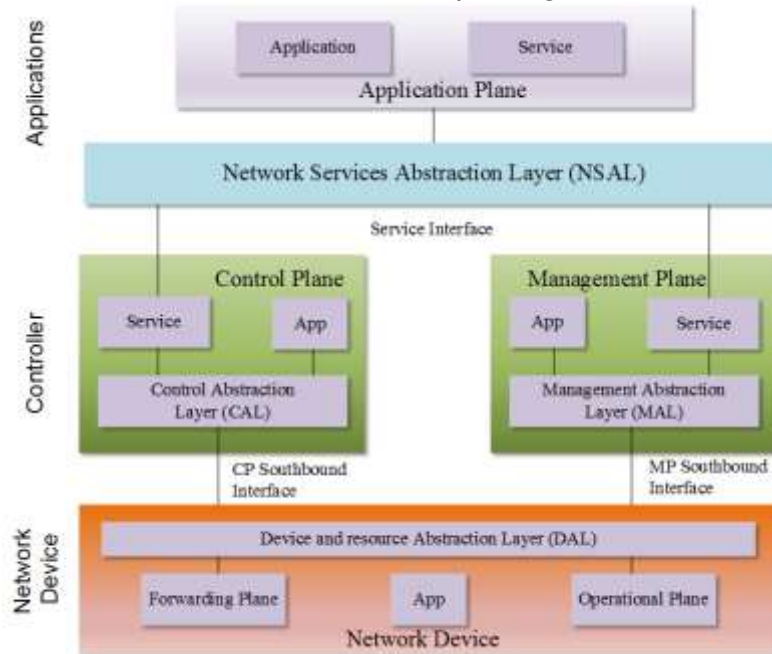


Figura 4.3: Arquitectura de Capas SDN

Los planos se comunican entre sí mediante protocolos cuando se encuentran distribuidos en varios dispositivos y se comunican mediante APIs internas dentro del mismo dispositivo. Una red SDN también puede ser vista como una composición de capas y sistemas (ver Figura 4.4 tomada de [22]).

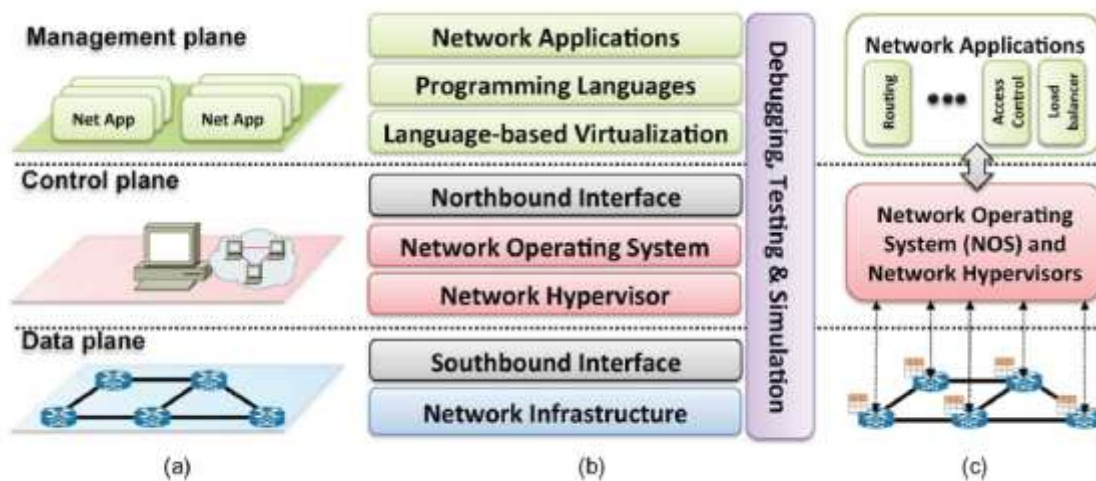


Figura 4.4: Arquitectura SDN basada en (a) Planos, (b) Capas y (c) Sistemas

Algunas capas se encuentran presentes en todos los despliegues SDN incluyendo: infraestructura de red, APIs Southbound y controladores, mientras que otras son opcionales incluyendo APIs Northbound, aplicaciones, hipervisores y virtualización de redes. Comenzando de abajo hacia arriba, se identifican los siguientes sistemas, planos y capas:

#### **4.2.1 Dispositivo SDN**

Los dispositivos SDN son elementos físicos o virtuales que manipulan y reenvían paquetes en base a reglas definidas por un controlador. Estos contienen un plano de datos (Forwarding Plane) encargado de transportar y manipular campos de cabecera de paquetes y un plano operacional (Operational Plane) encargado de efectuar tareas administrativas relacionadas con su funcionamiento. La comunicación de estos planos con el controlador se realiza a través de la capa de abstracción del dispositivo (Device and Resource Abstraction Layer) y a través de las interfaces de los planos de control y de gestión respectivamente. Los dispositivos SDN pueden ser switches, routers o elementos de reenvío físicos o virtuales que soportan planos de datos y APIs *Southbound* SDN, como OpenFlow [2] (ver **¡Error! No se encuentra el origen de la referencia.** tomada de [22]).

##### **Plano de Datos (Forwarding Plane)**

El plano de datos es el camino físico que toman los paquetes cuando viajan en la red. El plano de datos se encuentra en los dispositivos de red y se encarga de la manipulación y del reenvío de los paquetes. El plano de datos incluye, pero no está limitado a filtros, medidores, marcadores, y clasificadores.

##### **Plano Operacional (Operational Plane)**

El plano operacional se encarga de gestionar y notificar el estado de los dispositivos de red incluyendo: estado del dispositivo, número de puertos disponibles, estado de los puertos y memoria disponible entre otros. El plano operacional pudiera gestionar los siguientes recursos: memoria, CPU, puertos, interfaces y colas y constituye generalmente el punto de terminación para las aplicaciones y servicios del plano de gestión.

#### **4.2.2 Controlador SDN**

Un controlador SDN o NOS (Network Operating System) es un software que corre en un servidor de red para gestionar las reglas de reenvío de tráfico de los dispositivos SDN. Un controlador SDN provee abstracciones, servicios esenciales y APIs comunes a las aplicaciones de red. Entre los principales servicios ofrecidos se encuentran: (1) Topología de la red, (2) notificación del estado de la red, (3) descubrimiento de dispositivos y (4) distribución de la configuración de la red.

## Componentes de los Controladores SDN

Los controladores SDN cuentan en su núcleo con un conjunto de servicios y un conjunto de APIs e interfaces que permiten la comunicación con el resto de los componentes de la arquitectura (ver Figura 4.5 tomada de [22]).

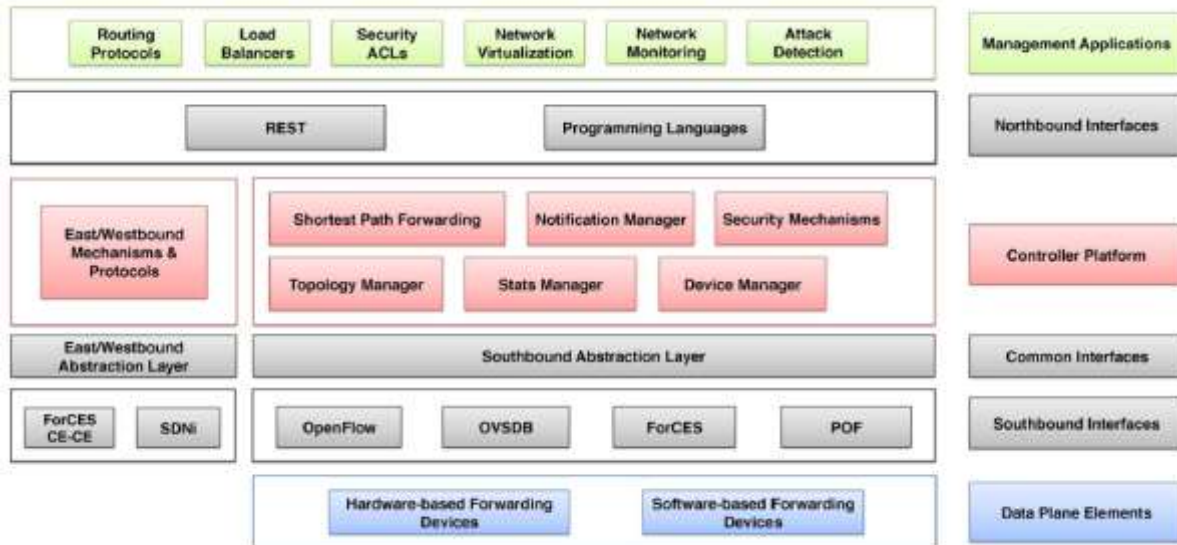


Figura 4.5: Componentes de un Controlador SDN

Existen tres tipos de interfaces en un controlador SDN: (1) Interfaces/APIs Northbound, (2) interfaces/APIs Southbound e (3) interfaces/APIs este/oeste (east/west). Las APIs Northbound permiten la comunicación entre las aplicaciones y los controladores SDN, las APIs Southbound permiten la comunicación entre el controlador y los dispositivos SDN y las APIs este/oeste permiten la comunicación entre controladores SDN.

### APIs Northbound

Las APIs Northbound presentan una interfaz común para el desarrollo de aplicaciones. Tradicionalmente, una API Northbound abstrae el conjunto de instrucciones de bajo nivel utilizado por las APIs Southbound para programar los dispositivos SDN. Actualmente los controladores de red ofrecen una variedad de APIs Northbound, como APIs RESTful [23][24], sistemas de archivos, lenguajes de programación como Python y Java (ver Figura 4.6) y lenguajes de programación específicos para SDN, como ProCera [20], FML [25] (Flow Based Management Language), Frenetic [26], NetCore [27] y Pyretic [28].

Un controlador SDN puede ofrecer APIs de bajo nivel y APIs de alto nivel. Las APIs de bajo nivel proveen acceso a los dispositivos individuales a través de primitivas de bajo nivel de una manera común y consistente. Las APIs de alto nivel presentan a los desarrolladores vistas abstractas de la red que ocultan los detalles del funcionamiento interno de los switches individuales y ofrecen servicios de conectividad y servicios de red. Las aplicaciones pueden afectar la operación de la red enviando métodos al controlador a través de APIs Northbound en respuesta a un evento o actuando de manera involuntaria.

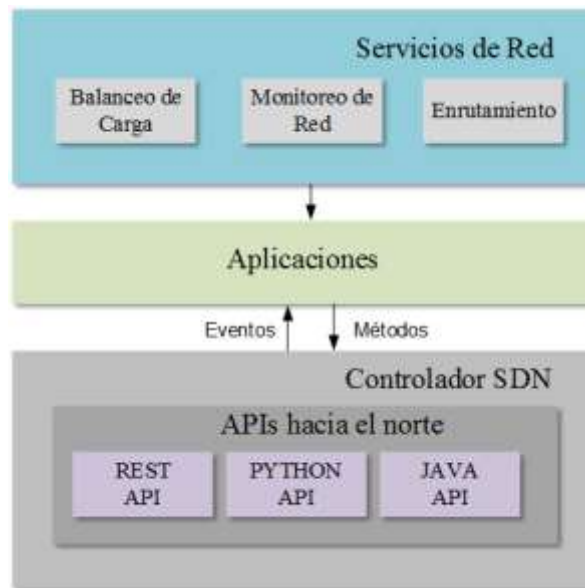


Figura 4.6: APIs hacia el Norte de un Controlador SDN

### APIs Southbounds

Las APIs Southbound constituyen el protocolo de comunicación que facilita la comunicación entre controladores y dispositivos SDN. Estas APIs pueden ser abiertas o propietarias<sup>10</sup>. El controlador SDN cuenta con las siguientes opciones de APIs Southbound abiertas: OpenFlow [2], OVSDDB [29] (Open vSwitch Database) y ForCES [7]. También existen las siguientes opciones de conectores: BGP [30] (Border Gateway Protocol), SNMP [3] y NETCONF [31] (Network Configuration Protocol) entre otros. OpenFlow es el API Southbound estándar de la industria de las redes para entornos SDN.

### APIs Este/Oeste

Las APIs este/oeste se utilizan en esquemas de controladores SDN distribuidos para permitir la comunicación entre los controladores y compartir información de alcanzabilidad y control (ver Figura 4.7 tomada de [22]). Entre las funciones de estas interfaces se encuentran: (1) importar o exportar datos entre controladores, (2) proveer algoritmos para modelos de consistencia de datos y (3) proveer capacidades de monitoreo y notificaciones. Para que exista compatibilidad e interoperabilidad entre diferentes controladores, se requiere disponer de APIs este/oeste estandarizadas, como SDNi [32]. A través de SDNi se definen requerimientos comunes para coordinar el establecimiento de flujos y el intercambio de información de alcanzabilidad a través de varios dominios, que permiten crear plataformas de control SDN escalables y confiables.

### Controladores SDN Centralizados vs Distribuidos

En la actualidad existen dos tipos de diseño de controladores SDN: (1) controladores SDN centralizados y (2) controladores SDN distribuidos. En un diseño de controlador SDN centralizado, una sola entidad de control gestiona

<sup>10</sup> <https://www.sdxcentral.com/resources/sdn/southbound-interface-api>

todos los dispositivos SDN. La limitante de este diseño es la condición de un solo punto de falla, en la cual al ocurrir una falla en el controlador o en alguno de sus enlaces, la red continúa operando de manera limitada imposibilitando la incorporación de cambios y nuevos servicios.

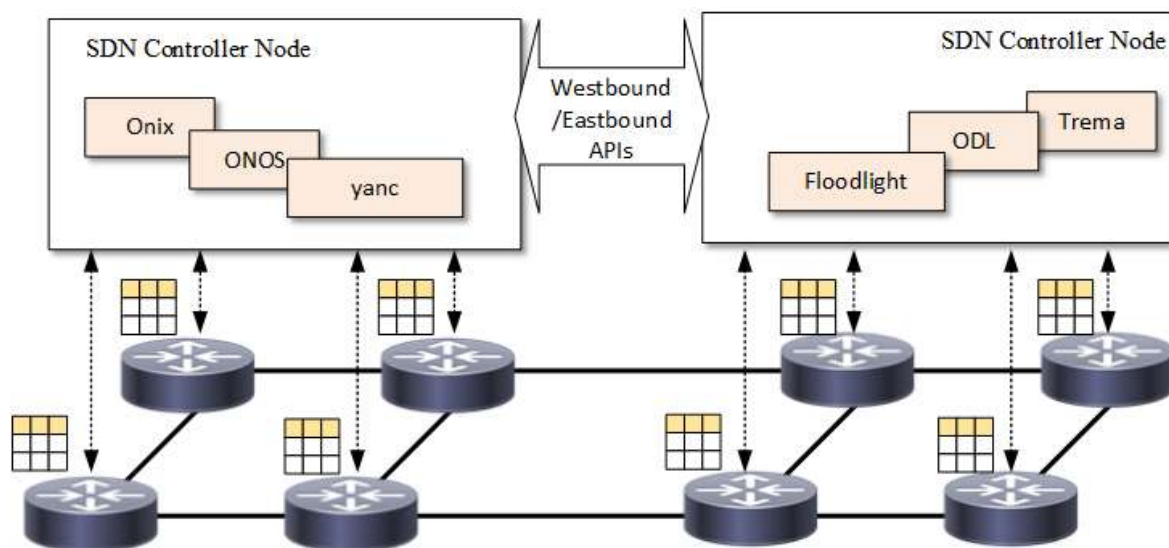


Figura 4.7: Comunicación entre Controladores vía APIs Este/Oeste

En un diseño de controladores SDN distribuidos, el control de la red se encuentra distribuido en varios controladores, ofreciendo mayor disponibilidad de la red en caso de fallas y posibilitando la escalabilidad de la red mediante la incorporación de controladores adicionales a medida que sea necesario. Un esquema de controlador SDN distribuido puede ser: (1) un clúster de nodos centralizados en un mismo lugar o (2) un grupo de controladores distribuidos geográficamente. La primera opción puede ofrecer un alto rendimiento en centros de datos muy densos, y la segunda opción puede ofrecer mayor resiliencia a condiciones de fallas físicas o lógicas. Un Proveedor de Servicios de Comunicación de nube conformado por múltiples centros de datos interconectados mediante una WAN puede optar por un diseño de controladores híbrido, con clústeres de controladores en cada centro de datos y controladores distribuidos en diferentes localidades.

### Funciones de Servicio de Red

El núcleo de un controlador SDN está conformado por los siguientes mecanismos y funciones: (1) gestión de la topología, encargado de descubrir la topología de la red, (2) gestión de estadísticas, encargado de recopilar contadores del tráfico de la red, (3) gestión de notificaciones, encargado de gestionar la comunicación del plano de control con los elementos de la red, (4) gestión de dispositivos, encargado de configurar y gestionar los elementos de la infraestructura de red, (5) reenvío de caminos más cortos, encargado de seleccionar los mejores caminos hacia los destinos y (6) mecanismos de seguridad, encargado de proveer mecanismos de protección a la red.

## **Plano de Control**

El plano de control es el lugar donde se toman las decisiones de como enviar los paquetes. Es responsable de comunicar el tratamiento del tráfico a los dispositivos de red. El rol principal del plano de control es definir las reglas de comportamiento que forman parte de las tablas de flujos de los dispositivos de red en base a la topología de la red y a las solicitudes de servicios externas. El plano de control puede interactuar con el plano operacional para determinar el estado de los puertos o las capacidades de un dispositivo para tomar decisiones.

Entre las funciones más importantes del plano de control se encuentran:

- Descubrir y mantener la topología de red.
- Seleccionar e instanciar rutas de paquetes.
- Proveer mecanismos de recuperación de caminos.

El controlador conecta la capa de abstracción de control (Control Abstraction Layer) con la capa de abstracción de recursos y dispositivos mediante una interfaz del plano de control CPSI (Control Plane Southbound Interface) para comunicar las reglas de reenvío de tráfico a los dispositivos de red.

El plano de control pudiera ser distribuido en un arreglo de controladores SDN. En este caso la comunicación entre controladores se logra a través de una interfaz este-oeste que soporte protocolos de red como BGP [30], PCEP [33] (Path Computational Element Communication Protocol), RCP [10] (Routing Control Platform) y SoftRouter [34].

## **Hipervisor de Red**

En un ambiente de virtualización de servidores, un hipervisor es una plataforma de software que permite correr varias VMs sobre un mismo dispositivo de cómputo. Un hipervisor cuenta con switches virtuales vSwitches que permiten la comunicación entre VMs. En una red SDN basada en vSwitches un controlador le comunica a los vSwitches las reglas de reenvío de tráfico asociadas a la comunicación entre las VMs. Los vSwitches disponen de mecanismos de túneles para comunicarse con VMs hospedadas en otros hipervisores externos.

### **4.2.3 Plano de Gestión**

El plano de gestión es responsable de las funciones de monitoreo, configuración y mantenimiento de los dispositivos de red. El plano de gestión conecta la capa de abstracción de gestión (Management Abstraction Layer) con la capa de abstracción de recursos y dispositivos mediante la interfaz del plano de gestión MPSI (Management-Plane Southbound Interface) para asegurarse que la red opere de manera óptima.

Entre las funcionalidades FCAPS [35] (Fault, Configuration, Accounting, Performance, Security) del plano de gestión se encuentran:

- **Gestión de Fallas:** (1) Mantiene y examina registros de error, (2) acepta y actúa bajo notificaciones de detección de errores, (3) traza e identifica fallas, (4) ejecuta pruebas de diagnóstico y (5) corrige fallas.
- **Gestión de Auditorías:** Notifica a los usuarios sobre los costos incurridos por la utilización de recursos asociados a un servicio de red.
- **Gestión de Configuración:** Identifica, recopila información y cambia la configuración de dispositivos de red.
- **Gestión de Desempeño:** (1) Recopila información de estadísticas de desempeño, (2) mantiene y examina registros de históricos de estado y (3) determina el desempeño de la red bajo condiciones naturales y artificiales.
- **Gestión de Seguridad:** (1) Crea, elimina y controla mecanismos y servicios de seguridad, (2) distribuye información relevante a la seguridad y (3) reporta eventos de seguridad en la red.

#### 4.2.4 Arquitectura de Gestión Basada en Modelos

En SDN la gestión de la red se implementa mediante una jerarquía de capas de Protocolos, Modelos y Datos (ver Figura 4.8 tomada de [36]).

#### Arquitectura de Gestión Basada en Modelos

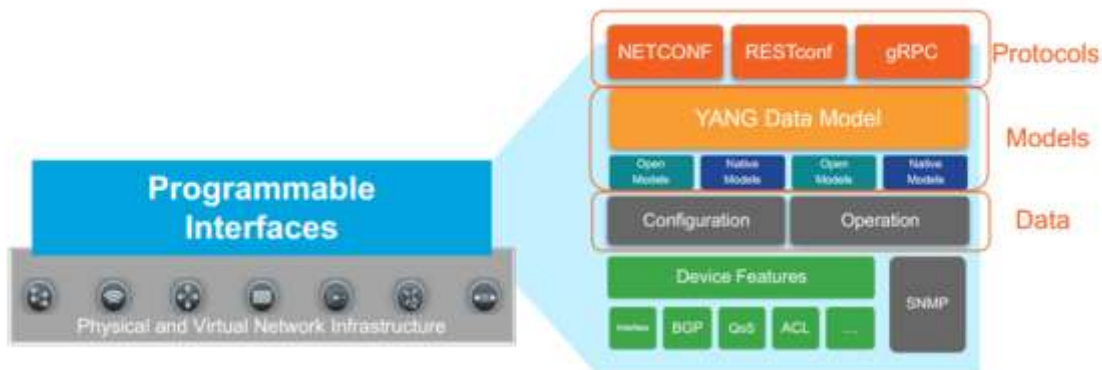


Figura 4.8: Arquitectura de Gestión Basada en Modelos

Los protocolos de gestión utilizan un lenguaje de modelo de datos como YANG [37] (Yet Another Next Generation) para configurar y obtener datos del estado de los dispositivos. Un lenguaje de modelado de datos [41] es un conjunto de herramientas y técnicas responsables para diseñar datos y estructuras estáticas. Un modelo de datos describe como los datos son representados y accedados. En una arquitectura SDN se utiliza el lenguaje de modelo de datos YANG para modelar configuraciones, datos y RPCs.

#### 4.2.5 Protocolos de Gestión de Red

Entre los protocolos de gestión de red para ambientes SDN se encuentran NETCONF [31], RESTCONF [38] y gRPC [39] (ver Tabla 4.1).

| <b>FUNCIONALIDAD</b>  | <b>NETCONF</b> | <b>RESTCONF</b> | <b>GRPC</b>                                                         |
|-----------------------|----------------|-----------------|---------------------------------------------------------------------|
| Estándar              | IETF RFC 6241  | IETF RFC 8040   | Código Abierto                                                      |
| Optimizado para redes | •              | •               | <b>X</b><br>Diseñado para comunicaciones cliente/servidor genéricas |
| Codificación          | XML            | XML/JSON        | Proto Bufer                                                         |
| Orientado a conexión  | •              | <b>X</b>        | •                                                                   |
| Transporte            | SSH            | HTTP/HTTPS      | HTTP/2                                                              |

**Tabla 4.1: Protocolos de Gestión SDN**

NETCONF es un protocolo de configuración de red estandarizado por el IETF en el RFC 6241 [31] que provee mecanismos simples para instalar, manipular, y eliminar la configuración de dispositivos de red. Sigue un paradigma solicitud-respuesta RPC (Remote Procedure Call) con codificación XML y transporte seguro SSH (Secure Shell). NETCONF define almacenes de datos que contienen la configuración de los dispositivos de red y operaciones CRUD (Create, Read, Update, Delete) que permiten recuperar, configurar, copiar y eliminar almacenes de datos.

RESTCONF es un protocolo de configuración de red basado en HTTP estandarizado por el IETF en el RFC 8040 [38]. A través de RESTCONF las aplicaciones Web pueden acceder y modificar la configuración de un dispositivo de red siguiendo una arquitectura cliente/servidor basada en RPC con clientes y servidores RESTCONF.

gRPC [39] es un framework para llamadas a procedimientos remotos RPC de código abierto y de alto desempeño liberado por la empresa Google para construir sistemas distribuidos masivos.

OF-CONFIG [40] es un protocolo de gestión desarrollado por la ONF (Open Networking Foundation) para gestionar switches OpenFlow físicos y virtuales. Utiliza NETCONF para la administración, XML para la codificación y SSH para el transporte.

#### **4.2.6 Plano de Aplicaciones**

El plano de aplicaciones está conformado por las aplicaciones y los servicios que corren encima de los controladores SDN.

#### **4.2.7 Interfaces**

Las interfaces son conexiones funcionales que permiten la comunicación entre los planos de la arquitectura SDN. Se encuentran tres tipos de interfaces principales: (1) interfaces de servicios para la comunicación entre el plano de aplicaciones y el

controlador, (2) interfaces del plano de control y (3) interfaces del plano de gestión. Entre las principales interfaces de servicios se encuentran las interfaces RESTful [23] y las interfaces RPC. La interfaz más utilizada para el plano de control es la API del protocolo estándar OpenFlow y las interfaces del plano de gestión más comunes son NETCONF y RESTCONF.

#### 4.2.8 Capas de Abstracción Adicionales

- **DAL (Device and Resource Abstraction Layer):** Abstrae y expone los recursos de los planos de datos (*medidores, colas, clasificadores*) y los recursos de los planos operacionales (*memoria, cpu, interfaces, puertos y colas*) a los planos de gestión y control.
- **CAL (Control Abstraction Layer):** Constituye la capa de abstracción del plano de control.
- **MAL (Management Abstraction Layer):** Representa la capa de abstracción de gestión.
- **NSAL (Network Services Abstraction Layer):** Provee abstracciones de servicios que pueden ser utilizadas por aplicaciones u otros servicios y comunica el plano de aplicación con los planos de gestión y control de un controlador por medio de interfaces de servicios.

#### 4.2.9 Virtualización de Red (Slicing)

El objetivo de la virtualización de red se centra en la capacidad de compartir el plano de datos del hardware de red a múltiples redes lógicas, cada una con su propio direccionamiento y su propio mecanismo de reenvío. En una red SDN se puede virtualizar la capa de hardware de red en *slices* y asignar a cada *slice* recursos y espacios de direcciones. Un *slice*<sup>11</sup> se define como una instancia de una red virtual, y dos redes virtuales distintas sobre el mismo hardware físico se conocen como *slices*. Virtualizar la red trae consigo los siguientes beneficios: (1) mejora la utilización de los recursos de la red, (2) permite verificar la disponibilidad de recursos antes de efectuar cambios en la red y (3) permite compartir el mismo hardware de red de una manera aislada y controlada. Existen varios esquemas de virtualización de redes SDN. La virtualización de una red SDN se puede implementar mediante un proxy o siguiendo un esquema de virtualización basado en lenguajes.

##### Virtualización de redes SDN mediante Proxies

Bajo un esquema de proxy se coloca un controlador SDN especial entre los controladores y los dispositivos SDN de la red. El proxy actúa como un multiplexor de tráfico y como un gestor de la asignación de recursos a *slices* independientes. El proxy intercepta todos los mensajes entre los dispositivos y los controladores SDN para garantizar que el tráfico de un *slice* no interfiera con el tráfico de otro *slice*. En la actualidad existen diferentes esquemas de implementación de *slicing* basados en proxies para entornos SDN: FlowVisor [42], OpenVirteX [43], y

---

<sup>11</sup> <http://www.geni.net>.

AutoSlice [44], el cual propone múltiples proxies para proveer una virtualización robusta del espacio de flujos.

**FlowVisor**<sup>12</sup> [42] es un controlador OpenFlow de propósito especial que actúa como un proxy transparente entre switches OpenFlow y controladores OpenFlow, permitiendo crear *slices* aislados que pueden ser asignadas a controladores particulares, ofreciendo una capa de virtualización de red basada en OpenFlow. Los slices pueden ser definidos por una combinación de puertos de switches Capa 1, direcciones Ethernet origen/destino Capa 2, direcciones IP origen/destino o tipo Capa 3, puertos TCP/UDP origen/destino o códigos/tipos ICMP Capa 4.

En la Figura 4.9 se muestra una topología de red conformada por tres controladores OpenFlow, un controlador FlowVisor y seis switches OpenFlow.

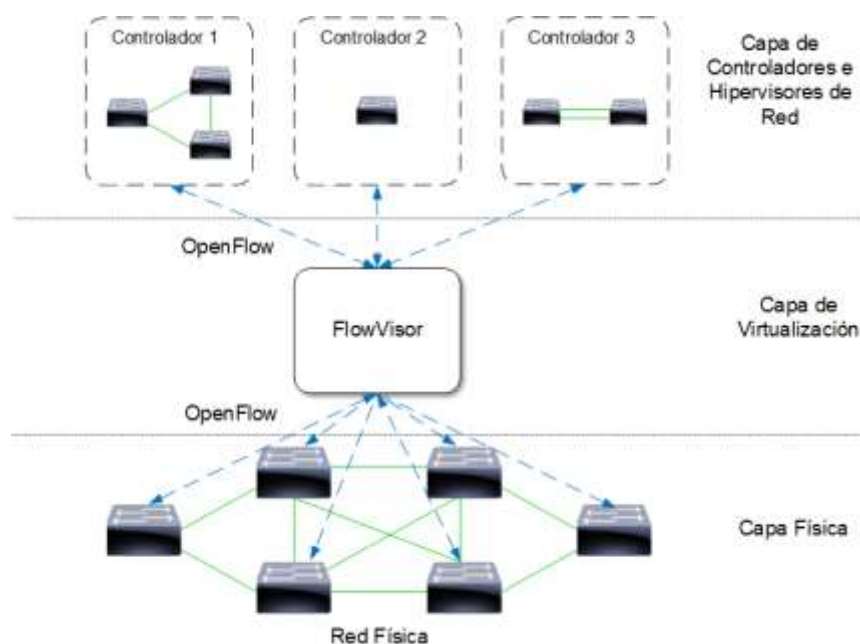


Figura 4.9: Topología de Red OpenFlow con FlowVisor

En este caso FlowVisor recibe todos los comandos OpenFlow de los controladores hacia los switches OpenFlow y las respuestas y notificaciones de estadísticas de los switches hacia los controladores respectivos garantizando el aislamiento de tráfico entre cada *slice* definido. En el ejemplo se puede observar como cada controlador tiene una vista particular y diferente de la misma red física.

FlowVisor opera como un multiplexor y conoce la subred OpenFlow a la cual pertenecen los paquetes que arriban a los switches OpenFlow. Cuando un paquete nuevo llega a un switch, el switch lo envía a los controladores SDN donde es interceptado por el módulo FlowVisor, el cual verifica a que controlador pertenece y envía la solicitud al controlador OpenFlow correspondiente. El

<sup>12</sup> <http://onlab.us/flowvisor.html#what>

controlador OpenFlow responde con la regla de aplicación al paquete y FlowVisor ejecuta la política al paquete permitiéndole el tránsito a su red destino.

A través de FlowVisor un grupo de investigadores puede crear sus propias instancias de red lógicas corriendo sus propios protocolos de enrutamiento y ejecutarlas sobre una red real en paralelo con una red de producción y mantener el aislamiento y las velocidades de reenvío del hardware existente.

FlowVisor maneja cinco dimensiones de *slicing*: (1) ancho de banda, (2) topología, (3) tráfico, (4) CPU de dispositivos, y (5) tablas de reenvío.

- Ancho de Banda: Permite asignar a cada *slice* su propia fracción de ancho de banda de un enlace.
- Topología: Cada *slice* puede tener su propia vista de la red con sus nodos y enlaces y experimentar condiciones particulares de la red, como bucles y fallas de enlaces.
- Tráfico: Se puede asociar un conjunto específico de tráfico a un *slice*. Por ejemplo, todos los paquetes hacia o desde un conjunto de direcciones, o todo el tráfico http.
- CPU de Dispositivos: Se pueden aislar los recursos de cómputo del hardware de la red, como el CPU y la memoria de los switches y routers.
- Tablas de reenvío: Se pueden aislar las entradas de las tablas de reenvío de flujos entre los slices para que no interfieran entre sí.

### **Virtualización basada en Lenguajes**

La virtualización mediante lenguajes utiliza lenguajes de virtualización de propósito especial como Pyretic [28] o Splendid [45]. A diferencia de otros mecanismos de bajo nivel como la segmentación por VLANs, el filtrado de tráfico a través de firewalls, o los aislamientos de control por medio de hipervisores, que interceptan y analizan cada evento y mensaje de control en tiempo real, en una abstracción a nivel de lenguaje, el compilador solo necesita ejecutarse una vez, antes de que el programa sea desplegado en la red, lo cual acelera el plano de datos y reduce la latencia.

#### **4.2.10 Lenguajes de Programación SDN**

Un framework de programación SDN consiste generalmente de un lenguaje de programación y las herramientas necesarias para compilar y validar las reglas OpenFlow generadas por los programas de aplicaciones.

Entre los principales lenguajes de programación SDN se encuentran Frenetic [26], FML [25] (Flow-Based Management Language), Procera [20], NetCore [27] (Network Core Programming Language), Nettle [46] y Pyretic [28].

#### **4.2.11 Aplicaciones**

En una arquitectura SDN las aplicaciones interactúan con la plataforma de controladores SDN a través de APIs Northbound que solicitan el estado de la red y manipulan los servicios provistos por la misma. Las APIs Northbound permiten

configurar funciones de red, incluyendo: (1) aprendizaje del switch, (2) cálculo de caminos, (3) enrutamiento, (4) monitoreo de tráfico, (5) control de acceso a la red, y (6) balanceo de carga hacia servidores, entre otras.

A través de una API, las aplicaciones son capaces de efectuar distintas tareas como seleccionar la mejor ruta para un conjunto de paquetes, balancear el tráfico a varios destinos, agregar una nueva ruta o un nuevo dispositivo, redirigir el tráfico a dispositivos especializados como firewalls, autenticadores o analizadores de tráfico. Las aplicaciones pueden interactuar con el controlador una vez que este haya inicializado los dispositivos SDN y reportado la topología de red. Las aplicaciones responden a eventos provenientes del controlador o a solicitudes externas de otros sistemas, como monitores de tráfico, servicios IDS (Intrusion Detection System), y peers BGP afectando el comportamiento de la red.

### **4.3 Comparativa entre Arquitectura de Red Tradicional y SDN**

En la siguiente sección se pone en perspectiva el análisis funcional de una Arquitectura de Red Tradicional y una Arquitectura de Red SDN.

#### **4.3.1 Análisis Funcional de una Arquitectura de Red Tradicional**

En la actualidad una Arquitectura de Red Tradicional se ha caracterizado por operar bajo un esquema de control distribuido en el cual los dispositivos de red aprenden sobre las capacidades, los enlaces y las rutas alcanzables por los dispositivos vecinos y cooperan entre sí a través de protocolos de conmutación y de enrutamiento encargados de mantener el estado de la red en las tablas de reenvío de cada dispositivo de red. Para mantener consistencia y coherencia con las políticas de red de una organización, el grupo de administración IT debe configurar de manera periódica parámetros como reglas ACLs, VLANs de acceso, filtrado de tráfico, métricas de enrutamiento y políticas QoS entre otros.

La infraestructura de red tradicional está conformada por dispositivos de comunicación, como routers y switches, que efectúan funciones de software y hardware en el mismo equipo. Las funciones de software como el intercambio de conocimiento de rutas, el cálculo de las métricas y la aplicación de políticas se ejecutan en el plano de control y las funciones de transporte de los datos se ejecutan en el plano de datos. En la Figura 4.10 se puede apreciar esta separación de funciones en un router o switch tradicional. Los routers y switches tradicionales efectúan las siguientes funciones:

1. Intercambiar información con otros routers y switches.
2. Notificar capacidades y conexiones.
3. Aprender las capacidades y conexiones de otros routers y switches.
4. Calcular los mejores caminos basado en la información aprendida.
5. Definir ACLs.
6. Transportar los datos basado en los caminos calculados.
7. Aplicar las ACLs al tráfico entrante/saliente de sus puertos.

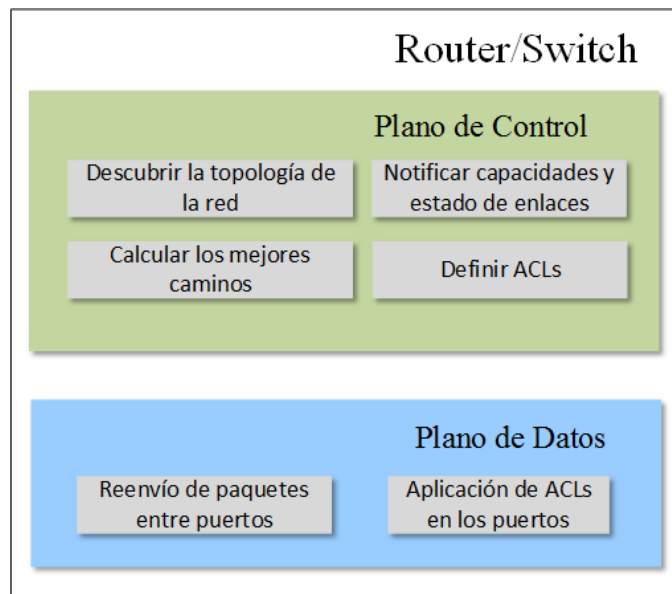


**Figura 4.10: Representación Simplificada de un Router/Switch**

Las primeras cinco funciones pertenecen al plano de control y las 2 últimas funciones pertenecen al plano de datos del equipo. En la Figura 4.11 se muestra una topología de red tradicional conformada por cinco routers/switches que opera un mecanismo de control distribuido.

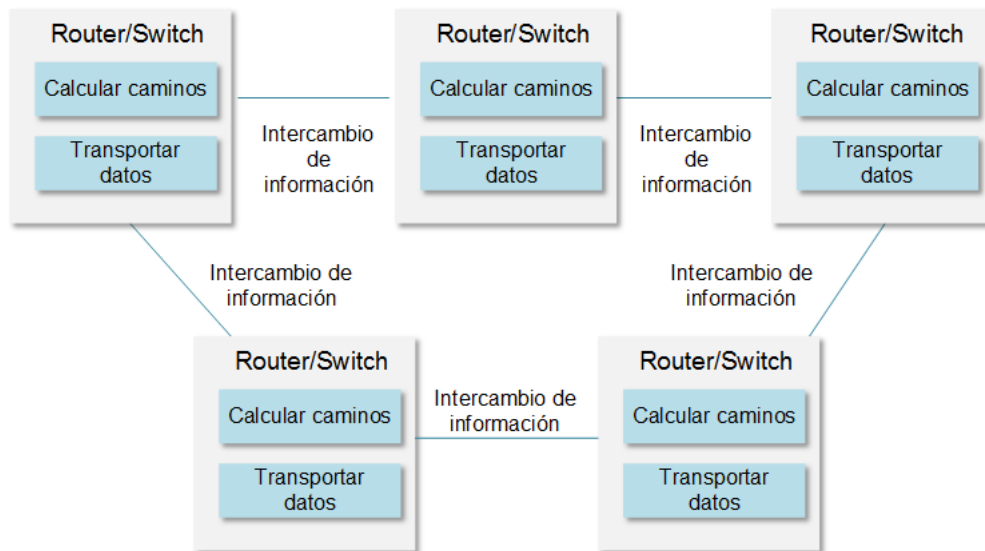


**Figura 4.11: Red Tradicional de Routers/Switches**

Cada dispositivo de la red está conformado de un plano de control y un plano de datos. En esta red de ejemplo se observa que cada equipo intercambia información con el resto de los elementos de la red e internamente efectúa el cálculo de rutas y el transporte de los datos. En este escenario, el administrador de la red debe configurar el protocolo de enrutamiento y las políticas de red en cada dispositivo para que la red sea funcional.

### 4.3.1 Análisis Funcional de la Arquitectura SDN

En una arquitectura SDN la topología de red está conformada por un controlador SDN que actúa como coordinador de la red y un conjunto de dispositivos SDN que reciben instrucciones para transportar los datos (ver Figura 4.12). En este escenario el intercambio de información se lleva a cabo entre el controlador SDN y los dispositivos SDN. El controlador SDN efectúa el cálculo de las rutas y notifica las reglas a los dispositivos SDN los cuales instalan las reglas en sus tablas de reenvío y transportan los datos.

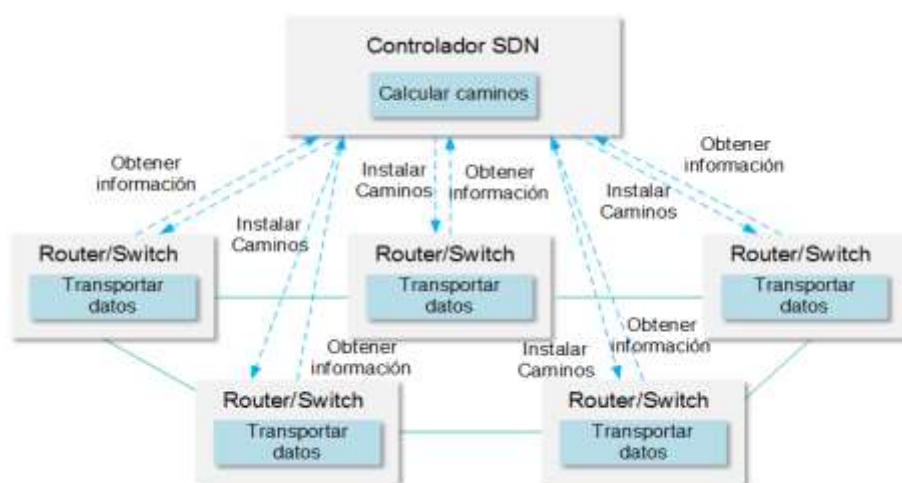


Figura 4.12: Red SDN de Routers/Switches

### 4.3.1 Comparación entre Redes Tradicionales y Redes SDN

La comparación entre las arquitecturas de Redes Tradicionales y SDN se puede analizar desde el punto de vista operacional y funcional.

#### Comparación Operacional entre Redes Tradicionales y SDN

SDN plantea un modelo de operación radicalmente distinto al utilizado en las redes tradicionales. En la Tabla 4.2 se resaltan las principales características de ambos tipos de redes.

| Funcionalidad            | Red Tradicional                                                                     | Red SDN                                                                                |
|--------------------------|-------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| Plano de Control y Datos | Ambos planos de control y datos están localizados en todos los dispositivos de red. | El plano de control es separado del plano de datos y es desplazado al controlador SDN. |
| Inteligencia de Control  | La inteligencia del control es distribuida en cada elemento de la red.              | La inteligencia del control está centralizada en el controlador SDN.                   |

|                           |                                                                                                  |                                                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Programabilidad de la Red | La red no puede ser programada por las aplicaciones. Se debe configurar cada elemento de la red. | La red puede ser programada por las aplicaciones. El controlador puede disponer de APIs para manipular la red. |
|---------------------------|--------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|

**Tabla 4.2: Comparación Operacional entre Redes Tradicionales y Redes SDN**

### **Características Generales de las Redes de Campus Tradicionales**

- Se caracterizan por disponer de un plano de control distribuido entre múltiples dispositivos conformado por la combinación de muchos protocolos de red discretos definidos de manera aislada y que resuelven problemas particulares.
- La red tiene alta resiliencia debido a que el control se encuentra distribuido entre los elementos de la red y al fallar un elemento de la red, el resto de los dispositivos de red implementan un mecanismo de recuperación automática en base al protocolo de comunicación utilizado en la red.
- Los cambios en las políticas de red involucran tocar múltiples protocolos y dispositivos.
- Las redes están conformadas por dispositivos de red propietarios de múltiples fabricantes con diferentes tipos y versiones de sistemas operativos, lo cual dificulta obtener el estado de la red y efectuar cambios sobre la misma.
- Existe un alto grado de dependencia del fabricante de los equipos, lo cual limita el desarrollo de nuevos protocolos o funcionalidades y dificulta la programación de la red.

### **Características Generales de la Arquitectura SDN**

- Permite el control y la gestión centralizada de dispositivos de red de múltiples fabricantes.
- Mejora la automatización y la gestión a través de APIs que ocultan los detalles de bajo nivel de la infraestructura de red subyacente a las aplicaciones.
- Acelera la innovación permitiendo ofrecer nuevos servicios sin tener que configurar dispositivos individuales.
- Permite programar la red a través de un ambiente de programación común.
- Mejora la experiencia de los usuarios al sacarle provecho a la información del estado de la red para adaptar el comportamiento de reenvío a las necesidades de los usuarios.
- La red tiene poca resiliencia y se puede presentar una condición de un solo punto de falla, lo cual requiere el despliegue de varios controladores con mecanismos de cooperación y recuperación automáticos.

Con SDN los administradores pueden programar la red en lugar de tener que codificar cientos de líneas de configuración entre cientos de dispositivos. Adicionalmente, al brindar inteligencia centralizada a los controladores SDN, el grupo IT puede modificar en tiempo real el comportamiento de la red y desplegar nuevas aplicaciones y servicios en cuestión de horas o días, a diferencia de semanas o meses, necesarios con el esquema de redes tradicionales.

La arquitectura SDN soporta un conjunto de APIs que permiten implementar servicios de red comunes, incluyendo enrutamiento, multicast, seguridad, control de acceso, gestión de ancho de banda, ingeniería de tráfico, QoS, optimización de almacenamiento, uso eficiente de energía y gestión de políticas para satisfacer los requerimientos de las empresas.

Desde el lado de las aplicaciones de red, estas ven al controlador a través de una sola API, lo cual facilita la creación y desarrollo de nuevas aplicaciones que permitan dirigir el flujo de tráfico de la red para satisfacer requerimientos específicos de las empresas en términos de gestión, desempeño y seguridad.

### **Comparación Funcional entre Redes Tradicionales y SDN**

Del análisis anterior se puede apreciar que las redes SDN ofrecen grandes beneficios en comparación con las redes tradicionales en 4 aspectos fundamentales: Operación, Configuración, Desempeño e Innovación (ver Tabla 4.3 tomada de [48]).

| <b>Funcionalidad</b> | <b>Red Tradicional</b>                                                                                                | <b>Red SDN</b>                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| Características      | un nuevo protocolo por problema, control de red complejo                                                              | plano de control y datos desacoplados y programabilidad                                                                                              |
| Configuración        | configuración manual propensa a errores                                                                               | configuración automatizada con validación centralizada                                                                                               |
| Desempeño            | información limitada y configuración estática limita la mejora del desempeño de la red                                | control global dinámico con información de cruce de capas                                                                                            |
| Innovación           | implementación en hardware difícil para nuevas ideas, ambiente de pruebas limitado y largo proceso de estandarización | facilita la implementación de nuevas ideas, ambiente de pruebas suficiente con aislamiento, desarrollo rápido utilizando actualizaciones en software |

**Tabla 4.3: Comparación Funcional entre Redes Tradicionales y Redes SDN**

## **4.4 Modelos de Despliegue SDN**

En la práctica se utilizan tres modelos de despliegue SDN: (1) Modelo SDN Basado en Dispositivos, (2) Modelo SDN Overlay, y (3) Modelo SDN Híbrido.

### **4.4.1 Modelo Basado en Dispositivos SDN**

El Modelo Basado en Dispositivos SDN [49] se refiere a una red de switches físicos SDN que operan solo bajo las instrucciones de un controlador SDN. En la Figura 4.13 (tomada de [49]) se presenta un despliegue bajo este esquema. En este ejemplo se cuenta con una red SDN/OpenFlow conformada por un controlador SDN central, y seis switches físicos SDN. En el ejemplo, el controlador SDN instruye a los switches la configuración de las reglas de reenvío que permiten el transporte de los datos en la red.

### **Ventaja y Limitación del Modelo Basado en Dispositivos SDN**

Se implementa con rapidez en escenarios *Greenfield* (despliegue donde la red y los equipos son totalmente nuevos), como un nuevo complejo de oficinas dentro de un campus. La principal limitación de este modelo es que se desaprovechan las funcionalidades del reenvío de paquetes de entornos tradicionales.

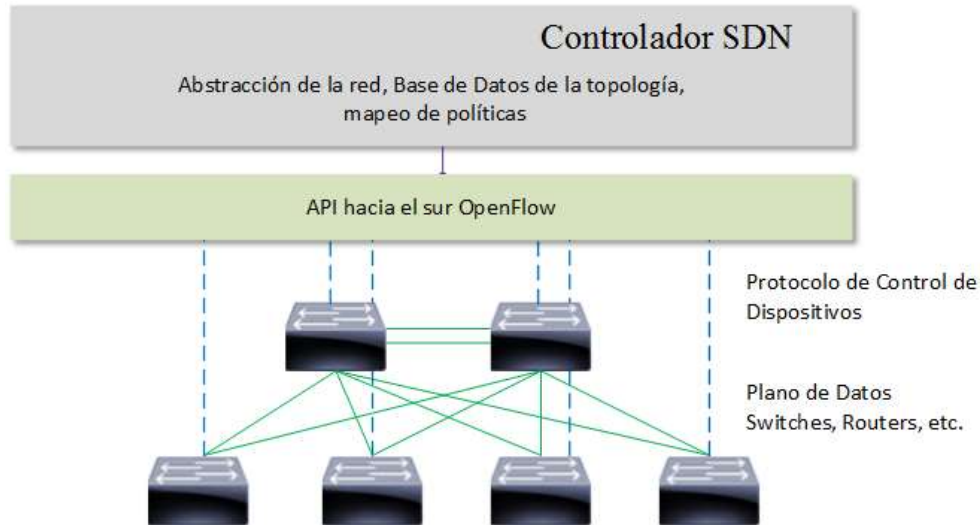


Figura 4.13: Ejemplo del Modelo Basado en Dispositivos SDN

#### 4.4.2 Despliegue SDN Overlay

El modelo SDN Overlay [49] está basado en la superposición de redes (overlay networks) sobre una infraestructura de red física subyacente. SDN Overlay utiliza tecnologías de túneles para crear puntos terminales dentro de los switches virtuales de los hipervisores y se sustenta en la fábrica de la red para transportar paquetes encapsulados a los puntos terminales relevantes usando los protocolos de enrutamiento y switching existentes. En una red SDN Overlay, los nodos finales SDN son dispositivos virtuales que forman parte de hipervisores en un ambiente de virtualización de servidores (ver Figura 4.14 tomada de [47]). En este escenario el controlador controla el reenvío del tráfico de los switches lógicos que se encuentran definidos en los hipervisores y no altera la red física actual ni el plano de control distribuido de la red subyacente. El plano de control centralizado provee una red superpuesta que utiliza la red subyacente como transporte de red. Para crear la red virtual, los nodos SDN lógicos establecen túneles overlay entre sí a través de alguno de los siguientes protocolos de túneles: (1) VXLAN [50] (Virtual Extensible LAN), (2) NVGRE [51] (Network Virtualization using Generic Routing Encapsulation), o (3) STT [52] (Stateless Transport Tunneling). Los túneles overlay usualmente terminan en los switches virtuales dentro de los hipervisores o en dispositivos físicos que actúan como gateways hacia la red existente. Un solo servidor físico puede hospedar múltiples VMs y cada VM puede pertenecer a una red virtual separada. Una VM puede comunicarse con otras VMs en su red virtual, o cruzar la frontera y comunicarse con VMs que formen parte de otras redes virtuales.

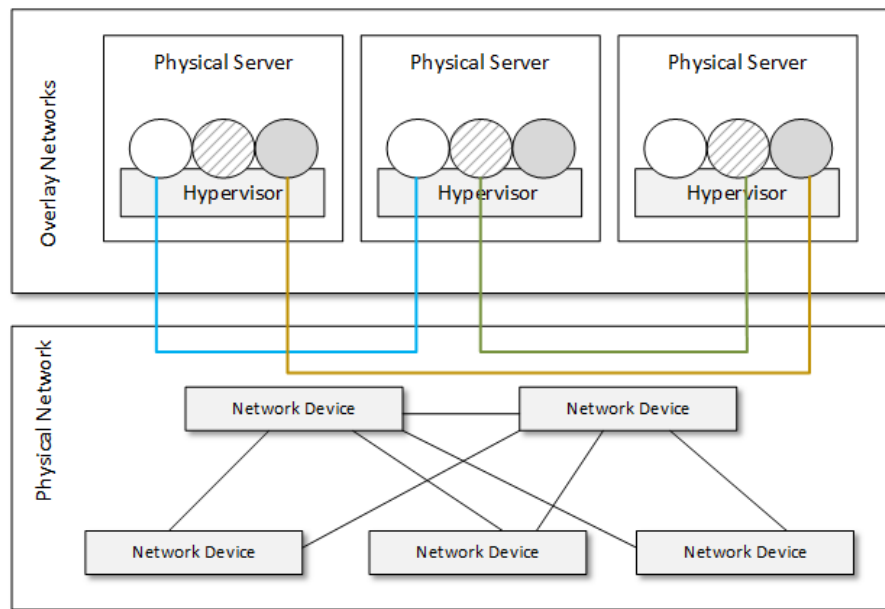


**Figura 4.14: Redes Overlay**

La red física puede utilizar cualquier tecnología de red tradicional y puede ser una red capa 2 o capa 3. En la Figura 4.15 (tomada de [49]) se muestra el despliegue de una red SDN Overlay.

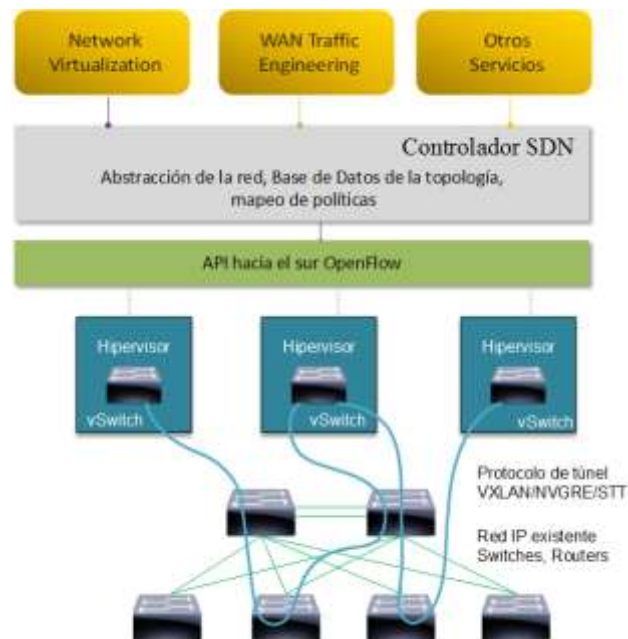


**Figura 4.15: Modelo SDN Overlay**

La red virtual consiste de switches lógicos interconectados por enlaces virtuales punto a punto. Bajo este escenario las aplicaciones SDN tienen acceso a los puertos de los switches virtuales que corren en cada hipervisor y los hipervisores

inyectan tráfico a la red virtual y reciben tráfico de la misma. En el ejemplo una aplicación de virtualización se encarga de gestionar los hosts y túneles que se establecen en una red overlay. La aplicación se comunica con el controlador SDN a través de una API hacia el norte.

El controlador SDN se apoya en el protocolo hacia el sur OpenFlow para aprovisionar las reglas de reenvío a las tablas de flujos de los switches virtuales vSwitch presentes en los hipervisores de las VMs. Las reglas indican la acción a tomar cuando existe una concordancia con la dirección IP de un switch virtual que conecta una VM destino. Una acción pudiera ser reenviar el paquete a un puerto virtual de un túnel específico o la creación de un túnel si no existe una entrada en la tabla de flujos del switch virtual que se encuentra en el hipervisor. Los switches virtuales se comunican entre sí estableciendo túneles a través de mecanismos de túneles overlay como VXLAN [50], NVGRE [51] o STT [52]. El tráfico de las redes virtuales pasa a través de los dispositivos físicos, pero los puntos terminales no tienen conciencia de los detalles de la topología física, ni la manera en la cual ocurre el enrutamiento. Debido a que estas redes virtuales se montan encima de la infraestructura física, estas pueden ser controladas por dispositivos en el borde de la red, que pudieran ser los hipervisores de las VMs que corren en cada servidor.

### **Ventajas del modelo SDN Overlay**

Entre las principales ventajas del modelo SDN Overlay se encuentran:

- Se pueden correr varias redes virtuales simultáneamente sobre la misma infraestructura de red física.
- Se agiliza el despliegue en centros de datos con manejo de virtualización de almacenamiento y cómputo en sus servidores, debido a que el modelo es implementado en software, y las redes virtuales se pueden construir y destruir en una fracción del tiempo requerido para cambiar la infraestructura de red física, reduciendo el tiempo de aprovisionamiento de días a minutos.
- Se resuelve el problema de la extenuación de las direcciones MAC en los centros de datos y ambientes de nube, debido a que las direcciones MACs son ocultadas en los frames encapsulados. Se resuelve el problema del agotamiento de las VLANs en los centros de datos, debido a que el tráfico pasa a través de los túneles y no se requiere de VLANs para aislar múltiples clientes.
- El modelo se puede utilizar en aquellos escenarios donde se requiere hacer una implementación rápida de una solución SDN montada sobre una red IP existente.

### **Limitaciones del modelo SDN Overlay**

La carencia de visibilidad sobre la red subyacente trae como consecuencia las siguientes implicaciones:

- Las redes virtuales y físicas son entidades separadas, posiblemente con diferentes políticas de garantías de servicio, gestión de políticas, aprovisionamiento y puntos de control.

- Las redes virtuales pueden crecer y evolucionar mucho más rápido que las redes físicas subyacentes.
- Los gateways entre las redes virtuales y físicas y los puntos de servicio de red en la red física pudieran requerir pasar altos volúmenes de tráfico, lo cual requiere un hardware de red con gran desempeño.
- Se pierde visibilidad en la red limitando la aplicación de servicios diferenciados e ingeniería de tráfico, distribución de cargas y aislamiento de tráfico.
- La superposición de redes hace compleja la depuración y la detección de fallas en la red.

#### 4.4.3 Despliegue SDN Híbrido

El Modelo SDN Híbrido [49] (ver Figura 4.16 tomada de [49]) está basado en la convivencia de tecnologías de redes tradicionales con tecnologías de red SDN en un mismo entorno.

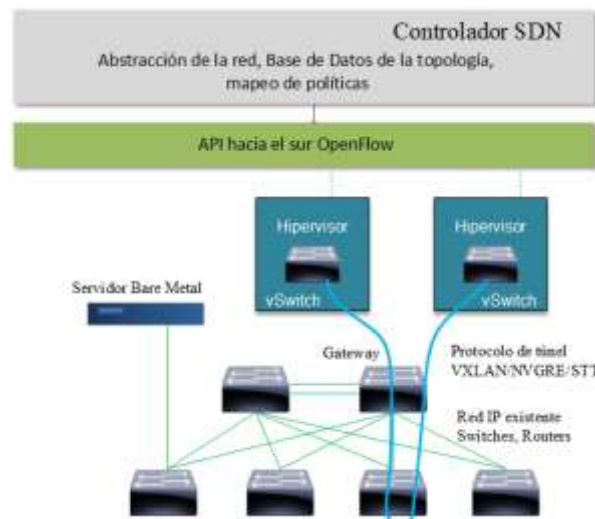


Figura 4.16: Ejemplo del Modelo SDN Híbrido

En el ejemplo anterior un gateway corre tanto el modelo SDN Overlay, como el modelo SDN Basado en Dispositivos. El gateway se enlaza con los switches virtuales que corren sobre los hipervisores de la red overlay a través de un protocolo de túnel que puede ser VXLAN, NVGRE, o STT. La red overlay corre sobre una red IP tradicional a través un protocolo de túnel. Adicionalmente la red IP existente se comunica con la red SDN a través del gateway SDN.

Se recomienda este modelo en despliegues *Brownfield* donde ya se encuentra en operación una red y se desea migrar a un modelo SDN. El modelo plantea la incorporación de gateways o switches SDN híbridos que soportan protocolos de red LAN tradicionales y protocolos SDN. Los gateways SDN definen reglas para que algunos puertos se rijan por reglas SDN y para que otros puertos se rijan por protocolos de red tradicionales. También plantea la combinación de modelos en una misma red, en la cual una parte de la red opera en base al modelo Basado en Dispositivos SDN y otra parte de la red opera bajo el modelo SDN Overlay.



## 5. OpenFlow

La ONF es una organización sin fines de lucro que está liderando el avance y la estandarización de elementos críticos de la arquitectura SDN, tales como el protocolo OpenFlow, el cual provee la interfaz de comunicación Southbound entre el controlador y los dispositivos SDN. OpenFlow [2] es la primera interfaz estandarizada diseñada específicamente para SDN, brindando alto desempeño, y control de tráfico granular en dispositivos de múltiples fabricantes. Una red SDN basada en OpenFlow puede ser implementada en switches de hardware y software, ofreciendo beneficios a las empresas y a las operadoras de servicios, incluyendo:

- Gestión y control centralizado de dispositivos de red de múltiples fabricantes.
- Mejoras en la automatización y gestión por medio de APIs abiertas que abstraen los detalles de la red física subyacente a las aplicaciones y sistemas de aprovisionamiento y orquestación de los elementos de red.
- Rapidez en la innovación permitiendo ofrecer nuevas capacidades y servicios de red sin tener que configurar dispositivos individuales o esperar la liberación de actualizaciones por parte de los fabricantes de los equipos.
- Capacidad de programar la red utilizando ambientes de programación comunes a usuarios, programadores de red y vendedores de software independientes.
- Incremento en la seguridad y confiabilidad de la red como resultado de centralizar y automatizar la gestión de los dispositivos de red.
- Control de red más granular, con la capacidad de aplicar políticas amplias de red a nivel de sesión, usuarios, dispositivos y aplicaciones.
- Mejora en la experiencia del usuario a medida que las aplicaciones aprovechan la información del estado de la red para adaptar su comportamiento a las necesidades de los usuarios.

OpenFlow es un protocolo SDN abierto que puede ser utilizado para controlar centralizadamente switches y flujos de tráfico en una red y permite el acceso directo y la manipulación del plano de datos de routers y switches, tanto físicos, como virtuales basados en hipervisores. En una red SDN, el plano de control es desacoplado de la red física y colocado en un controlador central. El controlador utiliza OpenFlow para comunicarse con todos los componentes de la red, como se indica en la Figura 5.1 (tomada de [106]). A través de este protocolo los administradores de red pueden gestionar la red como un todo, en lugar de configurar dispositivo por dispositivo. La primera implementación de referencia de OpenFlow versión 0.1.0 fue liberada el 30 de noviembre del 2007<sup>13</sup> con el propósito de permitir a los investigadores correr protocolos experimentales en redes de producción y está basado en switches Ethernet que tienen una tabla de flujos interna, y una interfaz estandarizada para añadir y remover entradas de flujos [2].

---

<sup>13</sup> <http://archive.openflow.org/wp/previous-versions-archive>

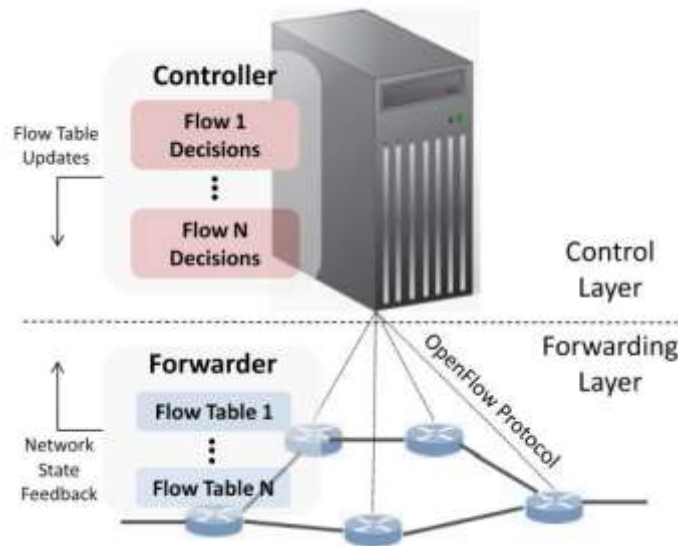


Figura 5.1: Arquitectura de Red OpenFlow

El protocolo se implementa en ambos lados de la interfaz entre los dispositivos de la infraestructura de red y el software de control SDN. Utiliza el concepto de flujos para identificar el tráfico de la red basado en reglas estáticas predefinidas o en reglas programadas dinámicamente por el software de control SDN. Debido a que OpenFlow permite que la red sea programada en una base por flujos, una arquitectura SDN basada en OpenFlow provee control granular del tráfico de red, permitiendo que la red responda a cambios en tiempo real a nivel de las aplicaciones, usuarios o sesiones.

## 5.1 Arquitectura OpenFlow

La arquitectura de red OpenFlow consiste de tres conceptos básicos: (1) la red está soportada por switches con capacidad OpenFlow que conforman el plano de datos, (2) el plano de control consiste de uno o más controladores OpenFlow que definen y publican las reglas de reenvío al plano de datos de los switches OpenFlow y (3) el controlador y los switches OpenFlow se comunican entre sí mediante un canal de control seguro.

Una red OpenFlow está conformada por dispositivos de switching con una tabla de flujos interna, y provee una plataforma de switching virtualizada, programable y abierta para controlar el hardware del switch via software [53]. Puede implementar las funciones de un switch, un router, o ambos, y permite que el plano de control de los dispositivos SDN sean controlados programáticamente [2].

### 5.1.1 Switch OpenFlow

Un switch OpenFlow [54] es un dispositivo de red que reenvía paquetes de acuerdo a una tabla de flujos interna. La tabla de flujos contiene un conjunto de entradas de flujos que tienen campos de cabeceras, contadores y acciones. Los

campos de cabecera se utilizan para determinar la concordancia de los paquetes con las entradas de las tablas de flujos. Los campos de cabecera pueden aplicar a diferentes tipos de protocolos, dependiendo de la versión de la especificación OpenFlow, como LAN Ethernet, IPv4, IPv6, o MPLS. Los contadores permiten recopilar estadísticas de los flujos, como el número de bytes y paquetes recibidos o la duración de flujos en las tablas. Las acciones definen el tratamiento que se le hace a los paquetes, como reenviar los paquetes a un puerto, modificar un paquete, o descartar un paquete.

### 5.1.2 Funcionamiento de Switches OpenFlow

En una arquitectura OpenFlow, el reenvío de los datos se efectúa en los switches de la red, y las decisiones de reenvío se hacen en un programa de software de un controlador externo implementado en un servidor que se comunica con los switches a través del protocolo OpenFlow (ver Figura 5.2 tomada de [22]).

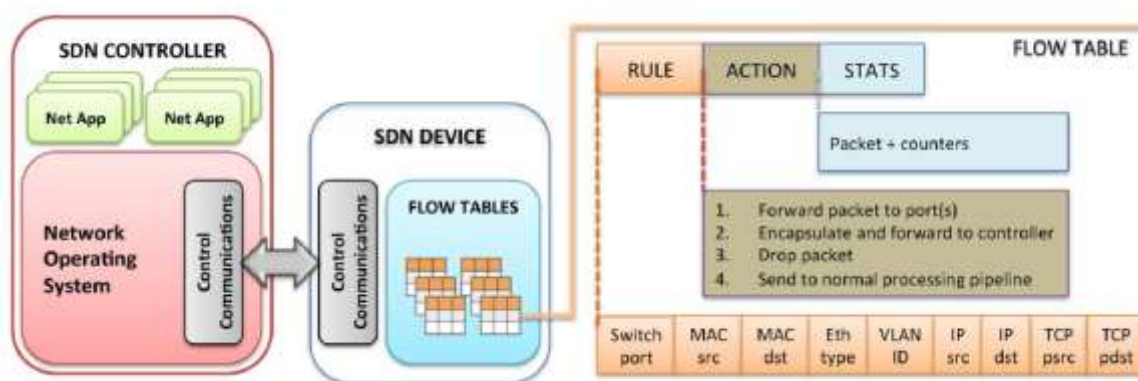


Figura 5.2: Dispositivo SDN/OpenFlow

A través del protocolo OpenFlow, un controlador puede añadir, actualizar, y eliminar entradas de flujos en las tablas de flujos, de manera proactiva o de manera reactiva en respuesta a la llegada de paquetes, modificando el comportamiento de reenvío del plano de datos de los switches.

Dentro de un dispositivo OpenFlow, cuando un paquete ingresa a alguno de sus puertos se inicia un proceso de consulta en la tabla de flujos para encontrar una entrada de flujos que concuerde con el paquete. Las entradas de flujos son evaluadas en orden de prioridad y se utiliza la primera concordancia en la tabla. Si existe una concordancia, se ejecutan las acciones indicadas en la entrada de flujo y se actualizan los contadores de paquetes para la entrada de flujos correspondiente. En caso de no existir concordancia en alguna entrada de la tabla de flujos, el paquete se envía al controlador sobre el canal OpenFlow o se descarta el paquete.

Entre las instrucciones asociadas con cada entrada de flujo se encuentran: (1) reenviar el paquete a un puerto, (2) modificar los campos de cabecera del paquete y (3) descartar el paquete. Las entradas de flujos pudieran indicar enviar un

paquete a un puerto físico, o a un puerto virtual definido por el switch, o a un puerto virtual reservado por la especificación OpenFlow. Los puertos virtuales reservados pudieran especificar acciones de reenvío genéricas como reenvío a un controlador, hacer una inundación de puertos, o reenviar los paquetes usando métodos no-OpenFlow, como el procesamiento de un switch Ethernet tradicional. Los puertos virtuales definidos por el switch permiten especificar grupos de agregación de enlaces, túneles o interfaces de *loopback*.

### Flujo OpenFlow

Un flujo OpenFlow es una secuencia unidireccional de paquetes que comparte características comunes, como las direcciones MAC origen y/o destino, las direcciones IP origen y/o destino y los puertos TCP/UDP origen y/o destino, entre otros.

### Tipos de Switches OpenFlow

Existen dos categorías de switches OpenFlow: switches solo-OpenFlow (OpenFlow-only), cuyo procesamiento de paquetes está basado únicamente en el protocolo OpenFlow y switches OpenFlow-híbrido (OpenFlow-hybrid), que soportan tanto la operación OpenFlow como la operación de switching Ethernet tradicional [55].

#### 5.1.3 Switch solo-OpenFlow (OpenFlow-only)

Un switch solo-OpenFlow soporta únicamente operaciones OpenFlow y todos los paquetes son procesados de acuerdo a las reglas definidas desde un controlador externo de acuerdo a las especificaciones del estándar OpenFlow. (ver Figura 5.3 (tomada de [2])). Los switches *solo-OpenFlow* soportan solamente la operación OpenFlow, y todos los paquetes son procesados por un *pipeline* OpenFlow, y no pueden ser procesados de otra forma.

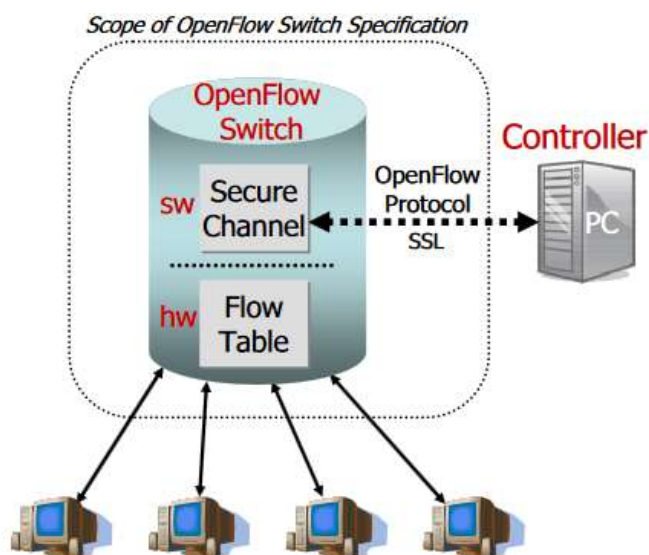


Figura 5.3: Operación de Switches Solo-OpenFlow

#### 5.1.4 Switch OpenFlow-híbrido (OpenFlow-hybrid)

Un switch OpenFlow-híbrido puede operar tanto en modo OpenFlow, como en modo de switching Ethernet tradicional. En la modalidad OpenFlow, el switch efectúa el procesamiento de paquetes dirigido por instrucciones de un controlador. En la modalidad de switching tradicional el switch ejecuta servicios de switching Ethernet Capa 2, segmentación de VLANs, enrutamiento Capa 3 (IPv4, IPv6), ACLs y QoS, entre otros. La Figura 5.4 (tomada de [2]) muestra una red OpenFlow híbrida conformada por APs solo-OpenFlow y switches solo-OpenFlow en la parte superior y por un switch comercial OpenFlow-híbrido en la parte inferior que puede comunicarse tanto a una red tradicional, como a una red OpenFlow.

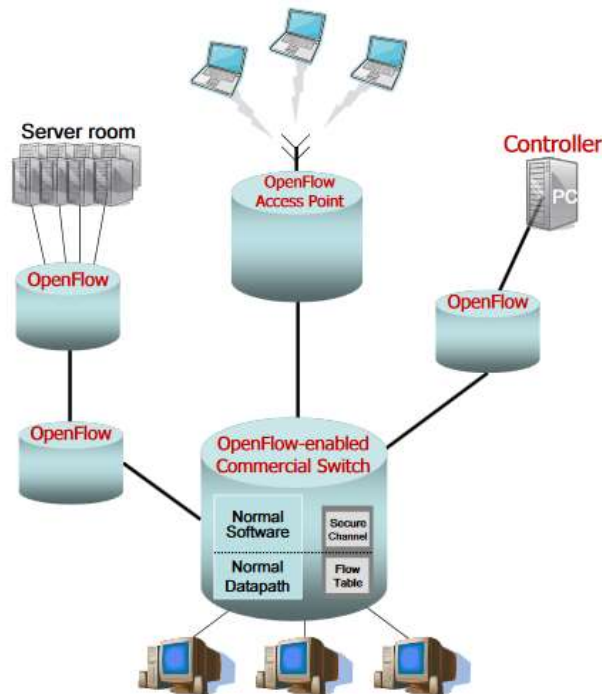


Figura 5.4: Ejemplo de Red SDN con Switch OpenFlow-Híbrido

En este ejemplo el switch comercial dispone de tablas de flujos OpenFlow alimentadas por un controlador OpenFlow y de tablas de reenvío Ethernet alimentadas a través de un proceso de aprendizaje dinámico de direcciones MAC.

Estos switches deben disponer de un mecanismo de clasificación fuera de la especificación OpenFlow que permita enrutar tráfico al *pipeline* OpenFlow o al *pipeline* Ethernet tradicional. Por ejemplo, un switch pudiera utilizar una etiqueta de VLAN o un puerto de entrada del paquete para decidir si procesar el paquete utilizando un *pipeline* u otro, o este pudiera dirigir todos los paquetes al *pipeline* OpenFlow. Un switch OpenFlow-híbrido pudiera permitir también que un paquete vaya del *pipeline* OpenFlow al *pipeline* normal a través de los puertos virtuales reservados FLOOD y NORMAL. Los paquetes enviados a un puerto FLOOD son inundados a todos los puertos estándar del switch, excepto el puerto origen, usando un *pipeline* de procesamiento de paquetes tradicional del switch. Los

paquetes enviados a un puerto NORMAL procesan los paquetes utilizando el *pipeline no-OpenFlow* Ethernet tradicional del switch.

### 5.1.5 Controlador OpenFlow

En una arquitectura de red SDN basada en OpenFlow un controlador crea, actualiza y remueve entradas de flujo en las tablas de flujos de los switches. En la actualidad existen varias implementaciones de controladores de código abierto implementados en varios lenguajes de programación como Python, C++, y Java<sup>14</sup>. Entre los controladores de código abierto más populares se encuentran NOX<sup>15</sup>, POX<sup>16</sup>, Trema<sup>17</sup>, Floodlight y OpenDaylight<sup>18</sup>.

Generalmente, un controlador corre sobre un servidor conectado a la red, y puede atender uno o múltiples switches dependiendo del diseño de la red. Los controladores pueden ser desplegados siguiendo una jerarquía centralizada donde un controlador maneja y controla todos los switches de una red o puede seguir una jerarquía distribuida donde dos o más controladores manejan y controlan dos o más grupos de switches en una red. En una jerarquía centralizada, si el controlador falla, la red sigue operando con las reglas de reenvío de tráfico presentes hasta el momento, pero no se pueden actualizar nuevas reglas, ni incorporar nuevos servicios.

En una jerarquía distribuida, todos los controladores deben tener la misma copia de la vista de la topología de red en tiempo real para evitar la pérdida de paquetes, y la red puede continuar operando y actualizándose en caso de falla del controlador principal gracias a un mecanismo de redundancia y cooperación entre los controladores de la red.

## 5.2 Tipos de Flujos

Los flujos pueden ser clasificados como microflujos y flujos agregados de acuerdo al número de hosts destino. Para cada tipo de flujo se tienen las siguientes consideraciones:

### Microflujos

- La tabla de flujos contiene una entrada por flujo.
- Cada flujo es establecido individualmente por el controlador.
- Las entradas de flujo deben coincidir exactamente.
- Aplica en escenarios donde se requiere un control grano fino, aplicación de políticas, y monitoreo.

---

<sup>14</sup> <http://yuba.stanford.edu/~casado/of-sw.html>

<sup>15</sup> <https://github.com/noxrepo/nox-classic/wiki>

<sup>16</sup> <https://openflow.stanford.edu/display/ONL/POX+Wiki>

<sup>17</sup> <https://github.com/trema/trema>

<sup>18</sup> <http://www.opendaylight.org>

### Flujos Agregados

- Una entrada de flujo cubre grandes grupos de flujos.
- Acepta entradas de flujos con comodines (*wildcard*).
- La tabla de flujo contiene una entrada por categoría de flujos.
- Aplica en escenarios donde se tiene gran cantidad de flujos como en redes de backbone.

## 5.3 Modos de Cargas de Flujos

Existen dos modos para poblar entradas en las tablas de flujos: Modo Reactivo y Modo Proactivo.

### Modo Reactivo

En este caso el primer paquete del flujo dispara una llamada al controlador para que pueda crear e insertar una nueva entrada en la tabla de flujos en los switches que haga referencia al nuevo paquete.

### Modo Proactivo

En este caso, las tablas de flujo de los switches son *pre-pobladas* por el controlador. Bajo este esquema en caso de pérdida de la conexión entre el controlador y el switch el tráfico no es interrumpido.

## 5.4 Tablas OpenFlow

Un switch OpenFlow puede contener una sola tabla de flujos en la especificación OpenFlow 1.0.0 [54] y varias tablas de flujos y una tabla de grupo especial *Tabla de Grupos* a partir de la especificación OpenFlow 1.1.0 [55][55].

### 5.4.1 Tablas de Flujos

Las tablas de flujos se encargan de encontrar concordancias de los paquetes entrantes con entradas de flujos particulares y especifican las acciones a ejecutar en dichos paquetes. Cada tabla de flujo en un switch OpenFlow contiene un conjunto de entradas de flujos. En la especificación OpenFlow 1.0.0 [54], una entrada de flujo consiste de tres campos: (1) cabecera (Header Fields), (2) contadores (Counters) y (3) acciones (Actions) (ver Tabla 5.1).

| Header Fields | Counters | Actions |
|---------------|----------|---------|
|               |          |         |

Tabla 5.1: Campos de la Tabla de Flujos de un Switch OpenFlow 1.0.0

Los switches OpenFlow toman sus decisiones de reenvío efectuando consultas de los campos de cabecera *Header Fields* de los paquetes entrantes con las entradas de flujos de la tabla de flujos y aplican las reglas del campo *Actions* cuando exista alguna concordancia. Los campos contadores se utilizan para actualizar estadísticas de los paquetes concordantes con los campos de cabecera y los

campos de acciones se utilizan para aplicar acciones a los paquetes que coinciden con las entradas de flujos.

### Campos (Header Fields)

OpenFlow 1.0.0 fue la primera especificación OpenFlow estandarizada y es la especificación de mayor uso. En esta especificación el campo *Header Fields* se utiliza para verificar la concordancia de paquetes con las entradas de la tabla de flujos de los switches a través de una 12-tupla, como se muestra en la Tabla 5.2.

| Port    | Ethernet |    |      | VLAN    |               | IP |    |       |          | TCP/UDP |     |
|---------|----------|----|------|---------|---------------|----|----|-------|----------|---------|-----|
| In Port | SA       | DA | Type | VLAN ID | VLAN Priority | SA | DA | Proto | ToS bits | Src     | Dst |
|         |          |    |      |         |               |    |    |       |          |         |     |

Tabla 5.2: Campos Header Fields OpenFlow Versión 1.0.0

Cada campo de cabecera puede tener expresado un valor comodín para permitir la agregación de flujos. Por ejemplo se puede definir una regla que incluya valores *wildcard* en todos los campos en la entrada de flujo, a excepción del campo VLAN ID para representar todo el tráfico de datos de una VLAN particular. La tabla de flujos pudiera incluir una entrada de flujo table-miss en la cual todos los campos tienen *wildcards* y tiene la prioridad más baja (prioridad 0) para indicar que no existe concordancia del paquete.

Cada entrada de la tabla de flujos tiene asociada una acción simple. Las tres acciones que deben cumplir los switches OpenFlow se enumeran a continuación:

- Reenviar los paquetes de un flujo a un puerto o grupo de puertos determinado.
- Encapsular y reenviar los paquetes de un flujo a un controlador por un canal seguro.
- Descartar los paquetes de un flujo.

En un switch OpenFlow los paquetes pueden ser reenviados a un puerto físico, o a un puerto lógico o a un puerto reservado. Un puerto lógico pudiera representar grupos de agregación de enlaces, túneles o interfaces de bucle. Un puerto reservado pudiera ejecutar una acción de reenvío específica, como enviar un paquete desconocido al controlador, hacer una difusión de tráfico, o reenviar el paquete usando un método de reenvío de switch tradicional. Las entradas de la tabla de flujos pueden incluir otras acciones adicionales incluyendo modificar valores en los campos de los paquetes, reenviar los paquetes a una tabla de grupos o enviar los paquetes a otras tablas de flujos.

#### 5.4.2 Ejemplo de Tabla de Flujos OpenFlow v1.0

En la Figura 5.5 tomada de [56] se presenta un ejemplo de una tabla de flujos de un switch OpenFlow v1.0. En este ejemplo, se presentan las siguientes reglas para todos los paquetes que ingresan al switch:

| Ingres<br>s Port | Source<br>MAC | Dest<br>MAC | Ether<br>Type | VLAN<br>ID | VLAN<br>Priorit<br>y | IP<br>SRC | IP<br>DEST  | IP<br>Protocol | IP<br>TOS | TCP/U<br>DP<br>SRC | TCP/U<br>DP<br>DEST | Action                       | Priority | Counter |
|------------------|---------------|-------------|---------------|------------|----------------------|-----------|-------------|----------------|-----------|--------------------|---------------------|------------------------------|----------|---------|
| *                | 3c:07:54:*    | *           |               |            |                      | *         | *           | *              | *         | *                  | *                   | Fwd Port 10                  | 100      |         |
| *                | *             | *           |               |            |                      | *         | 192.168.1.* | *              | *         | *                  | *                   | Fwd Port 12                  | 100      |         |
| Port 1           | *             | *           |               |            |                      | *         | *           | *              | *         | *                  | *                   | Fwd Port 14...24             | 100      |         |
| *                | *             | *           |               |            |                      | *         | *           | *              | *         | *                  | 25                  | Drop                         | 100      |         |
| *                | *             | *           |               |            |                      | *         | *           | 0x0800         | *         | *                  | *                   | Controller                   | 100      |         |
| *                | 00:01:E7:*    | *           |               | Vlan10     | *                    | *         | *           | *              | *         | *                  | 80                  | Fwd Port 8                   | 200      |         |
| *                | *             | *           |               |            |                      | *         | 192.168.1.* | *              | *         | *                  | 80                  | Rewrite 10.1.2.3; Fwd port 9 | 200      |         |
|                  |               |             |               |            |                      | *         | 10.*        | *              | *         | *                  | *                   | Local                        | 200      |         |

BRKSDN-1014

© 2015 Cisco and/or its affiliates. All rights reserved.

Cisco Public

24

Figura 5.5: Ejemplo de una Tabla de Flujos OpenFlow 1.0.0

- **Switching:** Todos los paquetes cuya dirección MAC origen sea 3c:07:54:\* son enviados al puerto 10.
- **Routing:** Todos los paquetes cuya dirección IP destino sea 192.168.1.\* son enviados al puerto 12.
- **Replicación/SPAN:** Todos los paquetes que ingresen al puerto 1 son replicados en los puertos que van desde el puerto 14 hasta el puerto 24.
- **Firewall/Seguridad:** Todos los paquetes con número de puerto TCP/UDP destino 25 son descartados.
- **Inspección:** Todos los paquetes con protocolo IP 0x0800 son reenviados al controlador para análisis ulterior.
- **Combinaciones:** Todos los paquetes cuya dirección MAC origen sea 00:01:E7:\* o que pertenezcan a la vlan 10 o que tengan puerto TCP/UDP destino son enviados al puerto 8.
- **Multi-acción/NAT:** Todos los paquetes cuya dirección IP destino sea 192.169.1.\* y con puerto TCP/UDP tendrán nueva dirección IP origen 10.1.2.3 y son enviados al puerto 9.
- **Manejo local:** Todos los paquetes pertenecientes a la subred 10.\*.\* son manejados localmente.

### 5.4.3 Tablas de Grupos

La tabla de grupos es una tabla especial que permite ejecutar acciones sobre un conjunto de flujos con características similares. A través de las tablas de grupos se pueden enviar entradas de flujos a un solo identificador, por ejemplo, un próximo salto común. La tabla de grupos contiene entradas de grupo y una lista de acciones que permiten ejecutar funcionalidades de broadcasts, reenvíos multicamino FRR (Fast Reroute), o servicios de agregación de enlaces, como el reenvío IP a un próximo salto común.

#### 5.4.4 Contadores

En una implementación OpenFlow 1.0.0 se mantienen contadores para cada uno de los siguientes elementos: tablas de flujos, entradas de flujos, puertos y colas.

### 5.5 Protocolo OpenFlow

El protocolo OpenFlow describe el intercambio de mensajes que tiene lugar entre un controlador OpenFlow y un switch OpenFlow. Generalmente el protocolo es implementado en el tope de SSL (Secure Socket Layer) o TLS (Transport Layer Security), para proveer un canal OpenFlow seguro o sobre un canal TCP.

El protocolo OpenFlow le permite al controlador ejecutar adiciones, actualizaciones, y eliminaciones a las entradas de las tablas de flujos. A partir de la especificación OpenFlow 1.0.0 [54] se definen tres tipos de mensajes entre el controlador y los switches OpenFlow (ver Tabla 5.3): (1) *Controller to Switch*, (2) *Asynchronous* y (3) *Symmetric*.

#### Controller to Switch

Esta clase de mensajes son enviados por el controlador a un switch para solicitar notificaciones de capacidades, estados de los puertos y estadísticas de paquetes, o para modificar el estado del reenvío del switch, como añadir, eliminar o modificar entradas en las tablas de flujos.

#### Asynchronous

Los switches envían mensajes asíncronos al controlador para denotar la llegada de un nuevo paquete, notificar el cambio de estado en el switch, o informar la ocurrencia de algún error.

#### Symmetric

Estos mensajes son enviados en cualquier dirección sin una solicitud expresa. Los mensajes *Hello* son enviados hacia adelante y hacia atrás entre el controlador y el switch cuando se establece la conexión. Los mensajes *Echo Request* y *Echo Reply* pueden ser utilizados por el controlador o el switch para medir la latencia o el ancho de banda de una conexión entre un controlador y un switch o para verificar la disponibilidad del dispositivo. El mensaje *Experimenter* está disponible para desarrollar funcionalidades en versiones futuras de OpenFlow.

| Campo                | Descripción                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Controller to Switch |                                                                                                                                 |
| Features             | Solicita las capacidades de un switch. El switch responde con una respuesta de funcionalidades que especifican sus capacidades. |
| Configuration        | Establece y consulta parámetros de configuración. El switch responde con los parámetros establecidos.                           |

|                     |                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Modify-State        | Añade, elimina, y modifica entradas en las tablas de flujos/grupos y establece propiedades a los puertos del switch.                                                                                                                                                                                                                                                     |
| Read-State          | Recopila información del switch, como la configuración actual, estadísticas, y capacidades.                                                                                                                                                                                                                                                                              |
| Send-Packet         | Son utilizados por el controlador para enviar paquetes a un puerto específico del switch o para reenviar paquetes recibidos a través de mensajes Packet-In. El mensaje debe contener una lista de acciones ordenadas. Una lista de opciones vacía implica descartar el paquete.                                                                                          |
| Barrier             | Son utilizados por el controlador para asegurarse de que se han cumplido las dependencias entre mensajes o para recibir notificaciones de operaciones completadas.                                                                                                                                                                                                       |
| <b>Asynchronous</b> |                                                                                                                                                                                                                                                                                                                                                                          |
| Packet-In           | Transfiere paquetes al controlador.                                                                                                                                                                                                                                                                                                                                      |
| Flow-Removed        | Informa al controlador sobre la remoción de una entrada en una tabla de flujo.                                                                                                                                                                                                                                                                                           |
| Port-Status         | Informa al controlador de un cambio en un puerto.                                                                                                                                                                                                                                                                                                                        |
| Error               | Notifica al controlador de un error o de una condición de un problema.                                                                                                                                                                                                                                                                                                   |
| <b>Symmetric</b>    |                                                                                                                                                                                                                                                                                                                                                                          |
| Hello               | Mensaje de intercambio entre el switch y el controlador al establecer la conexión de arranque.                                                                                                                                                                                                                                                                           |
| Echo                | Mensajes Echo Request/Reply que pueden ser enviados desde el controlador o el switch, los cuales deben retornar una respuesta Echo Reply. Se utilizan para indicar el estado de vida del controlador o del switch. También se utiliza para efectuar pruebas de medición de la velocidad del ancho de banda o el retardo de la conexión entre el controlador y el switch. |
| Experimenter        | Se utiliza para añadir funciones adicionales.                                                                                                                                                                                                                                                                                                                            |

**Tabla 5.3: Mensajes OpenFlow 1.0.0**

### 5.5.1 Conexiones al Canal OpenFlow

El switch establece una comunicación con el controlador en una dirección IP configurable por el usuario, utilizando un puerto específico e inicia una conexión TLS o TCP. El tráfico que viaja por el canal OpenFlow no transita el *pipeline* OpenFlow.

La primera vez que se establece una conexión OpenFlow, cada lado de la conexión debe enviar un mensaje *OFPT\_HELLO* con el campo versión definido como la versión OpenFlow más alta soportada por el emisor. Al recibir el mensaje, el destino debe calcular la versión del protocolo OpenFlow a ser utilizada como la versión más pequeña enviada y recibida. Si la versión negociada está soportada por el destino, se da inicio a la conexión.

Después de negociada la versión, el controlador emite un mensaje *OFPT\_FEATURES\_REQUEST* el cual es respondido con un mensaje *OFPT\_FEATURES\_REPLY* por el switch, que contiene el número de tablas, la

cantidad de paquetes soportadas en los buffers por el switch y el ID *datapath*. Adicionalmente puede incluir una lista de capacidades del switch, como: estadísticas de flujos, estadísticas de tablas, estadísticas de puertos, estadísticas de grupos, capacidad para reensamblar fragmentos IP, estadísticas de colas, e indicación de bloqueo de puertos haciendo bucles.

Este intercambio de mensajes se conoce como *OpenFlow Handshake*. Después de recibir el mensaje *OFPT\_FEATURES\_REPLY*, el controlador generalmente consulta los parámetros de configuración del switch.

El controlador es capaz de definir y consultar parámetros de configuración al switch con mensajes *OFPT\_SET\_CONFIG* y *OFPT\_GET\_CONFIG\_REQUEST*. El switch responde a una solicitud de configuración con un mensaje *OFPT\_GET\_CONFIG\_REPLY*.

## **5.6 Especificaciones OpenFlow**

En la actualidad existen diferentes versiones de la especificación OpenFlow y algunas especificaciones alcanzaron ya la obsolescencia. La primera versión de OpenFlow, la versión 0.1.0, fue liberada en noviembre del 2007, luego fueron liberándose las siguientes versiones: versiones 0.8.0 y 0.8.1 (mayo, 2008), versión 0.8.2 (octubre, 2008), versión 0.8.9 (diciembre, 2008), versión 0.9 (julio, 2009), versión 1.0 (diciembre, 2009), versión 1.1.0 (febrero 2011), versión 1.2 (diciembre, 2011), versión 1.3.0 (junio, 2012), versión 1.3.1 (septiembre, 2012), versión 1.3.2 (abril, 2013), versión 1.3.3 (septiembre, 2013), versión 1.4.0 (octubre, 2013), versión 1.3.4 (marzo, 2014), y versión 1.5.0 (diciembre, 2015).

En la actualidad las versiones 0.8 y 0.9 ya alcanzaron nivel de obsolescencia, y se recomienda utilizar las especificaciones a partir de la versión 1.0.0. La siguiente sección resalta las principales características de las versiones mayores de OpenFlow que han sido consideradas estables por la ONF.

### **5.6.1 OpenFlow Versión 1.0.0**

La especificación OpenFlow 1.0.0 [54] es la versión con mayor uso en implementaciones de redes SDN. Esta especificación permite una sola tabla de flujos en el switch.

#### **Componentes de un Switch OpenFlow 1.0.0**

Un switch OpenFlow 1.0.0 contiene los siguientes componentes:

- Una tabla de flujos.
- Un canal OpenFlow que conecta el switch al controlador a través de una conexión segura, como TLS [57] (Transport Layer Security) o a través de una conexión TCP plana, para permitir el intercambio de comandos y paquetes entre el controlador y los dispositivos de red.

- El protocolo OpenFlow, que permite el intercambio de mensajes y paquetes entre el controlador y los switches OpenFlow.

### Tabla de Flujos OpenFlow 1.0.0

La tabla de flujos de la especificación OpenFlow 1.0.0 consiste de los siguientes grupos de campos: *Header Fields*, *Counters* y *Actions* (ver Tabla 5.1). Los campos *Header Fields* permiten reconocer los paquetes que pertenecen a una entrada de flujo en la tabla de flujos del switch. Los campos *Counters* permiten obtener estadísticas de los paquetes que coinciden con una entrada de flujos. Los campos *Actions* permiten efectuar acciones a los paquetes que pertenecen a una entrada de flujos.

### Campos de Cabecera Header Fields OpenFlow 1.0.0

La especificación OpenFlow 1.0.0 utiliza 12 campos en la cabecera de los paquetes que ingresan al switch para reconocer una entrada en la tabla de flujos. Un paquete puede coincidir con una entrada de flujo particular en la tabla de flujos utilizando uno o más campos de cabecera del paquete. Un campo en la tabla de flujos puede tener el valor *ANY* y hacer una correspondencia con todos los paquetes.

En la siguiente sección se describen los 12 campos *Header Fields* de un switch OpenFlow 1.0.0 (ver Tabla 5.2):

- **Ingress Port:** Identifica el puerto en el cual arriba el paquete al switch. Puede indicar un puerto físico o un puerto virtual definido en el switch.
- **Ethernet Source Address:** Identifica la dirección Ethernet origen del paquete. Cada entrada puede ser una dirección MAC exacta, un valor enmascarado por bit, en el cual solo se chequea una parte de los bits de la dirección, o un valor comodín *ANY* en la cual coinciden todos los valores de los bits.
- **Ethernet Destination Address:** Identifica la dirección Ethernet destino del paquete. Cada entrada puede ser una dirección MAC exacta, un valor enmascarado por bit, en el cual solo se chequea una parte de los bits de la dirección, o un valor comodín *ANY* el cual representa cualquier valor.
- **Ethernet Type:** Indica cual es el protocolo encapsulado en el *payload* del frame Ethernet.
- **VLAN ID:** identifica la VLAN a la cual pertenece un paquete.
- **VLAN priority CoS:** Campo de prioridad de 3 bits presente en tramas Ethernet que puede ser utilizado por mecanismos de QoS para diferenciar tráfico de red.
- **IPv4 Source Address:** Identifica la dirección IPv4 origen del paquete. Cada entrada puede ser una dirección IPv4 exacta, un valor enmascarado por bit, un valor de una máscara subred, o un valor comodín *ANY*.
- **IPv4 Destination Address:** Identifica la dirección IPv4 destino del paquete. Cada entrada puede ser una dirección IPv4 exacta, un valor enmascarado por bit, un valor de una máscara subred, o un valor comodín *ANY*.
- **IPv4 Protocol Number:** Valor numérico de protocolo que indica la próxima cabecera en el paquete.

- **IP ToS:** Campo ToS (Type of Service) de la cabecera IPv4 que puede especificar la prioridad de un paquete IP.
- **TCP/UDP Source:** Número de puerto TCP o UDP origen o tipo ICMP (Internet Control Message Protocol). Solamente los 8 bits inferiores pueden utilizarse para ICMP.
- **TCP/UDP Destination:** Número de puerto TCP o UDP destino o tipo ICMP (Internet Control Message Protocol). Solamente los 8 bits inferiores pueden utilizarse para ICMP.

### Funcionamiento del Plano de Datos OpenFlow 1.0.0

En la Figura 5.6 (tomada de [58]) se muestran los detalles del plano de datos de un switch OpenFlow 1.0.0. En el paso 1, el paquete que ingresa al switch es enviado a un sistema de análisis de paquetes. En el paso 2, se extraen los campos de cabecera del paquete y se colocan en una cabecera de consulta de paquetes. En el paso 3, la cabecera de consulta de paquetes generada es enviada al sistema de reconocimiento de paquetes. En el paso 4, la cabecera de consulta de paquetes es comparada con las reglas definidas para cada entrada de flujo en la tabla de flujos OpenFlow por orden de prioridad y de manera descendente. En caso de ocurrir una concordancia, se ejecutan las acciones sobre los paquetes indicadas en el campo *Actions* de la entrada de flujo (Paso 5B). En caso contrario, se envían los primeros 200 bytes del paquete al controlador OpenFlow para su procesamiento (Paso 5A) [58].

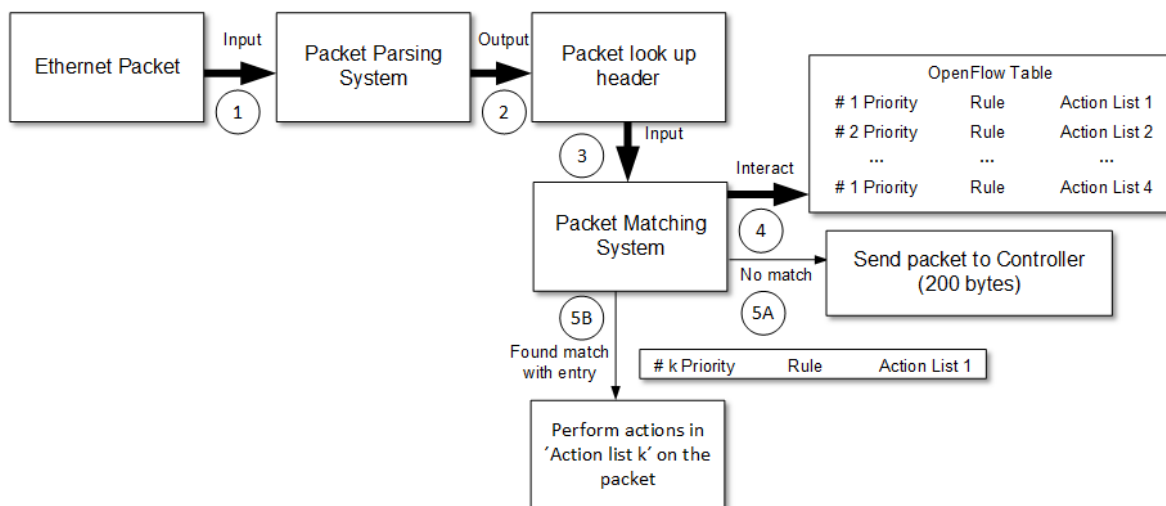


Figura 5.6: Procesamiento de Paquetes en un Switch OpenFlow 1.0.0

Existen varios tipos de acciones que se pueden aplicar a un flujo en la especificación 1.0.0. La acción más importante es la instrucción de reenvío. Esta acción reenvía un paquete a un puerto específico o inunda el paquete a todos los puertos del switch. Adicionalmente, el controlador puede instruir al switch a encapsular todos los paquetes de un flujo y enviarlos al controlador. También es posible descartar el paquete, lo cual permite, implementar servicios de control de acceso a la red con OpenFlow. Otro tipo de acción permitida es la modificación de campos de cabecera de los paquetes, como las direcciones IP origen y/o destino

del paquete y el VLAN ID, entre otros [59]. El controlador también se encuentra en la capacidad de consultar estadísticas, como el número de entradas activas y el número de paquetes procesados en las tablas de flujos de los switches. Adicionalmente un controlador puede recopilar estadísticas de puertos y colas y dirigir acciones como encolar paquetes a una cola en particular, y soportar funcionalidades QoS básicas utilizando mecanismos de colas.

### 5.6.2 OpenFlow Versión 1.1.0

La especificación OpenFlow 1.1.0 [55] incorpora cambios fundamentales en la arquitectura del switch OpenFlow, incluyendo el soporte de múltiples tablas de flujos y el soporte de etiquetas MPLS.

La especificación OpenFlow 1.1.0, incluye los siguientes cambios en la arquitectura del switch OpenFlow:

- Cambia la nomenclatura de los campos OpenFlow de acuerdo al siguiente esquema: los campos *Header Fields* cambian de nombre a *Match Fields* y los campos *Actions* cambian a *Instructions*.
- Incorpora los campos: *Metadata*, *MPLS label* y *MPLS EXP traffic class*.
- Incorpora múltiples tablas de flujos e implementa un mecanismo *pipeline* OpenFlow, en el cual un paquete atraviesa varias tablas de flujos en serie y recibe un procesamiento particular en cada tabla.
- Incluye una tabla de grupos para brindar funcionalidad común a un conjunto de paquetes.
- Permite disminuir el campo TTL (Time-To-Live) en la cabecera IP.
- Incrementa la cantidad de estadísticas del switch, incluyendo estadísticas para las tablas de grupos.

#### Componentes de un Switch OpenFlow 1.1.0

Un switch OpenFlow 1.1.0 [55] contiene los siguientes componentes:

- Una tabla de flujos o un conjunto de Tablas de flujos que conforman un *pipeline*, en el cual un flujo de datos pasa por una serie de procesos en secuencia.
- Un canal OpenFlow que conecta el switch al controlador a través de una conexión segura, como TLS [57] (Transport Layer Security) o a través de una conexión TCP plana, para permitir el intercambio de comandos y paquetes entre el controlador y los dispositivos de red.
- El protocolo OpenFlow, que permite el intercambio de mensajes y paquetes entre el controlador y los switches OpenFlow.
- Una tabla de grupos que permite aplicar tratamiento común a un conjunto de paquetes, como la agregación de tráfico o el reenvío a un próximo salto común.

En la Figura 5.7 (tomada de [55]) se muestran los componentes de un switch OpenFlow bajo la especificación 1.1.0.

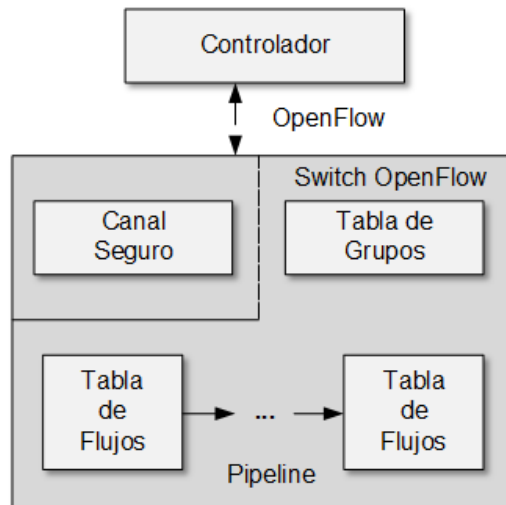


Figura 5.7: Arquitectura de un Switch OpenFlow 1.1.0

### Tabla de Flujos OpenFlow 1.1.0

Las tablas de flujos de la especificación OpenFlow 1.1.0 constan de los siguientes campos: *Match Fields*, *Counters* e *Instructions* (ver Tabla 5.4).

| Match Fields | Counters | Instructions |
|--------------|----------|--------------|
|              |          |              |

Tabla 5.4: Tabla de Flujos de un Switch OpenFlow 1.1.0

- **Match Fields:** Permiten reconocer paquetes. Consiste de puertos de ingreso, campos de cabecera de paquetes y opcionalmente de metadatos especificados por una tabla previa en el *pipeline* OpenFlow.
- **Counters:** Permiten obtener estadísticas de los paquetes concordantes con las entradas de las tablas de flujos.
- **Instructions:** Permiten modificar el *action set* o el procesamiento del pipeline. El *action set* es un conjunto de acciones que se acumulan durante la estadía de un paquete en el pipeline OpenFlow y que se ejecutan al salir del mismo.

### Instrucciones OpenFlow 1.1.0

Las instrucciones soportadas en la especificación OpenFlow 1.1.0 se presentan en la Tabla 5.5.

| Instrucciones  | Argumentos    | Semántica                                                                        |
|----------------|---------------|----------------------------------------------------------------------------------|
| Apply-Actions  | Action(s)     | Aplica acciones inmediatamente a los paquetes sin añadirlas al <i>action set</i> |
| Write-Actions  | Action(s)     | Agrega acciones al <i>action set</i>                                             |
| Clear-Actions  | -             | limpia el <i>action set</i>                                                      |
| Write-Metadata | Metadata mask | Actualiza el campo <i>Metadata</i> y su máscara                                  |
| Goto-Table     | Table ID      | Reenvía el paquete a la tabla especificada en el argumento <i>Table ID</i>       |

Tabla 5.5: Lista de Instrucciones OpenFlow 1.1.0

### Campos Match OpenFlow 1.1.0

La especificación OpenFlow 1.1.0 utiliza 15 campos en la cabecera de los paquetes que ingresan al switch para reconocer una entrada en la tabla de flujos (ver Tabla 5.6). Los siguientes campos amplían la cantidad de campos de la especificación 1.0.0 anterior:

- **Metadata:** Valor de registro que se utiliza para llevar información de una tabla a la siguiente en un pipeline OpenFlow.
- **MPLS label:** Identificador de etiqueta MPLS de 20 bits utilizado para soportar el protocolo MPLS. Se reconoce la etiqueta MPLS más externa de una pila MPLS.
- **MPLS EXP traffic class:** Identificador de la clase de tráfico MPLS utilizada para implementar mecanismos de QoS MPLS. Se reconoce la etiqueta MPLS más externa de una pila MPLS.

|              |          |           |           |            |         |               |            |                    |          |          |                         |               |                                     |                                     |
|--------------|----------|-----------|-----------|------------|---------|---------------|------------|--------------------|----------|----------|-------------------------|---------------|-------------------------------------|-------------------------------------|
| Ingress Port | Metadata | Ether src | Ether dst | Ether type | VLAN ID | VLAN priority | MPLS label | MPLS traffic class | IPv4 src | IPv4 dst | IPv4 Proto / ARP opcode | IPv4 ToS bits | TCP/UDP/ SCTP src port<br>ICMP Type | TCP/UDP/ SCTP dst port<br>ICMP Code |
|--------------|----------|-----------|-----------|------------|---------|---------------|------------|--------------------|----------|----------|-------------------------|---------------|-------------------------------------|-------------------------------------|

Tabla 5.6: Campos Match de un Switch OpenFlow 1.1.0

### Funcionamiento del Plano de Datos OpenFlow 1.1.0

En la Figura 5.8 (tomada de [55]) se muestra el plano de datos de un switch OpenFlow 1.1.0.

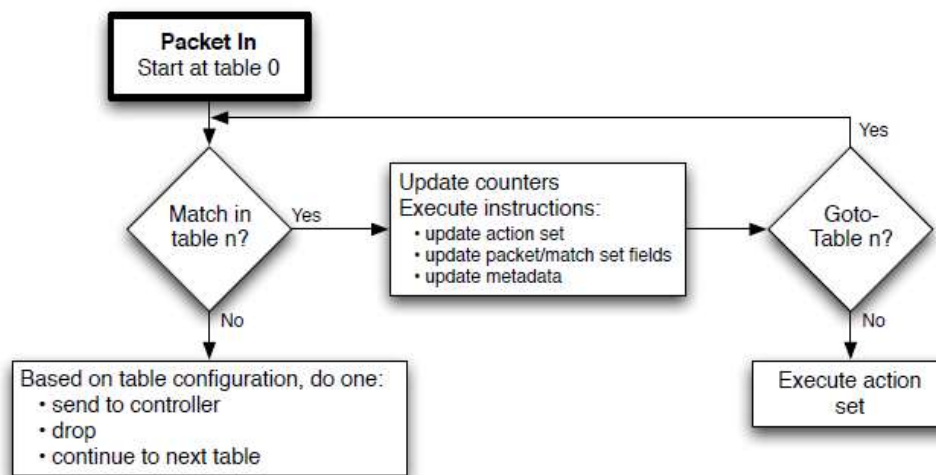


Figura 5.8: Procesamiento de Paquetes en un Switch OpenFlow 1.1.0

Cuando un paquete ingresa al switch, se inicia una consulta en las entradas de la *Tabla 0* (primera tabla de flujos del *pipeline*) y pueden ocurrir 2 casos: (1) que no exista una entrada en la tabla de flujos que concuerde con los campos de cabecera del paquete y (2) que exista una concordancia. En el primer caso se pueden efectuar tres tipos de acciones: (1) encapsular y enviar el paquete al controlador, (2) descartar el paquete o (3) enviar el paquete a la próxima tabla. En el segundo caso se ejecutan las instrucciones asociadas a la entrada de flujos, se actualizan los contadores y el paquete es procesado por el *pipeline* OpenFlow.

### Pipeline OpenFlow

La especificación OpenFlow 1.1.0 introduce el concepto de *pipeline* OpenFlow en los switches OpenFlow el cual consta de múltiples tablas, donde cada tabla contiene múltiples entradas de flujos. El procesamiento del pipeline define como interactúan los paquetes con estas tablas (ver Figura 5.9).

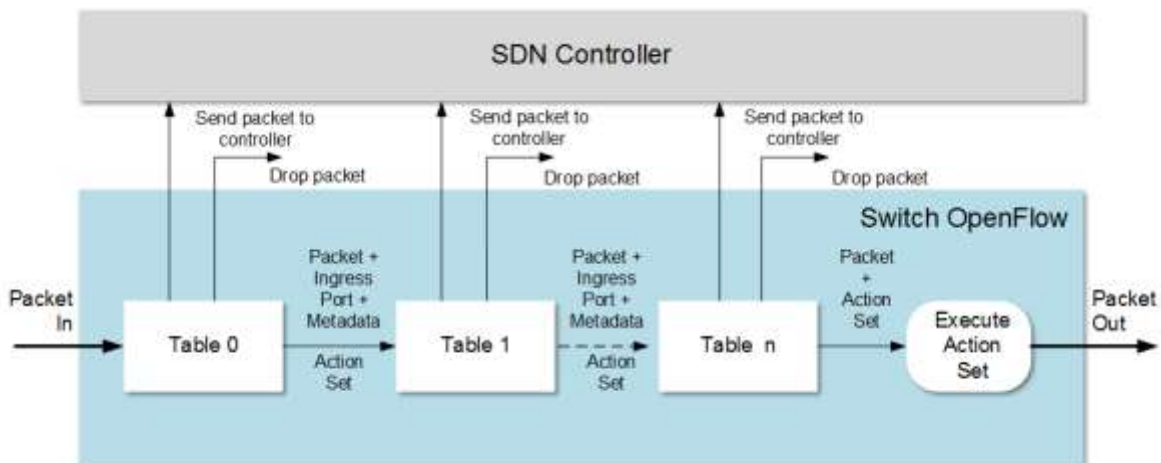


Figura 5.9: Pipeline OpenFlow 1.1.0

Las tablas de flujos del pipeline se encuentran enumeradas en orden secuencial partiendo desde 0. El procesamiento del pipeline comienza en la tabla 0 donde se hace una consulta en la tabla de flujos en base a un orden de prioridad. Si no existe concordancia en ninguna entrada y si no existe una entrada *table-miss*, el paquete se descarta. Si existe una concordancia solamente en la entrada *table-miss*, entonces esa entrada especifica alguna de las siguientes acciones:

- Enviar el paquete al controlador. Esta acción le permite al controlador definir un nuevo flujo para este tipo de paquetes, o descartar el paquete.
- Descartar el paquete.
- Dirigir el paquete a la tabla de flujos siguiente en el pipeline.

Si existe una concordancia en una o más entradas distintas a la entrada *table-miss*, se selecciona la concordancia con la entrada cuya prioridad sea más alta.

Se pueden ejecutar las siguientes acciones:

- Actualizar contadores asociados con la entrada.

- b) Ejecutar instrucciones asociadas con la entrada. Estas instrucciones pudieran incluir actualizar el *action-set*, actualizar el campo *metadata* y ejecutar acciones.
- c) Reenviar el paquete y el metadatos a la tabla de flujos siguiente del pipeline, o a la tabla de grupos, o a un puerto de salida.

Al momento de que el paquete sea dirigido finalmente a un puerto de salida, se ejecuta el conjunto de acciones acumuladas en el *action set* y el paquete es encolado para salir del switch.

### Tabla de Grupos

El segundo cambio importante en la especificación OpenFlow 1.1.0 es la incorporación de la tabla de grupos. La tabla de grupos permite aplicar acciones a un conjunto de flujos, como el reenvío multicamino o la agregación de enlaces. Los campos de la tabla de grupos se muestran en la Tabla 5.7.

| Group Identifier | Group Type | Counters | Action Buckets |
|------------------|------------|----------|----------------|
|                  |            |          |                |

Tabla 5.7: Tabla de Grupos de un Switch OpenFlow 1.1.0

- **Group Identifier:** Identifica un grupo dado.
- **Group type:** Determina la semántica de un grupo.
- **Counters:** Permite recopilar estadísticas sobre los paquetes procesados por el grupo.
- **Action buckets:** Presenta una lista ordenada de cubetas de acciones.

### Tipos de Grupos

Se encuentran definidos los siguientes tipos de grupos (*group types*):

- **All:** Ejecuta todas las cubetas en el grupo. Cada paquete es clonado para cada cubeta y el paquete es procesado por cada cubeta en el grupo. Se utiliza para reenvíos de tipo broadcast y multicast.
- **Select:** Ejecuta una cubeta seleccionada en el grupo. Los paquetes se envían a una sola cubeta en el grupo, basado en un algoritmo de selección ejecutado en el switch. Por ejemplo, una función de *hash* de alguna tupla configurada por el usuario, o un esquema *round robin*.
- **Indirect:** Ejecuta la cubeta definida en este grupo. Permite que múltiples flujos o grupos apunten a un identificador de grupo común. Por ejemplo, indicar los próximos saltos para el reenvío IP. El tipo de grupo *indirect* es equivalente al tipo de grupo *all* con una sola cubeta.
- **Fast failover:** Ejecuta las acciones de la primera cubeta activa. Cada cubeta está asociada con un puerto o un grupo de puertos específicos que determina su estado. Si no existen cubetas activas, los paquetes serán descartados. Puede ser utilizado para implementar mecanismos de recuperación a caminos de respaldo. Por ejemplo, se puede configurar una tabla de grupos donde una cubeta de acciones indique una acción “enviar los paquetes al puerto 3” y una segunda cubeta con la acción “enviar los

paquetes al puerto 4”, y así, implementar un mecanismo de recuperación en caso de fallas. En este caso, si el puerto 3 está arriba, los paquetes pertenecientes a este grupo son enviados al puerto 3, en caso contrario, los paquetes son enviados al puerto 4. Con este mecanismo, se puede implementar un mecanismo de re-enrutamiento rápido del tráfico en caso de fallas, sin requerir, una intermediación directa del controlador OpenFlow.

La especificación OpenFlow 1.1.0 también reconoce etiquetas y clases de tráfico MPLS, permitiendo ejecutar acciones MPLS.

### 5.6.3 OpenFlow Versión 1.2

La especificación del protocolo OpenFlow 1.2 [60] describe los formatos y protocolos mediante los cuales un switch OpenFlow recibe, reacciona y responde a mensajes de controladores OpenFlow. Esta especificación ofrece las siguientes mejoras: (1) añade soporte para el protocolo IPv6, (2) permite el soporte de paquetes extendidos y (3) soporta controladores redundantes.

#### Soporte IPv6

La especificación OpenFlow 1.2 ofrece soporte IPv6 con el apoyo de la concordancia extensible y se pueden reconocer los siguientes campos: IPv6 *source address*, IPv6 *destination address*, *protocol number*, *traffic class*, ICMPv6 *type*, ICMPv6 *code*, campos de cabecera IPv6 *neighbor discovery*, y etiquetas de flujos (*flow labels*) IPv6.

#### Soporte de Paquetes Extendidos

Con la especificación 1.2 se puede extender la concordancia de paquetes del protocolo OpenFlow, soportando capacidades de análisis de campos de paquetes adicionales, a través de una estructura TLV (Type-Length-Value), referenciada como OXM (OpenFlow Extensible Match).

#### Controladores Redundantes

A partir de la especificación OpenFlow 1.2.0 los switches se pueden conectar a varios controladores OpenFlow a través de varios canales OpenFlow y la red puede continuar operando en modo OpenFlow en caso de falla del controlador o de su enlace de comunicación. Los controladores en un esquema de redundancia pueden manejar un rol *maestro* o *esclavo*. El rol *maestro* permite tener control completo de los switches bajo su dominio administrativo incluyendo la modificación del plano de datos de los switches, mientras que el rol *esclavo* tiene funcionalidades limitadas y solo puede recibir notificaciones de los switches. Los controladores emplean un mecanismo de transferencia de mando *handover* para facilitar la recuperación rápida de fallas y permitir el balanceo de cargas. Un controlador esclavo puede ser promovido al rol de maestro y un controlador maestro puede asumir un nuevo rol de esclavo en un momento determinado, permitiendo implementar mecanismos de recuperación de controladores. En la Figura 5.10 se muestra la arquitectura de un switch OpenFlow 1.2.

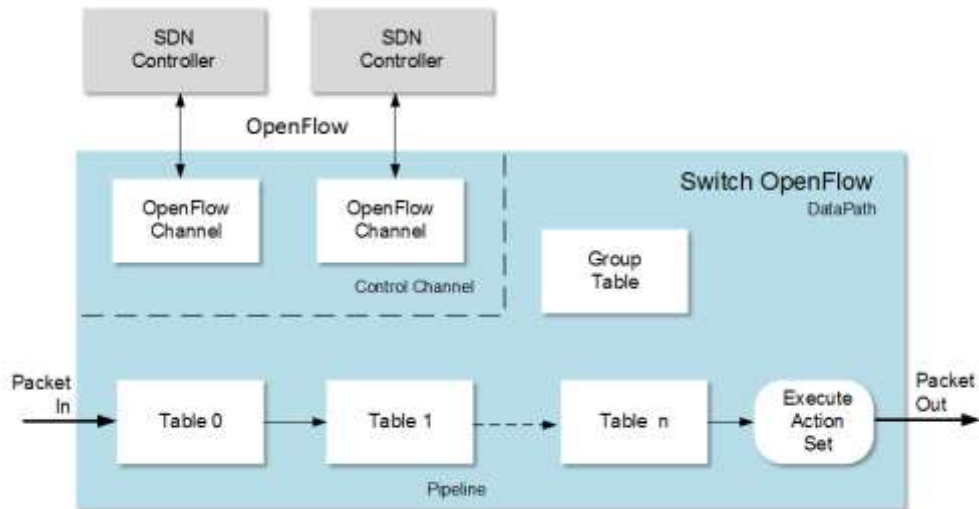


Figura 5.10: Arquitectura de un Switch OpenFlow 1.2.0

En este escenario se muestran 2 controladores que se comunican con el switch a través de dos canales de control OpenFlow.

#### 5.6.4 OpenFlow Versión 1.3.0

La especificación OpenFlow 1.3.0 [61] introduce nuevas funcionalidades para el soporte de funciones administrativas OAM (Operations Administration and Management) e incorpora la tabla especial *Meter* al plano de datos del switch (ver Figura 5.11).

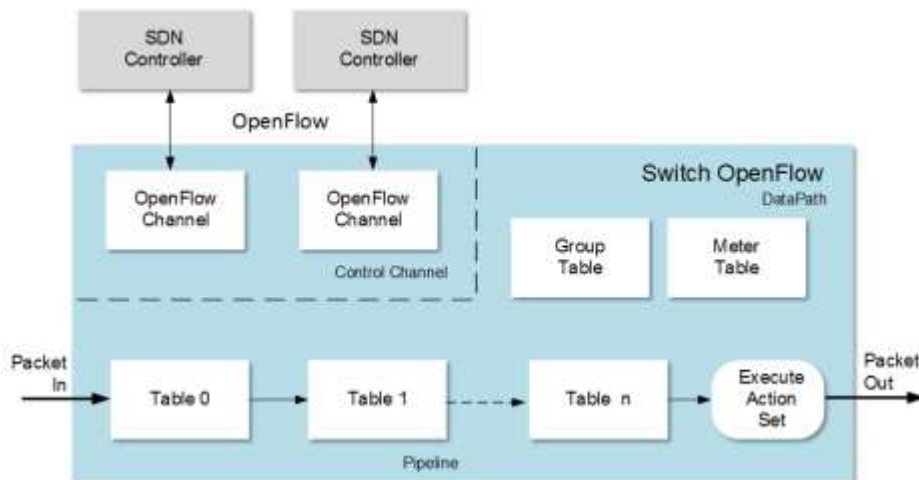


Figura 5.11: Arquitectura de un Switch OpenFlow 1.3.0

#### Tabla de Medidores (Meter)

Un switch OpenFlow 1.3.0 contiene una tabla de medidores asociados a los flujos. Por ejemplo, la tasa de bytes o paquetes correspondientes a un flujo particular. Las mediciones permiten implementar operaciones QoS simples, como limitar la tasa de tráfico, y pueden ser combinadas con contadores de colas por puertos para implementar estructuras QoS más complejas.

La tabla de medidores consta de tres campos (ver Tabla 5.8).

| Meter Identifier | Meter Bands | Counters |
|------------------|-------------|----------|
|                  |             |          |

Tabla 5.8: Campos de la Tabla Meter de un Switch OpenFlow 1.3.0

- **Meter Identifier:** Entero de 32 bits sin signo para identificar el medidor.
- **Meter Bands:** Lista ordenada de bandas de medición. Cada banda especifica la tasa de la banda y la manera de procesar el paquete.
- **Counters:** Campo que se actualiza cuando los paquetes son procesados por un medidor. Una medición está directamente asociada a una entrada de la tabla de flujos por su identificador en el campo de identificación de la medición *Meter Identifier*. El campo de la banda de medición *Meter Bands* especifica un umbral en la cantidad de paquetes o bytes permitidos por un flujo. Se puede utilizar para limitar la tasa de un flujo de tráfico descartando los paquetes futuros de un flujo cuando se superan los valores de la banda. El campo *Meter Band* también se puede utilizar para modificar el campo DS (Differentiated Service) en lugar de descartar los paquetes, permitiendo implementar funcionalidades QoS más avanzadas.

### Controladores Auxiliares

Otra mejora importante de la especificación OpenFlow 1.3.0 es el soporte extendido de múltiples controladores auxiliares. Se pueden utilizar conexiones auxiliares arbitrarias que complementen la conexión con el controlador maestro y los switches, permitiendo implementar servicios de balanceo de carga. La especificación 1.3.0 permite el filtrado de eventos por conexión, lo cual permite a los controladores suscribirse solamente a tipos de mensajes de interés. Por ejemplo, un controlador responsable de recopilar estadísticas puede asumir el rol de un controlador auxiliar y suscribirse solamente a eventos de estadísticas generados por los switches.

### Extensiones IPv6

La especificación OpenFlow 1.3.0 soporta las siguientes extensiones de la cabecera IPv6: Cabecera IPv6 ESP (Encrypted Security Payload), IPv6 Authentication Header, y Hop-by-Hop IPv6.

## 5.6.5 OpenFlow versión 1.4.0

La especificación OpenFlow 1.4.0 [62] añade estructuras TLV para puertos, tablas y colas. Adicionalmente se provee el soporte de puertos ópticos y se pueden enviar mensajes de control en lotes en un solo mensaje.

## 5.6.6 OpenFlow versión 1.5.0

La especificación OpenFlow 1.5.0 [63] provee las siguientes funcionalidades:

- **Tablas de Egreso:** En las versiones previas de la especificación, todo el procesamiento se hacía en el contexto del puerto de entrada. La especificación 1.5 introduce el mecanismo de Tablas de Egreso permitiendo que el procesamiento se realice en el contexto de puertos de salida. Los paquetes que son enviados a un puerto de salida, son procesados por la primera tabla de egreso (ver Figura 5.12).

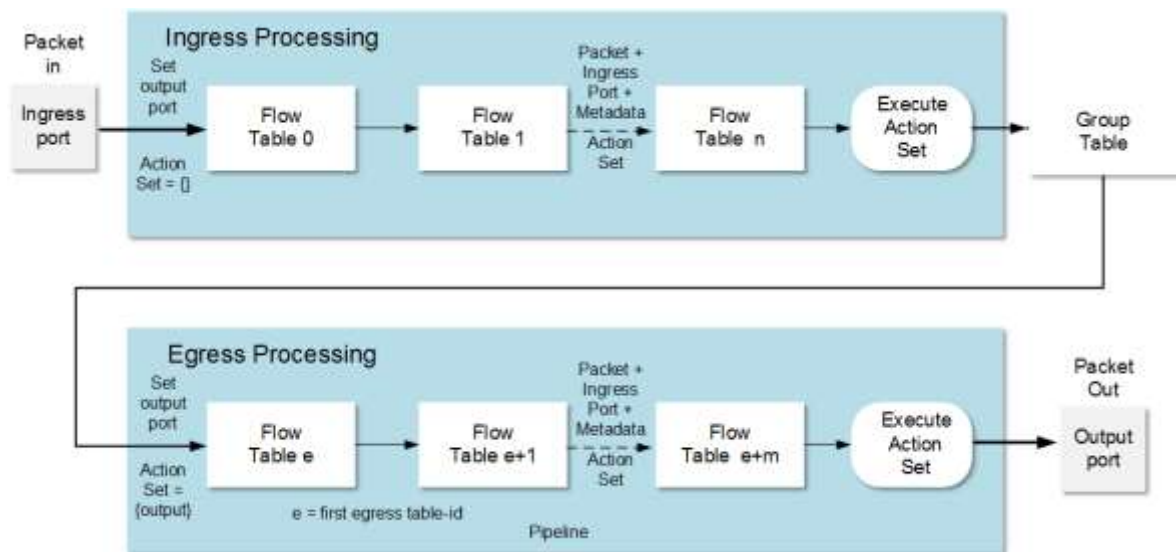


Figura 5.12: Tablas de Egreso de un Switch OpenFlow 1.5.0

- **Pipeline con Conciencia de Tipo de Paquete:** En las versiones previas de la especificación, todos los paquetes debían ser Ethernet. La versión 1.5 introduce un pipeline con conciencia de paquetes, permitiendo el procesamiento de otros tipos de paquetes, como IP o PPP entre otros.
- **Estadísticas de Entradas de Flujos Extensibles:** Las versiones previas de la especificación utilizaban una estructura fija para las estadísticas de las entradas de flujo. La versión 1.5 introduce una codificación flexible, OXS (OpenFlow eXtensible Statistics), para codificar estadísticas de entradas de flujos arbitrarias.
- **Activador de Estadísticas de Entradas de Flujos:** El sondeo de estadísticas de entradas de flujo puede inducir una alta sobrecarga y utilización para un switch. Un nuevo mecanismo de activación de estadísticas permite que las estadísticas sean enviadas automáticamente al controlador basado en varios umbrales.
- **Reconocimiento de Banderas TCP:** Se añade un campo *OXM* que permite reconocer bits de banderas en la cabecera TCP, como *SYN*, *ACK* y *FIN*, los cuales pueden ser utilizados para detectar el inicio y el fin de conexiones TCP.
- **Mensajes Agrupados:** Las versiones previas de la especificación OpenFlow introdujeron el concepto de mensajes *bundle messages*. Un *bundle* es una secuencia de solicitudes de modificación OpenFlow desde el controlador que son aplicadas como una sola operación OpenFlow. La especificación 1.5 extiende la funcionalidad *bundle* permitiendo:

- **Bundles Planificados:** Un mensaje *bundle* pudiera incluir un tiempo de ejecución, especificando cuando el switch debe ejecutar la acción.
- **Solicitud de Funcionalidades Bundle:** Permite al controlador consultar a un switch sobre el tipo de bundle soportado que incluye: atómico, ordenado y planificado.
- **Estatus de la Conexión del Controlador:** Permite al controlador conocer el estatus de todas las conexiones desde el switch a los controladores, lo cual le permite a un controlador detectar particiones en la red de control, o monitorear el estatus de otros controladores.

### 5.6.7 Cuadro Comparativo de las Especificaciones OpenFlow

En la Tabla 5.9 se muestra la evolución en funcionalidades de la especificación OpenFlow a través de su historia, desde la publicación 1.0.0 en diciembre del 2009 hasta la especificación 1.5.0 actual publicada en diciembre del 2015.

| Capacidad                                                                   | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 |
|-----------------------------------------------------------------------------|-----|-----|-----|-----|-----|-----|
| Comunicación entre el Switch y el Controlador vía TLS                       | X   |     |     |     |     |     |
| La tabla de flujos consiste de los campos: Match Fields, Counters y Actions | X   |     |     |     |     |     |
| Tres tipos de mensajes: Controller to switch, Asynchronous y Symetric       | X   |     |     |     |     |     |
| Múltiples tablas                                                            |     | X   |     |     |     |     |
| Tabla de Grupos                                                             |     | X   |     |     |     |     |
| Etiquetas: MPLS y VLAN                                                      |     | X   |     |     |     |     |
| Puertos virtuales                                                           |     | X   |     |     |     |     |
| Falla de conexión al controlador                                            |     | X   |     |     |     |     |
| Soporte match extensible                                                    |     |     | X   |     |     |     |
| Soporte IPv6 básico                                                         |     |     | X   |     |     |     |
| Mecanismo de cambio de rol del controlador                                  |     |     | X   |     |     |     |
| Mediciones por flujo                                                        |     |     |     | X   |     |     |
| Metadatos túnel ID                                                          |     |     |     | X   |     |     |
| Soporte IPv6 extendido                                                      |     |     |     | X   |     |     |
| Soporta extensiones de protocolos vía el formato TLV                        |     |     |     |     | X   |     |
| Soporta puertos ópticos                                                     |     |     |     |     | X   |     |
| Soporte de tablas de egreso                                                 |     |     |     |     |     | X   |
| Pipeline con conciencia del tipo de paquete                                 |     |     |     |     |     | X   |
| Soporta estadísticas de entradas de flujos extensibles                      |     |     |     |     |     | X   |
| Reconoce banderas TCP                                                       |     |     |     |     |     | X   |

Tabla 5.9: Cuadro Comparativo de Especificaciones OpenFlow

## 6. Contexto Actual de SDN

OpenFlow es un proyecto de código abierto que ha sido el resultado de 7 años de investigación y que tuvo sus inicios gracias al trabajo en conjunto entre la Universidades de Stanford y California en Berkeley, Estados Unidos. En la siguiente sección se enumeran los eventos más importantes alrededor de SDN y OpenFlow hasta la fecha:

- 2008, Disposición del controlador NOX bajo licencia GPL.
- 2008, Demostración del primer switch basado en hardware con soporte OpenFlow por el fabricante HP en el evento ACM SIGCOMM 2008.
- Diciembre 2009, se publica la especificación OpenFlow 1.0.0.
- Marzo 2011, Deutsche Telekom, Facebook, Google, Microsoft, Verizon y Yahoo! dan inicio a la fundación de redes abiertas ONF<sup>19</sup> (Open Networking Foundation) para encargarse de promover la adopción de SDN a través del desarrollo de estándares.
- Octubre 2011, se celebra el evento ONS (Open Networking Summit) en la Universidad de Stanford, California, Estados Unidos, con una participación de más de 200 asistentes.
- Febrero 2012, se publica la especificación OpenFlow versión 1.2.0.
- Abril 2012, se celebra la segunda conferencia ONS en California, Estados Unidos, con la participación de Google, NTT Communications, Verizon, Big Switch, Cisco, IBM, NEC, Nicira y otros jugadores SDN claves.
- Junio 2012, se publica la especificación OpenFlow ver 1.3.0.
- Abril 2013, se celebra la tercera conferencia ONS en California, Estados Unidos, con una participación de 3200 asistentes y el auspicio de Brocade, Cisco, HP, Huawei, NEC y NTT Communications.
- La ONF alcanza el soporte de 100 miembros y cuenta con la participación de fabricantes, desarrolladores y empresas de servicios como Google y Facebook.
- Octubre 2012, la ONF lanza el segundo evento de pruebas de compatibilidad *plugfest* donde se realizaron pruebas de conformidad, interoperabilidad, desempeño y pruebas de concepto de controladores y switches SDN en el laboratorio InCNTRE (Indiana Center for Network Translational Research and Education) de la Universidad de Indiana, Estado Unidos.
- Primavera de 2012, lanzamiento de la primera conferencia SDN global, SDN & OpenFlow World Congress, co-patrocinado por la ONF y endorsado por el Instituto Europeo de Estandares de Telecomunicaciones ETSI.
- Abril 2013, la fundación Linux anuncia el lanzamiento del proyecto colaborativo de código abierto ODL (OpenDaylight). La meta del proyecto es la adopción de SDN y crear las bases para NFV (Network Function Virtualization). El software está escrito en Java y el proyecto fue promocionado por Arista Networks, Big Switch Networks, Brocade, Cisco,

---

<sup>19</sup> <https://www.opennetworking.org/about/onf-overview>

Citrix, Ericsson, HP, IBM, Juniper Networks, Microsoft, NEC, Nuage Networks, PLUMgrid, Red Hat y VMware.

- Octubre 2013, se publica la especificación OpenFlow ver 1.4.0.
- Febrero 2014, se anuncia la primera versión de ODL, denominada Hydrogen.
- Marzo 2014, se celebra la cuarta conferencia ONS en Santa Clara, Estados Unidos.
- Octubre 2014, se lanza la segunda versión de ODL, denominada Helium.
- Junio 2015, se lanza la tercera versión de ODL, denominada Lithium.
- Diciembre 2015, se publica la especificación OpenFlow ver 1.5.0.
- Febrero 2016, se lanza la cuarta versión de ODL, denominada Beryllium.
- Mayo 2016, se celebra el octavo evento de interoperabilidad AppFest organizado por ONF para enfocarse en la interoperabilidad de aplicaciones SDN con controladores y switches OpenFlow de múltiples vendedores.
- Octubre 2016, SDN & OpenFlow World Congress, The Hague, Netherlands.
- Noviembre (7-9) 2016, 2016 IEEE NFV-SDN. Palo Alto, CA.
- Noviembre (7-10) 2016, MEF16. Baltimore, DC, USA.
- Mayo 2017, se lanza la sexta versión de ODL, denominada Carbon<sup>20</sup>.
- Junio 2017, se anuncia el evento OPNFV Summit 2017<sup>21</sup>.
- En resumen en los últimos años se ha visto un gran interés por parte de los investigadores y fabricantes de equipos de red por desarrollar productos y aplicaciones compatibles con la especificación OpenFlow con la meta de acelerar el desarrollo de nuevas tecnologías que superen las limitantes de las redes tradicionales actuales.

En este capítulo, se mencionan las principales herramientas de simulación SDN, los switches y controladores SDN actuales y algunos casos de uso de esta importante tecnología.

## **6.1 Herramientas de Simulación SDN**

En esta sección, se muestran las principales herramientas de simulación SDN disponibles.

### **6.1.1 ns-3**

ns-3<sup>22</sup> es un simulador de red de eventos discretos de código abierto utilizado en la investigación y aprendizaje de tecnologías de redes. Corre bajo licencia pública GNU GPLv2 y está disponible al público para investigación, desarrollo y uso. La herramienta está construida completamente en lenguaje C++ y puede contar con envoltorios Python para correr códigos de usuarios. La herramienta funciona descargando y compilando código fuente para crear una librería y compilando y enlazando el código de los usuarios con la librería.

---

<sup>20</sup> <https://www.opendaylight.org/software/downloads/carbon>

<sup>21</sup> <https://www.opendaylight.org/events/2017-06-12-000000-2017-06-15-000000/opnfv-summit-2017>

<sup>22</sup> <http://www.nsnam.org>

### **Soporte OpenFlow v0.8.9 en ns-3**

En la actualidad ns-3 puede utilizar switches con especificación OpenFlow v0.8.9 [64], en conformidad con la documentación de las APIs públicas de la herramienta ns-3<sup>23</sup>. Los switches OpenFlow son configurables a través del API OpenFlow y además cuentan con una extensión MPLS para soportar SLAs (Service Level Agreement). La distribución de la implementación del software se refiere como OFSID en la documentación ns-3. Existe además una OFSID desarrollada por un grupo de investigadores de Ericsson para añadir capacidades MPLS.

### **Soporte OpenFlow 1.3 en ns-3**

En paralelo al soporte de la especificación OpenFlow 0.8.9 se ha desarrollado el módulo OFSwitch13 con soporte del pipeline OpenFlow 1.3 conformado por múltiples tablas de flujos, tablas de grupos y tablas de medición entre otras características.

### **Limitaciones de OpenFlow en ns-3**

Las siguientes funcionalidades de OpenFlow no se encuentran soportadas actualmente en ns-3:

- Múltiples Controladores: Cada switch puede ser gestionado por un solo controlador.
- Conexiones auxiliares: Solamente es posible una conexión entre un switch y un controlador.
- Encriptación del canal OpenFlow: Solamente soporta el protocolo TCP para la conexión OpenFlow.
- Control en-banda: El controlador solo puede gestionar los switches sobre una conexión fuera de banda.

### **6.1.2 Mininet**

Mininet [65] es un emulador de red liviano que permite crear una red OpenFlow de hosts, switches, controladores y enlaces virtuales sobre una laptop o una PC de escritorio. A través de Mininet se pueden efectuar actividades de investigación, desarrollo, aprendizaje, prototipos, depuración y pruebas de concepto relacionadas con un despliegue SDN OpenFlow de manera práctica y económica, en una etapa temprana de desarrollo, antes de migrar hacia una red de producción real. Mininet sigue un enfoque de virtualización liviano utilizando funcionalidades de virtualización a nivel del sistema operativo basado en contenedores Linux. Un contenedor le permite a un grupo de procesos, incluyendo los procesos de los hosts virtuales y los procesos de las aplicaciones que corren sobre el contenedor, tener una visión independiente de los recursos del sistema que incluyen: identificación de procesos, sistemas de archivos e interfaces de red, y compartir el kernel con otros contenedores. Mininet ofrece una escalabilidad de hasta 4096

---

<sup>23</sup> [https://www.nsnam.org/doxygen/classns3\\_1\\_1\\_open\\_flow\\_switch\\_net\\_device.html#details](https://www.nsnam.org/doxygen/classns3_1_1_open_flow_switch_net_device.html#details)

hosts <sup>24</sup>. Se basa en la virtualización para crear host emulados y utiliza switches de software Open vSwitch<sup>25</sup> para crear switches OpenFlow.

Los enlaces que conectan el switch OpenFlow a los hosts emulados se implementan utilizando el mecanismo de pares de Ethernet virtuales provistos por el Kernel de Linux. Los hosts emulados se comportan como VMs, lo que les permite correr aplicaciones reales listas para intercambiar información en la red simulada. Los controladores OpenFlow reales, también constituyen una aplicación real y pueden correr en un host emulado para establecer conexiones TCP a los switches OpenFlow bajo su dominio.

Actualmente se encuentra disponible las imágenes pre-construidas VM que incluyen Mininet 2.2.1 en Ubuntu 14.04 LTS para su descarga y utilización en un sistema de virtualización X86<sup>26</sup>, como VirtualBox, o Hyper-V.

### Creación de una Red de Ejemplo en Mininet

Para crear una red de switches OpenFlow en Mininet se utiliza la herramienta CLI mn. En la Figura 6.1 se muestra un ejemplo de emulación de una red OpenFlow a través de Mininet.

```
mn --switch ovsk --controller nox --topo tree, depth=2, fanout=8 --test pingAll
```

Figura 6.1: Ejemplo de Creación de Red Mininet

En el ejemplo se crea una red OpenFlow de switches de kernel Open vSwitch que sigue una topología de árbol con profundidad 2 y fanout 8 equivalente a 9 switches y 64 hosts bajo el mando de un controlador NOX seguido por una prueba que verifica la conectividad entre cada par de nodos:

### Alcance de Mininet

- Permite efectuar pruebas de red de manera simple y económica para desarrollar aplicaciones OpenFlow.
- A través de Mininet una red completa puede ser empaquetada como una VM, de manera que otros pueden descargarla, correrla, examinarla y modificarla.
- Se pueden crear pruebas de nuevos servicios de red o una nueva arquitectura de red completa, hacer pruebas con topologías grandes con tráfico de aplicación, y desplegar el mismo código y los scripts de las pruebas en una red de producción.
- Permite probar topologías complejas, sin tener que cablear o cambiar una red física existente.

---

<sup>24</sup> <http://mininet.org/overview>

<sup>25</sup> <http://openvswitch.org>

<sup>26</sup> <https://github.com/mininet/mininet/wiki/Mininet-VM-Images>

- Incluye una línea de comandos que tiene conciencia OpenFlow y conciencia de la topología.
- Soporta topologías básicas predefinidas y topologías arbitrarias personalizables.
- Se utiliza directamente y de manera interactiva sin necesidad de programación.
- Provee un API Python extensible para crear y experimentar con nuevas redes y topologías.
- Corre código real incluyendo aplicaciones de red Unix/Linux, kernel de red Linux, y stack de red Linux, incluyendo extensiones de kernel que sean compatibles con espacios de nombres de red, permitiendo que el código que un experimentador desarrolle y pruebe en Mininet pueda moverse a switches de hardware real.
- Provee un *workflow* de prototipo rápido para crear, interactuar, personalizar y compartir redes definidas por software, así como, un camino simple para ejecutar las configuraciones en hardware real.
- Combina una virtualización liviana con una línea de comandos y una API extensible.

### **Limitaciones de Mininet**

- Actualmente no permite correr aplicaciones con otros sistemas operativos distintos a Linux.
- No puede exceder las capacidades de CPU y ancho de banda de un solo servidor. La mayor limitante de la herramienta es la falla para proveer fidelidad de desempeño cuando los recursos requeridos por la red emulada exceden la capacidad del CPU disponible o el ancho de banda de la máquina física, particularmente bajo cargas de trabajo grandes y cuando el número de eventos activos concurrentes es mayor que el número de núcleos paralelos. En estos casos, la herramienta no garantiza que un host emulado sea planificado oportunamente por el sistema operativo y no garantiza que todos los switches OpenFlow basados en software reenvíen los paquetes a la misma tasa. Esto significa, que bajo grandes cargas de trabajo y bajo recursos de cómputo limitado la tasa de reenvío de paquetes de un switch OpenFlow sea impredecible y varía en cada corrida de un experimento y depende de factores como: (1) la velocidad del CPU, (2) la cantidad de memoria principal disponible, (3) el número de hosts, (4) el número de switches emulados, y (5) la carga del sistema actual donde corre la simulación.

### **6.1.3 VT-Mininet (Virtual Time Mininet)**

VT-Mininet [66] es un emulador SDN OpenFlow basado en Mininet que incluye modificaciones al kernel de Linux para proveer un sistema de tiempo virtual a los contenedores y mejorar la fidelidad de desempeño de la herramienta Mininet. Sigue el enfoque de dilación de tiempo para construir un sistema de tiempo virtual liviano en el contenedor Linux el cual se integra a Mininet y es transparente a las aplicaciones.

Bajo el concepto de tiempo virtual, los contenedores perciben el tiempo virtual como si estuvieran corriendo independientemente y concurrentemente, por lo cual las interacciones entre los contenedores y los sistemas físicos son escaladas artificialmente, aparentando que la red es mucho más rápida desde el punto de vista de las aplicaciones dentro de los contenedores que la realidad. El enfoque de dilación de tiempo está basado en un factor de dilación de tiempo TDF que se define como la razón entre la tasa del reloj real del sistema y la percepción del tiempo del host emulado. Un TDF de 10 significa que para cada 10 segundos de tiempo real, todas las aplicaciones que corren sobre el host emulado con dilación de tiempo perciben el avance del tiempo como un segundo. En este sentido, un enlace de 100 Mbps ahora aparenta ser un enlace de 1 Gbps desde el punto de vista del host emulado.

#### **Alcance de VT-Mininet**

Los autores de la herramienta [65] efectuaron pruebas experimentales y los resultados arrojados demuestran mejoras significativas en la fidelidad del desempeño en escenarios de red con cargas de trabajo grandes.

#### **6.1.4 Estinet**

Estinet [67] es un simulador y emulador de red comercial para simular redes SDN OpenFlow. Actualmente está disponible la versión Estinet 9.0<sup>27</sup>, la cual puede simular miles de switches OpenFlow y puede soportar tanto el modo simulación, como el modo emulación. En el modo simulación se puede correr el programa controlador (Ej. NOX, POX, Floodlight, OpenDaylight o Ryu) en un nodo con rol controlador en la red simulada, mientras que en el modo emulación el programa se puede ejecutar en una máquina externa al ambiente de simulación, o en un dispositivo de hardware dedicado, que puede controlar los switches simulados vía un cable Ethernet.

#### **Alcance de Estinet 9.0**

- Puede operar en modo simulación o en modo emulación.
- Simula la interacción entre un programa controlador y un switch simulado con exactitud, y rapidez.
- Soporta switches OpenFlow 1.0.0 y 1.3.4.
- Puede correr aplicaciones Linux reales en una red SDN OpenFlow simulada sin ninguna modificación.
- Está basado en una metodología de simulación de reentrada al kernel [68] que le brinda fidelidad y escalabilidad.
- Puede ser utilizado para estudiar el desempeño de una aplicación o red, como el rendimiento o el retardo extremo-a-extremo de un flujo de datos en una red OpenFlow.
- Simula con fidelidad las propiedades de los enlaces que conectan los switches OpenFlow, como el ancho de banda, el retardo, y los errores.

---

<sup>27</sup> [http://www.estinet.com/ns/?page\\_id=21140](http://www.estinet.com/ns/?page_id=21140)

- El avance del reloj de la simulación es controlado con precisión, permitiendo ofrecer resultados de simulación realísticos y repetibles.

### 6.1.5 OFNet

OFNet [70] es un nuevo emulador SDN de código abierto que ofrece funcionalidades similares al emulador de red mininet y complementa la emulación de redes con herramientas para generar tráfico, monitorear mensajes OpenFlow y evaluar el desempeño de controladores SDN. Sigue la licencia Apache v2 y ofrece mejoras importantes en términos de representación visual, monitoreo de desempeño y agilidad necesarios para la resolución de problemas en simulaciones de red OpenFlow.

#### Funcionalidades de OFNet

- OFNet incluye los siguientes servicios: (1) Depurador visual y (2) generador de tráfico y monitoreo de desempeño.  
**Depurador Visual:** Permite visualizar el comportamiento de la red OpenFlow y resolver aspectos relacionados a la fallas en la red de manera visual. Por ejemplo, se puede determinar la razón por la cual un controlador no incluye una entrada en la tabla de flujos a través de un panel de visualización acelerando el tiempo de respuesta en la resolución de problemas. Colocando en perspectiva OFNet con respecto a un ambiente Mininet, la detección de falla del funcionamiento del protocolo OpenFlow se hace de manera expedita mediante un ambiente gráfico, mientras que en un ambiente Mininet la misma operación involucra muchas corridas y trazas de seguimiento a través de la herramienta WireShark incurriendo en un gran esfuerzo operacional y en una inversión de tiempo significativa.
- **Generador de Tráfico y Monitoreo de Desempeño:** Provee un generador de tráfico realista para probar un controlador SDN mas allá de efectuar pruebas de ping [69] permitiendo evaluar los siguientes aspectos:
  - Evaluar las características de desempeño de varios controladores.
  - Determinar si un despliegue OpenFlow satisface los tiempos de respuestas de las aplicaciones de red.
  - Identificar la cantidad de ciclos de CPU consumidos por un switch OpenFlow.
  - Analizar la eficiencia del controlador en el uso de entradas TCAM en switches OpenFlow de hardware.
  - Depurar fallas de flujos en escenarios de miles de flujos activos en la red.

La herramienta incluye las siguientes métricas para reportes: Tasa de generación de flujos/segundo, tasa de fallas de flujos/segundo, latencia promedio para el establecimiento de flujos, cantidad de mensajes OpenFlow hacia el controlador/segundo, cantidad de mensajes OpenFlow desde el controlador/segundo, utilización de CPU en un switch Open vSwitch, cantidad de ausencias de flujos hacia el controlador, cantidad de

mensajes Flow\_Mod/segundo, cantidad de entradas en las tablas de flujos, cantidad promedio de entradas en las tablas de flujos, tiempo de ida y vuelta de paquetes RTT (Round Trip Time) promedio (en milisegundos), latencia de acceso a Internet (en segundos).

### **Alcance de OFNet**

- Provee un panel de visualización de la topología de red de una simulación OpenFlow que facilita el acceso a los contenidos de las tablas de flujos de los switches a través de una interfaz GUI.
- Permite visualizar el establecimiento de flujos en un controlador.
- Lleva la traza de las transacciones entre nodos controladores y switches y la representa mediante grafos. La visualización despliega el estado de reenvío antes de un evento, las transacciones del plano de control durante el evento y el estado de reenvío después del evento
- Permite correr animaciones gráficas de eventos que muestran la interacción de los nodos OpenFlow en el tiempo.
- Muestra de forma gráfica varias métricas de características de desempeño de un controlador incluyendo la cantidad de mensajes OpenFlow enviados o recibidos por un controlador, la utilización de CPU de un nodo OpenFlow o la cantidad de entradas de flujos de un switch.
- Es compatible con los principales controladores de código abierto incluyendo: Floodlight, Beacon, Opendaylight y ONOS.

### **6.1.6 Integración OpenFlow en OMNeT++**

En [71] Klein y Jarschel proponen un modelo de simulación que integra el protocolo OpenFlow versión 1.2 [60] en un framework INET [72] para OMNeT++ [73]. El modelo está conformado por un framework INET, un switch OpenFlow versión 1.2 y un controlador básico. El modelo considera un tiempo de servicio promedio de reenvío de colas en los switches de 9.8 us y un tiempo de servicio de procesamiento promedio en el controlador de 240 us.

## **6.2 Switches OpenFlow basados en Software**

En la Tabla 6.1 se muestran switches SDN basados en software disponibles.

### **6.2.1 Open vSwitch (OVS)**

Open vSwitch<sup>28</sup> es un switch de software multicapa de código abierto licenciado bajo la licencia Apache 2.0 orientado a gestionar ambientes virtualizados de gran escala. Utiliza un enfoque de controlador centralizado para conectar a switches con capacidad OpenFlow y puede utilizar interfaces de gestión adicionales, como SNMP para efectuar configuraciones. Funciona como un switch virtual, proveyendo conectividad entre VMs e interfaces físicas. Adicionalmente emula el

---

<sup>28</sup> <http://openvswitch.org/>

protocolo OpenFlow en ambientes virtualizados basados en Linux incluyendo Xen Server, KVM (Kernel Based Virtual Machine) y VirtualBox. Soporta OpenFlow y otros mecanismos de switching tradicional incluyendo 802.1Q VLAN, QoS, gestión de fallas de conectividad 802.1ag, NetFlow y técnicas de tunneling como GRE (Generic Routing Encapsulation), GRE over IPSEC, VXLAN y LISP.

| Switch                | Implementación | Versión | Descripción                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|----------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Open vSwitch          | C/Python       | v1.0    | Implementación de código abierto de un switch virtual multicapa distribuido. Puede operar como un switch de software corriendo dentro de un hipervisor, y como un stack de control para chipsets de switches de silicio. Forma parte del kernel de Linux 3.3 y los paquetes se encuentran disponibles para Ubuntu, Debian y Fedora.                   |
| Pantou/OpenWRT        | C              | v1.0    | Puerto OpenFlow para el ambiente inalámbrico OpenWRT.                                                                                                                                                                                                                                                                                                 |
| ofsoftswitch13        | C/C++          | v1.3    | Switch de software construido bajo el switch de referencia de Stanford OpenFlow 1.0 y el switch OpenFlow 1.1 de Ericsson's Traffic Lab.                                                                                                                                                                                                               |
| Indigo Virtual Switch | C              | v1.0    | Switch de software para la distribución de Linux Ubuntu 11.10/12.04 Linux compatible con el hipervisor KVM. Aprovecha el módulo de kernel ovsch para el reenvío de paquetes. Está diseñado para soportar aplicaciones de virtualización de red de gran escala y distribución a través de múltiples servidores físicos usando un controlador OpenFlow. |

Tabla 6.1: Implementaciones de Switches SDN basados en Software

### 6.2.2 Pantou/Open WRT

Pantou/Open WRT es un switch que convierte un router inalámbrico comercial o AP (Access Point) en un switch *OpenFlow-híbrido*. OpenFlow se implementa como una aplicación en el tope de OpenWrt. OpenWrt es una distribución de Linux basada en firmware implementada en dispositivos como gateways, routers y APs residenciales<sup>29</sup>. Provee un sistema de archivo completamente modificable con gestión de paquetes lo cual libera de la configuración y selección de paquetes provistas por el fabricante y permite personalizar el dispositivo a través de paquetes que se adapten a nuevas necesidades. Pantou está basado en la

<sup>29</sup> <https://openwrt.org>

liberación BackFire OpenWrt (Linux 2.6.32). El módulo OpenFlow está basado en la implementación de referencia de Stanford.

La Tabla 6.2 muestra la lista de dispositivos soportados con Pantou<sup>30</sup>.

| Dispositivo                | Chipset          | CPU    |
|----------------------------|------------------|--------|
| LinkSys WRT54GL            | Broadcom         | 200MHz |
| TP-LINK TL-WR1043ND (v1.7) | Atheros          | 400MHz |
| TP-LINK TL-WR1043ND (v1.8) | Atheros          | 400MHz |
| Generic Broadcom           | Broadcom BCM47xx |        |

Tabla 6.2: Lista de Dispositivos con Soporte Pantou

### 6.2.3 ofsoftswitch14

ofsoftswitch14<sup>31</sup> es una implementación de un switch en software en el espacio-de-usuario compatible con OpenFlow 1.3. El código está basado en la implementación ofsoftswitch1.1<sup>32</sup> de Ericsson, la cual cambia el plano de reenvío para soportar OpenFlow 1.3.

### 6.2.4 Indigo

Indigo es una implementación de código abierto que corre en switches de hardware Ethernet para correr OpenFlow a la velocidad del medio<sup>33</sup>. La versión actual está basada en la implementación de referencia de OpenFlow de Stanford y actualmente corre todas las funcionalidades de OpenFlow 1.0. La distribución de Indigo se encuentra disponible en dos formas: (1) como imagen de firmware pre-construida y (2) como una distribución de código en una VM. Indigo soporta interfaces de usuarios vía CLI a través de conexiones telnet o SSH e interfaces GUI basadas en web. Las interfaces permiten la configuración de la interfaz de control y el monitoreo de los puertos y la tabla de flujos. Los usuarios pueden descargar la versión del código del sistema IODS (Indigo Open Development System) y modificar el servidor web, la interfaz CLI, la lógica de procesamiento OpenFlow y añadir programas adicionales.

## 6.3 Switches OpenFlow Comerciales basados en Hardware

En la Tabla 6.3 se muestra una lista de switches de hardware comerciales con soporte OpenFlow. En la primera columna se indica el nombre del fabricante, en la segunda columna se muestran los modelos de switch disponibles y en la tercera columna se especifica la versión OpenFlow soportada.

<sup>30</sup> [http://archive.openflow.org/wk/index.php/Pantou:\\_OpenFlow\\_1.0\\_for\\_OpenWRT](http://archive.openflow.org/wk/index.php/Pantou:_OpenFlow_1.0_for_OpenWRT)

<sup>31</sup> <https://github.com/TrafficLab/of14softswitch>

<sup>32</sup> <https://github.com/TrafficLab/of11softswitch>

<sup>33</sup> <http://www.openflowhub.org/display/Indigo/Indigo+-+Open+Source+OpenFlow+Switches+-+First+Generation>

| Fabricante       | Modelo de Switch                                                                                             | Version      |
|------------------|--------------------------------------------------------------------------------------------------------------|--------------|
| Hewlett-Packard  | 5400, 3800, 3500, 2930, 2920                                                                                 | v1.0 - 1.3.1 |
| Hewlett-Packard  | 12900, 7900, 5400, 10500, 7500, 5900, 5700, 3800, 3500, 2920, 5500, 5130                                     | v1.3.1       |
| Brocade          | ICX7750, ICX 6450 apilado con ICX 6610                                                                       | v1.0 - 1.3   |
| Brocade          | ICX 7750, MLXe, ICx7250, ICX 6610, ICX 6450 (apilado con ICX 6610), ICX 7450                                 | v1.3         |
| Lenovo           | RackSwitch G8332, RackSwitch G8296, RackSwitch G8272, RackSwitch G8264CS, RackSwitch G8264, RackSwitch G8052 | v1.0 - 1.3.1 |
| NEC              | PF5240                                                                                                       | v1.0         |
| NEC              | PF5248                                                                                                       | v1.0 - 1.3   |
| Juniper          | EX9200 (OpenFlow-híbrido)                                                                                    | v1.0 - 1.3.1 |
| Juniper          | QFX5100 Switches (solo-OpenFlow)                                                                             | v1.0 - 1.3.1 |
| Pica8            | P-3297, P-3922, P-3930, P-5101                                                                               | v1.4         |
| Arista           | 7050QX, 7050TX, 7050SX, 7050QX232S, 7050TX2-128, 7050SX2-72Q, 7050SX2-128, 7250QX                            | v1.0         |
| Cisco            | Agente OpenFlow <i>Cisco Plug-in 2.0.2</i> para switches: 2960X/XR, 3650, 3850, 4500-X, 4500-E, Nexus 70007k | v1.0/v.1.3   |
| Cisco            | Agente OpenFlow <i>Cisco Plug-in 1.1.5</i> para switches Nexus 3000, Nexus 5000 y Nexus 9000.                | v1.0/v.1.3   |
| Dell             | S4810, S4820T, S6000, Z9000, Z9500, MXL                                                                      | v1.0         |
| Dell             | S4810, S4820T, S5000, S6000, Z9000, Z9500, and MXL                                                           | v1.0 - 1.3   |
| Dell             | S4048-ON                                                                                                     | v1.3         |
| Extreme Networks | SSA130, SSA150, SSA180, S1, S3, S4, S6, S8                                                                   | v1.3         |

Tabla 6.3: Switches OpenFlow Comerciales Basados en Hardware

### 6.3.1 Switches Bare Metal

Los switches *bare metal* son switches elaborados por fabricantes ODM (original design manufacturers) que no incluyen un sistema operativo de red precargado. Algunos ODMs disponibles en el mercado son Accton<sup>34</sup>, Alpha Networks<sup>35</sup>, Quanta Cloud Technology (QCT)<sup>36</sup>, Celestica<sup>37</sup>, Edge Core<sup>38</sup>. Los switches vienen equipados con el cargador ONIE<sup>39</sup>(Open Network Installation Environment) el cual

<sup>34</sup> <http://www.accton.com/default.asp>

<sup>35</sup> <http://www.alphanetworks.com/en>

<sup>36</sup> [http://www.quantatw.com/Quanta/english/product/subsidiary\\_qct.aspx](http://www.quantatw.com/Quanta/english/product/subsidiary_qct.aspx)

<sup>37</sup> <https://www.celestica.com/our-expertise/markets/enterprise-and-cloud-solutions>

<sup>38</sup> <http://www.edge-core.com>

<sup>39</sup> <http://onie.org>

consta de un sistema operativo pequeño, pre-instalado como firmware en switches bare metal, que permite el aprovisionamiento de sistemas operativos de red.

### 6.3.2 Switches White Box

Los switches white box son switches *bare metal* con chipsets genéricos y con un sistema operativo de red pre-instalado o de un tercero. Linux es un sistema operativo muy utilizado en estos sistemas debido a su amplia aceptación en los círculos académicos y a la gran cantidad de herramientas de código abierto disponibles en la industria. Google y Facebook utilizan switches white box en sus despliegues de red para minimizar los costos de CAPEX, debido a la gran cantidad de switches de sus infraestructuras. Los switches white box ofrecen grandes beneficios en las implementaciones de red incluyendo la ruptura de la dependencia de los fabricantes, la flexibilidad en la implementación y la capacidad de programar servicios de acuerdo a necesidades particulares. En un campus empresarial el uso de switches white-box resulta una alternativa efectiva en costo cuando se requiere una gran cantidad de dispositivos SDN. Estos dispositivos pueden ser gestionados mediante herramientas de automatización de código abierto, como OpenStack<sup>40</sup>, Puppet<sup>41</sup> y Chef<sup>42</sup>.

El fabricante Pica8 ofrece los siguientes switches white box pre-cargados<sup>43</sup>:

- P-5401: 32 x 40G, P-5101: 40 x 10G y 8 x 40G, P-5101: 40 x 10G y 8 x 40G, P-3930: 48 x 10G-T y 4 x 40G, P-3922: 48 x 10G y 4 x 40G, P-3297: 48 x 1G-T y 4 x 10G.

### 6.3.3 Switches Brite Box

Los switches brite box son switches white box con OEM (Original Equipment Manufacturer). Entre los principales OEMs se encuentran HP, DELL o Big Switch Networks. Con esta opción se obtiene la ventaja de recibir soporte y servicio directo del fabricante.

Juniper incluye el switch OCX1100 basado en Linux y tiene precargado el firmware ONIE OCP<sup>44</sup> (Open Compute Project), el cual le permite correr sistemas operativos de otros fabricantes, incluyendo Big Switch Networks, Cumulus Networks y Pica8. El switch sigue los principios de diseño del switch *Wedge*<sup>45</sup> diseñado por Facebook que disgrega el plano de control x86 del plano de reenvío Broadcom.

---

<sup>40</sup> <https://www.sdxcentral.com/cloud/open-source/definitions/openstack-networking>

<sup>41</sup> <https://www.sdxcentral.com/listings/puppet-labs>

<sup>42</sup> <https://www.sdxcentral.com/listings/opscode-inc>

<sup>43</sup> <http://www.pica8.com/products/pre-loaded-switches>

<sup>44</sup> <http://www.opencompute.org/>

<sup>45</sup> <https://code.facebook.com/posts/145488969140934/open-networking-advances-with-wedge-and-fboss/>

Dell ofrece varios switches abiertos y soporta sistemas operativos de red de los fabricantes Big Switch Networks y Cumulus.

HP ofrece el hardware HP/Accton que puede cargar sistemas operativos de red de varios proveedores a través del ONIE de Cumulus Linux. Sistemas Operativos de Red

## **6.4 Controladores SDN OpenFlow de Código Abierto**

En la última década el desarrollo de controladores SDN ha experimentado un alza importante gracias al esfuerzo de organizaciones y cuerpos de estándares de investigación como la ONF y el grupo GENI (Global Environment for Network Innovation)<sup>46</sup>. Algunos controladores han sido depreciados por la falta de soporte, mientras que otros han evolucionado e inspirado el desarrollo de controladores OpenFlow comerciales. Una lista completa de controladores abiertos OpenFlow se puede obtener en el directorio de proyectos de código abierto de sdx central<sup>47</sup>. En la siguiente sección se muestran los controladores SDN de código abierto vigentes.

## **6.5 Controladores SDN de Código Abierto Vigentes**

En la siguiente sección se analizan 3 controladores de código abierto disponibles en la industria para su uso en redes de producción y en el campo de la investigación incluyendo OpenDaylight, Floodlight y Ryu. Los criterios de selección de los controladores utilizados fueron: (1) tiempo en el mercado mayor a 3 años, (2) foco activo en su desarrollo, (3) soporte del API OpenFlow 1.3 o superior y (4) disponibilidad de documentación y soporte incluyendo soporte en línea con FAQ (frequently asked questions), guías de implementación y guías de operación.

En la siguiente sección se hace un análisis comparativo de las alternativas del plano de control SDN enfocado en los aspectos del software de los controladores existentes. Para el análisis del software se utiliza un subconjunto de la plantilla de análisis de controladores de G. Landi y los coautores del entregable *D3.1 SDN Framework Functional Architecture* [76] para cubrir los aspectos más importantes de un controlador SDN incluyendo: (1) Flexibilidad de la arquitectura del software, (2) estabilidad, (3) mantenimiento del software, (4) disponibilidad y calidad de la documentación, (5) usabilidad del software y soporte de la comunidad.

### **6.5.1 OpenDaylight**

OpenDaylight<sup>48</sup> es un controlador de código abierto que sigue una arquitectura modular, abierta y extensible [77]. Está implementado en software e incluye su

---

<sup>46</sup> <https://www.geni.net>

<sup>47</sup> <https://www.sdxcentral.com/directory/nfv-sdn/open-source-projects>

<sup>48</sup> <https://www.opendaylight.org/>

propia VM Java, lo cual le permite ser desplegado en cualquier plataforma de hardware y software con soporte Java.

### Alcance de OpenDaylight

- Está conformado por una combinación de componentes incluyendo un controlador completamente conectable (*fully pluggable*), interfaces A-CPI (Application-Controller Plane Interface) y D-CPI (Data-Controller Plane Interface), conectores multi-protocolo y aplicaciones.
- Cuenta con APIs northbound y southbound bien definidas y documentadas.
- Conformar una plataforma común a usuarios y vendedores que pueden innovar y colaborar para comercializar soluciones basadas en SDN y NFV.
- Cuenta con un amplio respaldo de la industria, tiene el patrocinio de la fundación Linux y la contribución de empresas líderes en el mercado incluyendo Brocade, Cisco, Ericsson, Hewlett Packard Enterprise, Intel, redhat, Inocyb, Nec, A10Networks y muchas otras.
- Cuenta con el soporte de la comunidad de código abierto incluyendo numerosos medios como wiki, IRC, listas de correos, grupos de reuniones, *hackfests*, foros y tutoriales de video.
- Está basado en un framework modular basado en Java y soporta el framework de programación OSGi<sup>49</sup> (Open Specifications Group Initiative) y soporta el API Northbound REST bidireccional. La especificación OSGi también conocida como *Dynamic Module System for Java*, define una arquitectura para el desarrollo de aplicaciones modulares. Implementaciones de contenedores OSGi como Knopflerfish<sup>50</sup>, Equinox<sup>51</sup> y Apache Felix<sup>52</sup> permiten romper una aplicación en múltiples módulos y facilitan la gestión de interdependencia entre ellos. El framework implementa un modelo de componentes dinámicos que especifica como deben ser definidas las interfaces y los elementos que deben ser incluidos en la definición de la interfaz. Los componentes son aplicaciones que pueden ser instaladas, iniciadas, detenidas, actualizadas y desinstaladas remotamente sin recurrir a un reinicio de la máquina.

### Software

- Última versión y fecha de liberación: Fluorine<sup>53</sup> (versión actual): agosto 30, 2018.
- Arquitectura de Software: La arquitectura de la plataforma consta de 3 capas (ver Figura 6.2 tomada del sitio web del proyecto<sup>54</sup>):

---

<sup>49</sup> <https://www.osgi.org>

<sup>50</sup> <http://www.knopflerfish.org>

<sup>51</sup> <http://www.eclipse.org/equinox>

<sup>52</sup> <http://felix.apache.org>

<sup>53</sup> <https://docs.opendaylight.org/en/stable-fluorine/downloads.html>

<sup>54</sup> <http://www.opendaylight.org>

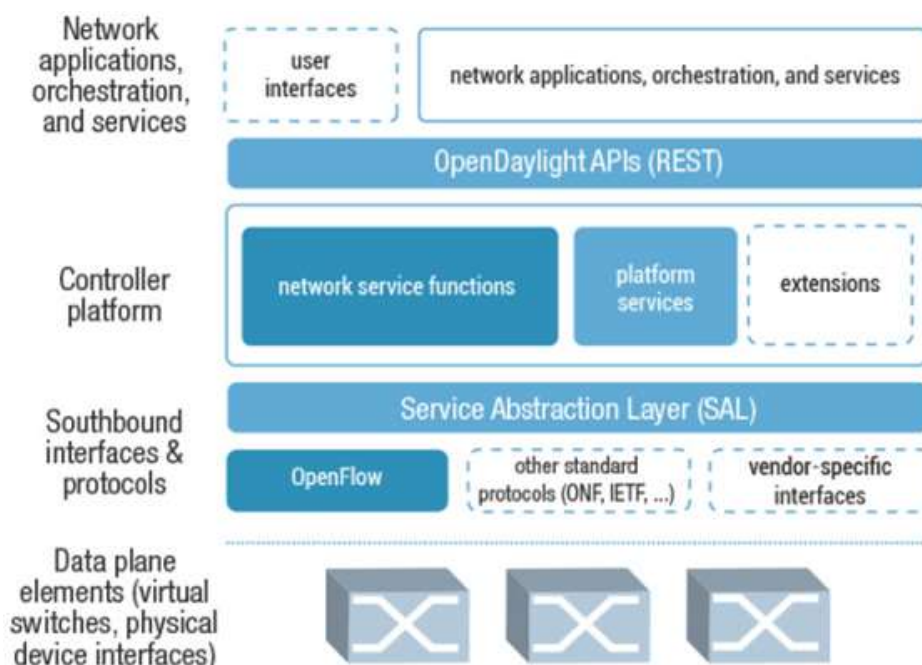


**Figura 6.2: Arquitectura General OpenDaylight**

- **Servicios y Orquestación de Aplicaciones de Red:** La capa superior está conformada por las aplicaciones del negocio y las aplicaciones de control, aprovisionamiento y gestión de la red. Adicionalmente se encuentran aplicaciones de nube, centros de datos, funciones de red y servicios de virtualización.
- **Plataforma de Controladores:** La capa intermedia está representada por la abstracción SDN. Está compuesta de una colección de módulos que se conectan dinámicamente para ejecutar servicios y tareas de red incluyendo el gestor de la topología (*Topology Manager*), el gestor de estadísticas (*Statistics Manager*), el gestor de switches (*Switch Manager*), el gestor de reglas de reenvío (*Forwarding Rules Manager*) y el rastreador de hosts (*Host Tracker*). Los módulos proveen APIs comunes a las aplicaciones y a la capa de orquestación para permitirles controlar y programar la red.
- **Conectores de Protocolos e Interfaces Southbound:** La capa inferior incluye múltiples interfaces Southbound a través de conectores que se cargan dinámicamente, incluyendo OpenFlow 1.0, OpenFlow 1.3, Of-Config, NETCONF, LISP, OVSDB, BGP, CAPWAP y otros para controlar el comportamiento de reenvío de la red y los dispositivos SDN físicos y virtuales, como switches y routers, que proveen la fábrica de conectividad de la red. Los módulos son enlazados dinámicamente a la capa de abstracción de servicios SAL (*Service Abstraction Layer*). La SAL expone los servicios de los dispositivos a los módulos de red del controlador y define como satisfacer los servicios independientemente de los protocolos subyacentes entre el controlador y los dispositivos SDN.

- Distribución de Código Fuente: El proyecto OpenDaylight mantiene un repositorio *git*<sup>55</sup>. OpenDaylight tiene más de 40 proyectos, cada uno de los cuales mantiene su propio repositorio<sup>56</sup>.
- Distribución del Paquete del Software: Las diferentes versiones del proyecto pueden ser descargadas desde el sitio web del proyecto<sup>57</sup>.
- Lenguaje de Núcleo: Java.
- Interfaces/Lenguaje de Aplicaciones: OpenDaylight soporta el *framework* OSGi y REST bidireccional para el API Northbound. El *framework* OSGi es utilizado por las aplicaciones que corren en el mismo espacio de direcciones del controlador. La API REST, basada en web, es utilizada por las aplicaciones que no corren en el mismo espacio de direcciones del controlador. La capa MD-SAL (Model Driven Service Abstraction Layer) soporta APIs DOM, las cuales son utilizadas por tipos de aplicaciones conducidas por XML.

## Usabilidad

- Documentación de la Instalación: Las instrucciones de la instalación del controlador se encuentran en el wiki del proyecto<sup>58</sup>. La guía de instalación también se encuentra disponible en el sitio de descarga del controlador<sup>59</sup>.
- Documentación Operativa: La guía del usuario puede ser descargada con la versión OpenDaylight correspondiente. Adicionalmente se encuentra disponible una guía de inicio rápido.<sup>60</sup>

## Desarrollo

- Documentación de Desarrollo: Se encuentra disponible un wiki para desarrolladores<sup>61</sup>, el cual es actualizado periódicamente. Adicionalmente existe una lista de correos y un canal IRC.
- Roadmap de Desarrollo: Se encuentra una sección de desarrolladores<sup>62</sup> en el wiki del proyecto. Adicionalmente se encuentran aplicaciones de ejemplo<sup>63</sup>.
- Toolchain: Java 1.7 + Maven + OSGi.
- Integración IDE: Hasta el presente solamente el controlador y los proyectos han sido validados para importarse a Eclipse sin errores de compilación<sup>64</sup>.

## Implementación

- Requerimientos del Sistema: El controlador OpenDaylight corre sobre una máquina virtual Java. Al ser el controlador una aplicación Java, este puede

---

<sup>55</sup> <http://git.opendaylight.org>

<sup>56</sup> <https://git.opendaylight.org/gerrit/#/admin/projects>

<sup>57</sup> <http://www.opendaylight.org>

<sup>58</sup> <https://wiki.opendaylight.org>

<sup>59</sup> <http://www.opendaylight.org/software/downloads>

<sup>60</sup> <http://www.opendaylight.org/resources/getting-started-guide>

<sup>61</sup> <https://wiki.opendaylight.org>

<sup>62</sup> [https://wiki.opendaylight.org/view/GettingStarted:Developer\\_Main](https://wiki.opendaylight.org/view/GettingStarted:Developer_Main)

<sup>63</sup> [https://wiki.opendaylight.org/view/OpenDaylight\\_Controller:Sample\\_Applications](https://wiki.opendaylight.org/view/OpenDaylight_Controller:Sample_Applications)

<sup>64</sup> [https://wiki.opendaylight.org/view/GettingStarted:\\_Eclipse](https://wiki.opendaylight.org/view/GettingStarted:_Eclipse)

ejecutarse en cualquier sistema operativo que soporte Java. Para mejores resultados, se recomienda una distribución Linux y Java 1.7.

- El controlador generalmente forma parte de una VM pre-configurada, como ocurre con la versión *Hydrogen* del controlador la cual viene pre-configurada en una VM con una distribución GNU/Linux Ubuntu y se recomienda al menos 4GB de RAM.
- OpenDaylight hace bastante uso del lenguaje *Xtend* por lo cual se recomienda utilizar *Eclipse* con el conector *Xtend*.
- Sistema Operativo Soportado: Se recomienda una distribución Linux actualizada, sin embargo, cualquier sistema operativo corriendo una VM Java 1.7 debe funcionar.
- Dependencias Runtime: Java 1.7.

### Principales Funcionalidades

- Versión OpenFlow Soportada: OpenFlow versión 1.0 y 1.3.x.
- Protocolos D-CPI: TTP, Of-Config, OVSDb, NETCONF, LISP, BGP, PCEP, CAPWAP, OCP, OPFLEX, SXP, SNMP, USC, SNBI, IoT Http/CoAP, LACP, PCMM/COPS.
- Integración con Sistemas de Gestión de Nube: Incluye un driver para el conector Neutron ML2 (*Modular Layer 2*) para permitir la comunicación entre Neutron y OpenDaylight. También incluye APIs hacia el norte para interactuar con Neutron y utiliza OVSDb para la configuración de switches virtuales hacia el sur.
- Lenguajes A-CPI: Soporta el framework OSGi y Java como lenguaje de desarrollo y soporta REST bidireccional empleando JSON y API DOM empleando XML.
- Servicios/Funciones del Core:
  - Topology Manager.
  - Statistics Manager.
  - Switch Manager.
  - Forwarding Rules Manager.
  - Host Tracker.
- Aplicaciones: OpenDOVE y VTN para implementar virtualización de red overlay. Integración con OpenStack a través de un API Neutron.
- Métodos A-CPI: Incluye el siguiente directorio de módulos para el API REST:
  - Topology REST API: Permite acceder la topología almacenada por en el controlador y mantenida por el módulo *Topology Manager*.
  - Host Tracker REST API: Lleva la traza de hosts en la red.
  - Flow Programmer REST API: Permite programar flujos en la red OpenFlow.
  - Static Routing REST API: Permite gestionar rutas estáticas capa 3 en la red.
  - Statistics REST API: Retorna información de estadísticas de los conectores de los protocolos hacia el sur incluyendo estadísticas OpenFlow: número de paquetes procesados, estado de los puertos, estadísticas de tablas, flujos y otros.

- Subnets REST API: Permite gestionar subredes capa 3 en un contenedor.
- Switch Manager REST API: Permite acceder a los nodos, a sus conectores y propiedades.
- User Manager REST API: Provee primitivas para gestionar los usuarios que se conectan al controlador.
- Container Manager REST API: Provee primitivas para crear, eliminar y gestionar contenedores en la red controlada por OpenDaylight. Es requerido un contenedor por defecto para operar con normalidad.
- Connection Manager REST API: Permite gestionar los nodos conectados a un controlador.
- Bridge Domain REST API: Permite acceder primitivas del protocolo OVSDDB para programar un switch virtual Open vSwitch.
- Neutron ML2/Network Configuration API: Permite integrarse con OpenStack.
- Soporte de Comunicación C2C (Controlador-a-Controlador): Incluye la aplicación ODL-SDNi para proveer el establecimiento de la interfaz Este-Oeste (SDNi communication) entre múltiples controladores OpenDaylight. La aplicación es responsable de compartir y recopilar información hacia/desde controladores federados.

### **Licenciamiento**

- Licencia: Eclipse Public License (EPL-1.0).

### **Impacto Potencial**

- Principales socios de la industria: IBM, Cisco, Brocade, Juniper, Microsoft, Dell, HPE, Ericsson, Intel, A10Networks, y otros.
- Popularidad: Alta. Tiene fuerte soporte de la industria y de la comunidad de código abierto.
- Uso comercial: El controlador Brocade Vyatta Controller está basado en OpenDaylight.

## **6.5.2 Floodlight**

Floodlight<sup>65</sup>[78] es un controlador OpenFlow extensible basado en Java con licencia Apache desarrollado por una comunidad abierta de programadores y patrocinado por Big Switch Networks y un conjunto de aplicaciones construidas en el tope del controlador.

### **Breve Descripción**

- Controlador OpenFlow de clase empresarial basado en Java bajo licencia Apache.
- Ofrece un sistema de carga modular que simplifica su extensión y la adición de mejoras.
- Se integra bien con el IDE Eclipse y es fácil de usar, construir y ejecutar.

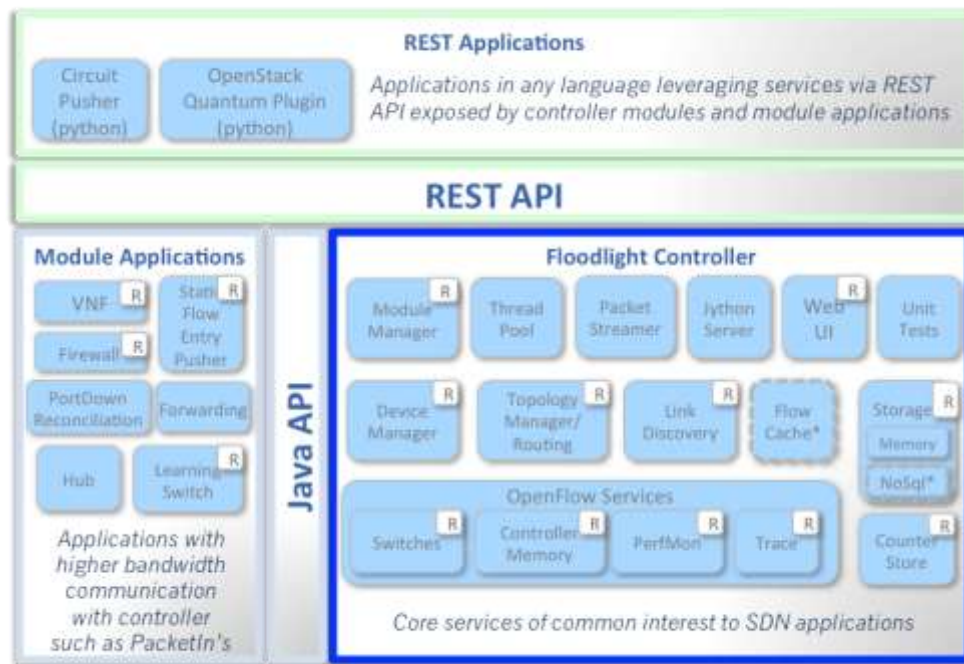
---

<sup>65</sup> <http://www.projectfloodlight.org/floodlight>

- Soporta un rango amplio de switches OpenFlow físicos y virtuales.
- Ha sido probado y soportado activamente por la comunidad de desarrolladores del proyecto.
- Soporte completo de OpenFlow 1.3 y soporte experimental de OpenFlow 1.1, OpenFlow 1.2 y OpenFlow 1.4.

## Software

- Última versión y fecha de liberación: Floodlight v1.2<sup>66</sup>. Febrero. 2016.
- Arquitectura de Software: La arquitectura del controlador Floodlight está conformada por: (1) un conjunto de módulos de servicios de red (*Core Services*) que conforman el núcleo del controlador, (2) un conjunto de módulos de aplicaciones (*Module Applications*) para uso interno y (3) varias aplicaciones REST que corren en el tope del controlador: *Circuit Pusher* y el conector *OpenStack Quantum* (ver Figura 6.3 tomada del sitio web del controlador<sup>67</sup>). Los módulos marcados con la letra "R" exponen su funcionalidad a través de APIs REST a las aplicaciones externas al controlador. Los módulos también pueden ser accedidos por otros módulos del controlador a través de APIs Java. Todos los módulos pueden ser iniciados o detenidos desde un archivo de configuración, solamente al momento de la compilación.



\* Interfaces defined only & not implemented: FlowCache, NoSql

**Figura 6.3: Arquitectura del Controlador Floodlight**

- Distribución de Código Fuente: El controlador puede ser obtenido desde un repositorio git localizado en GitHub<sup>68</sup>.

<sup>66</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+v1.2>

<sup>67</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/The+Controller>

<sup>68</sup> <https://github.com/floodlight/floodlight>

- Distribución del Paquete del Software: Se puede obtener el controlador directamente desde GitHub como un archivo .gz/zip. Existe también una VM con el controlador pre-instalado que se puede descargar desde el sitio de descarga del sitio web del controlador<sup>69</sup>.
- Lenguaje de Núcleo: Java
- Interfaces/Lenguaje de Aplicaciones: Los módulos del controlador pueden ser escritas en Java. Las aplicaciones que hacen uso del API REST pueden ser escritas en cualquier lenguaje siempre y cuando interactúen con el controlador a través de un API REST.

## Usabilidad

- Documentación de la Instalación: La documentación actualizada sobre la instalación se encuentra disponible en la sección *Installation Guide*<sup>70</sup> del sitio web del controlador.
- Documentación Operativa: La documentación actualizada sobre el uso del controlador se encuentra disponible en la sección *Installation Guide*<sup>71</sup> del sitio web del controlador.

## Desarrollo

- Documentación de Desarrollo: La documentación actualizada para los desarrolladores se encuentra disponible en la sección *For Developers*<sup>72</sup> del sitio web del controlador.
- Roadmap de Desarrollo: El mapa de ruta del proyecto se encuentra en la sección *Releases and Roadmap*<sup>73</sup> del sitio web del proyecto del controlador.
- Toolchain: Java 1.7 + Maven.
- Integración IDE: Ha sido probado en Eclipse. Se pueden utilizar otros IDE disponibles.

## Implementación

- Requerimientos del sistema: No se ha encontrado información relacionada.
- Sistema Operativo Soportado: Se recomienda una distribución Linux Ubuntu 14.04 TLS *Trusty Tahr* o Ubuntu 16.04.1 LTS *Xenial Xerus*. Para instalar Floodlight en Linux se requiere un cliente Git, Python y la herramienta Apache Ant. El controlador también puede ser obtenido como una VM pre-configurada con mininet, Open vSwitch y Floodlight v1.1 desde el sitio web del proyecto<sup>74</sup>.
- Dependencias Runtime: Java 1.7.

## Principales Funcionalidades

- Versión OpenFlow Soportada: Soporta OpenFlow v1.0 a v1.4.

<sup>69</sup> <http://www.projectfloodlight.org/download>

<sup>70</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Installation+Guide>

<sup>71</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Installation+Guide>

<sup>72</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/For+Developers>

<sup>73</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Releases+and+Roadmap>

<sup>74</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+VM>

- Protocolos D-CPI soportados: OpenFlow solamente. Es relativamente sencillo añadir un nuevo módulo en Floodlight que pueda servir como un conector hacia el sur.
- Integración con Sistemas de Gestión de Nube: Incluye el módulo Virtual Network Filter (VNF), el cual provee un API REST para integrarse con Quantum/OpenStack.
- Lenguajes A-CPI: Utiliza REST con formato de datos JSON. Las aplicaciones internas pueden ser escritas con un API Java.
- Servicios/Funciones del Core: Entre las funciones de núcleo de Floodlight se encuentran:
  - Inventario de dispositivos (hosts).
  - Administrador de la topología (inventario de switches, enlaces, y otros).
  - Módulo para insertar flujos y grupos en la red OpenFlow (*Static Entry Pusher*).
  - Monitoreo del desempeño del controlador.
  - Capacidad de reenvío Dijkstra que permite interconectar islas de switches OpenFlow con switches no-OpenFlow.
- Aplicaciones: El controlador Floodlight provee las siguientes aplicaciones principales<sup>75</sup>:
  - Circuit Pusher<sup>76</sup>: Utiliza APIs REST Floodlight para crear un circuito bidireccional entre 2 puntos terminales IP. Por ejemplo, crear entradas de flujos permanentes en todos los switches en la ruta entre 2 dispositivos basado en direcciones IP con una prioridad específica.
  - Conector OpenStack Quantum<sup>77</sup>: Permite que el controlador Floodlight corra como una red de *backend* para el sistema de control de nubes OpenStack<sup>78</sup> utilizando el conector Neutron<sup>79</sup>. Neutron expone un modelo red-como-un-servicio vía una API REST que haya implementado Floodlight. Existen 2 componentes principales para implementar esta solución: (1) el módulo *VirtualNetworkFilter* en Floodlight encargado de implementar el API Neutron y (2) el conector Neutron *RestProxy* que conecta el controlador Neutron al controlador Floodlight.
  - Forwarding<sup>80</sup>: Aplicación de reenvío reactiva de paquetes entre 2 dispositivos habilitada por defecto.
  - Static Flow Entry Pusher: Aplicación para instalar una entrada de flujos específica a un switch específico. Está habilitada por defecto y expone un conjunto de APIs REST para añadir, remover o consultar entradas de flujos.
  - Virtual Network Filter<sup>81</sup>: Módulo para virtualizar redes basadas en direcciones MAC capa 2. Permite crear varias redes lógicas capa 2

<sup>75</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Applications>

<sup>76</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Circuit+Pusher>

<sup>77</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/OpenStack>

<sup>78</sup> <http://www.openstack.org/>

<sup>79</sup> <http://docs.openstack.org/developer/neutron/>

<sup>80</sup> <https://floodlight.atlassian.net/wiki/pages/viewpage.action?pageId=1343630>

en un solo dominio capa 2. Puede ser utilizada para un despliegue OpenStack o de manera individual.

- Learning Switch: Switch de aprendizaje capa 2 común. No está habilitado por defecto. Expone un API REST para listar la tabla de un switch incluyendo hosts conocidos, vlans y puertos.
- Firewall<sup>82</sup>: Módulo para implementar reglas ACL a switches OpenFlow en la red utilizando flujos y monitoreando el comportamiento *packet-in*. Las reglas ACL establecen condiciones para permitir o negar un flujo de tráfico en un switch de ingreso. El módulo es cargado pero deshabilitado por defecto.
- Hub: Aplicación hub que siempre inunda un paquete de ingreso a todos los puertos activos de un switch. No está habilitada por defecto.
- Load Balancer<sup>83</sup>: Módulo para balanceo de cargas de flujos TCP, UDP y ping. El módulo es accedido vía un API REST definido cercano a la propuesta del API OpenStack Quantum LBaaS<sup>84</sup> (Load-balancer-as-a-Service) v1.0. El código no está completo hasta este momento y presenta las siguientes limitaciones: (1) las estadísticas de flujos y los registros de los clientes no son purgados después de su uso, lo cual exhausta las tablas de flujos del switch en el tiempo, (2) sigue un esquema de balanceo de carga round robin entre servidores basado en conexiones y no considera el volumen de tráfico y (3) no ha sido implementado todavía la funcionalidad de monitoreo de la salud de los servidores.
- Métodos A-CPI: El API REST disponible en el A-CPI soporta las siguientes funciones<sup>85</sup>:
  - Información de la red y la topología (hosts conectados, switches, enlaces y otros.)
  - Estadísticas OpenFlow.
  - Estado del controlador (uso de memoria, estado).
  - Configuración de flujos estáticos en la red.
  - Configuración de reglas ACL en el firewall.
  - Configuración de redes virtuales a través de un conector OpenStack.
  - Balanceador de carga para flujos ping, tcp y udp.
- Soporte de Comunicación C2C: Floodlight no provee soporte para ningún tipo de comunicación C2C, por lo cual no soporta alta disponibilidad, ni interfaces este-oeste, ni controladores jerárquicos.

## Licenciamiento

- Licencia: Licencia Apache 2.0.

---

<sup>81</sup> <https://floodlight.atlassian.net/wiki/pages/viewpage.action?pagelId=1343627>

<sup>82</sup> <https://floodlight.atlassian.net/wiki/pages/viewpage.action?pagelId=1343599>

<sup>83</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Load+Balancer>

<sup>84</sup> <https://wiki.openstack.org/wiki/Neutron/LBaaS>

<sup>85</sup> <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+REST+API+pre-v1.0>

### Impacto Potencial

- Principales Socios de la Industria: Está respaldado por Big Switch Networks.
- Popularidad: Fue el principal controlador disponible antes de la llegada de Floodlight.
- Uso Comercial: Representa el núcleo del controlador comercial Big Switch Network Controller.

### 6.5.3 Ryu

**Ryu**<sup>86</sup> es un *framework* SDN basado en componentes con APIs bien definidas que facilitan la creación de aplicaciones de control y gestión de redes (ver Figura 6.4 tomada de [78]).

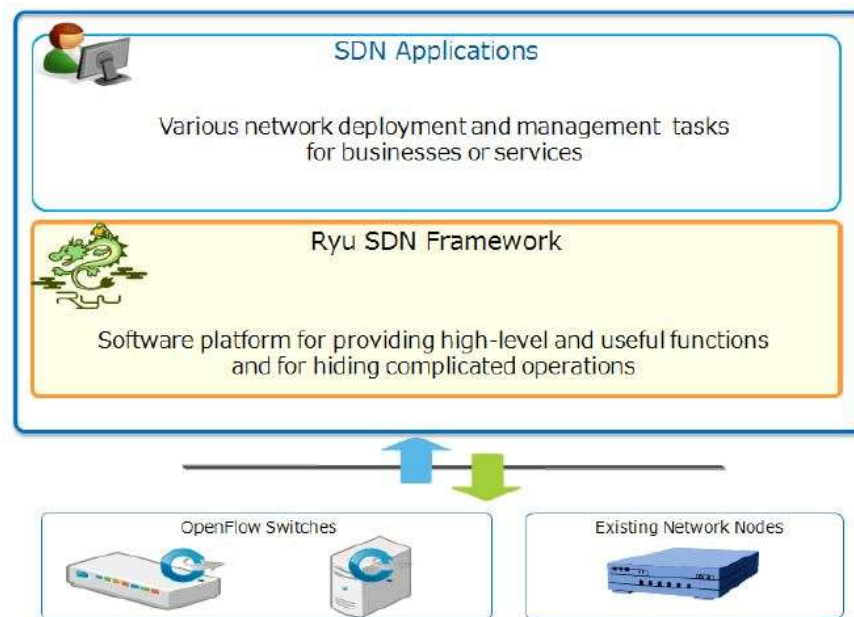


Figura 6.4: Framework SDN Ryu

El framework SDN Ryu es un sistema de software de código abierto OSS (Open Source Software) desarrollado y soportado por el grupo NTT que permite el desarrollo de aplicaciones SDN y soporta switches SDN físicos y virtuales a través de múltiples protocolos de control hacia el sur, como OpenFlow. Para los dispositivos de red no OpenFlow, Ryu provee librerías como NetConf y SNMP. Adicionalmente el framework incluye los protocolos OVSDB y OF-Config 1.1 para configurar y gestionar switches SDN y NetFlow y sFlow para monitorear y recopilar estadísticas.

### Breve Descripción

- Está implementado totalmente en Python y está soportado por el grupo de investigación NTT<sup>87</sup>.

<sup>86</sup> <http://osrg.github.io/ryu>

- Provee componentes de software con APIs bien definidas que facilitan el desarrollo de nuevas aplicaciones de gestión y control.
- Soporta varios protocolos para gestionar dispositivos de red, incluyendo OpenFlow, Netconf y OF-config.
- Soportar las especificaciones completas de OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5 y Extensiones Nicira.
- Está orientado a la investigación y a la creación de prototipos rápidos de aplicaciones de red.

## Software

- Última versión y fecha de liberación: Ryu v4.8<sup>88</sup>. Noviembre. 2016.
- Arquitectura de Software: El framework SDN Ryu está conformado por un conjunto de componentes, librerías e interfaces para el intercambio de información. Los componentes del framework se muestran en la Figura 6.5 (tomada de [78]). El framework SDN Ryu sigue una arquitectura de manejo de eventos y contempla una capa de control y una capa de aplicaciones. La capa de control incluye el soporte de protocolos hacia el sur y los siguientes componentes principales: manejador de eventos, analizador de mensajes OpenFlow, gestor de memoria, gestor de aplicaciones, servicios de infraestructura y un conjunto de librerías que incluyen NETCONF, sFlow y Netflow. La capa de aplicaciones contempla las siguientes aplicaciones de red: switch capa 2, firewall, IDS (Snort), abstracciones de túneles GRE, VRRP (*Virtual Router Redundancy Protocol*) y los servicios de descubrimiento de la topología y manejo de estadísticas entre otros.

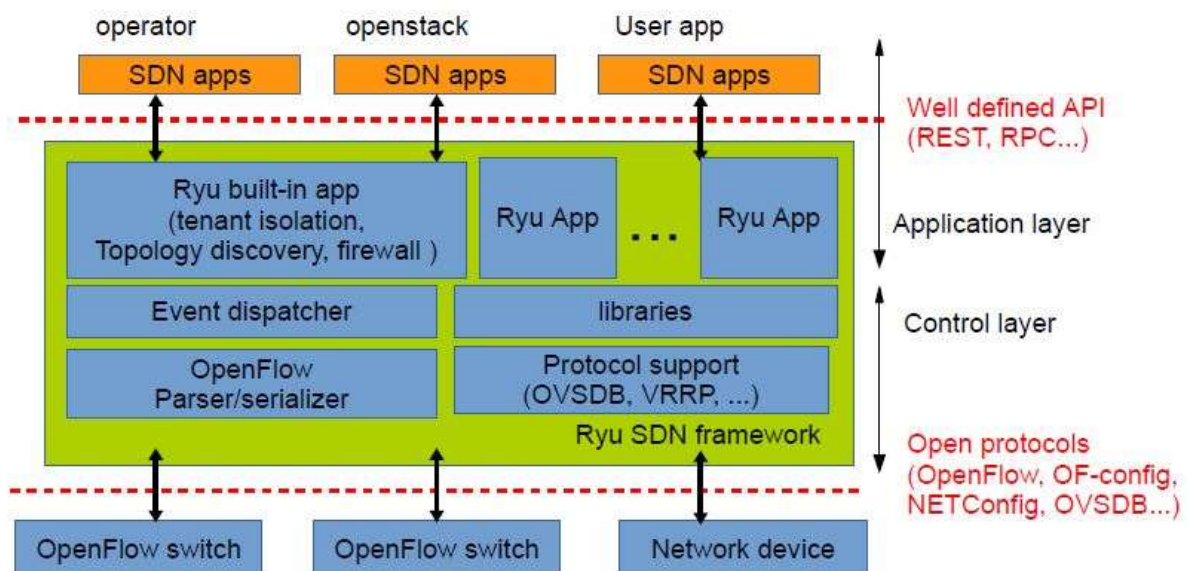


Figura 6.5: Componentes e Interfaces del Framework SDN Ryu

<sup>87</sup> [http://www.ntt.co.jp/about\\_e/r\\_d.html](http://www.ntt.co.jp/about_e/r_d.html)

<sup>88</sup> <https://pypi.python.org/pypi/ryu/4.8>

- Adicionalmente Ryu tiene un conector para integrarse al controlador de redes de nube OpenStack Neutron que soporta túneles *overlay* basados en GRE y configuraciones de VLANs.
- Distribución de Código Fuente: El controlador puede ser obtenido desde un repositorio git localizado en GitHub<sup>89</sup>.
- Distribución del Paquete del Software: El controlador se instala utilizando la herramienta PIP de Python a través del comando "pip install ryu". También se pueden obtener VMs pre-configuradas desde el sitio web del controlador<sup>90</sup>.
- Lenguaje de Núcleo: Python 2.6+, *greenlets*, pseudo-multihilo cooperativo (utilizando un sólo núcleo).
- Interfaces/Lenguaje de Aplicaciones: Python 2.6+ para las aplicaciones del controlador y REST para las aplicaciones externas.

## Usabilidad

- Documentación de la Instalación: Disponible, actualizada y detallada.
- Documentación Operativa: Disponible, actualizada y detallada.

## Desarrollo

- Documentación de Desarrollo: Disponible, actualizada, pero no muy detallada.
- Roadmap de Desarrollo: Existe documentación para el desarrollo de aplicaciones disponible, actualizada y detallada con tutoriales de ejemplos.
- Cadena de Herramientas (*Toolchain*): Python 2.6+ con las siguientes librerías:
  - python-eventlet
  - python-routes
  - python-webob
  - python-paramiko

También se recomienda la herramienta Postman<sup>91</sup> para acceder el API REST de Ryu. Postman es una aplicación basada en Chrome que utiliza una tecnología de navegación web que provee un ambiente para construir, enviar y recibir solicitudes HTTP. Es especialmente útil cuando se trabaja con estructuras JSON grandes requeridas por el API REST de Ryu para consultar switches OpenFlow.

- Integración IDE: Cualquier editor de texto. También se puede utilizar el conector Python para Eclipse PyDev<sup>92</sup>. Otra opción disponible es utilizar el IDE para Python PyCharm<sup>93</sup>.

## Implementación

- Requerimientos del sistema: Proceso python liviano individual.

---

<sup>89</sup> <https://github.com/osrg/ryu>

<sup>90</sup> <https://osrg.github.io/ryu/resources.html>

<sup>91</sup> <https://www.getpostman.com>

<sup>92</sup> <http://www.pydev.org>

<sup>93</sup> <https://www.jetbrains.com/pycharm>

- Sistema Operativo Soportado: Distribución GNU/Linux reciente.
- Dependencias en Tiempo de Ejecución: Paquetes Python (python-eventlet, python-routes, python-webob y python-paramiko).

### **Principales Funcionalidades**

- Versión OpenFlow Soportada: 1.0, 1.2, 1.3, 1.4, 1.5 y extensiones Nicira.
- Protocolos D-CPI soportados: Netconf, OF-config, SNMP y OVSDB.
- Integración con Sistemas de Gestión de Nube: Incluye el conector OpenStack. Utiliza APIs REST.
- Lenguajes A-CPI: Utiliza Python para aplicaciones internas y servicios web REST para aplicaciones externas.
- Servicios/Funciones del Core: Incluye analizadores y generadores de paquetes de diferentes protocolos de red.
- Aplicaciones:
  - Respondedor OpenFlow 1.0 simple para efectuar pruebas de desempeño (ryu.app.cbench).
  - Implementación de switch de aprendizaje capa 2 OpenFlow 1.0 (ryu.app.simple\_switch).
  - Módulo de descubrimiento de enlaces y switches (ryu.topology).
- Métodos A-CPI: El framework SDN Ryu ofrece las APIs RESTful hacia el norte con formato de datos JSON para acceder información y configuración. Cada módulo del framework puede ofrecer su propio API REST, como se indica en el siguiente listado:
  - Recuperar información al controlador, indicar switches conectados o puertos individuales y añadir, modificar y eliminar entradas de flujos (ofctl\_rest-py).
  - Recopilar información de la topología (rest\_topology.py).
  - Recopilar información de los puntos terminales (rest.py).
  - Ajustar parámetros QoS (rest\_qos-py).
  - Manejar tablas de enrutamiento y direccionamiento del módulo de enrutamiento (rest\_router.py).
  - Recuperar estatus y logs y añadir y remover reglas en el módulo de firewall (rest\_firewall.py).
- Soporte de Comunicación C2C: Es capaz de soportar servicios de alta disponibilidad a través del componente Zookeeper. El uso actual de las capacidades de alta disponibilidad debe ser provisto por aplicaciones de red externas.

### **Licenciamiento**

- Licencia: Apache 2.0.

### **Impacto Potencial**

- Principales Socios de la Industria: NTT.
- Popularidad: Es utilizado principalmente en campos de investigación debido a que es fácil de aprender y fácil de expandir.
- Uso comercial: Ha tenido poco uso comercial debido en parte a que varias evaluaciones de desempeño[74][79] han reportado que el controlador Ryu es lento en comparación con otros controladores. Adicionalmente el

controlador Ryu no es escalable, debido a que el controlador corre sobre el lenguaje interpretado Python y el framework completo es esencialmente un solo hilo, por lo cual, no puede escalar con núcleos de procesadores adicionales en el servidor donde se ejecuta.

## 6.6 Controladores SDN de Propósito Especial

En la categoría de controladores especiales se encuentran los controladores que se encargan de efectuar una función específica dentro de una red SDN.

### 6.6.1 FlowVisor

FlowVisor<sup>94</sup> [42] es un controlador de propósito especial que actúa como un proxy entre varios controladores y la infraestructura de switches de una red OpenFlow. FlowVisor particiona el ancho de banda y las tablas de flujos de cada switch en *slices*, donde cada *slice* recibe una tasa de datos de ancho de banda mínima y cada controlador invitado gestiona el reenvío de datos en las tablas de flujos virtuales de los switches pertenecientes a su *slice* correspondiente. FlowVisor está basado en estándares abiertos que corren sobre entornos multi-vendedor y está soportado por múltiples fabricantes incluyendo: NEC, IBM, Juniper, HP, Dell, Brocade, Extreme, Pronto, Intel, OVS y otros. *FlowVisor* se utiliza para virtualizar un número pequeño de redes virtuales inferior a 100 redes virtuales.

### 6.6.2 Ovs-controller

Ovs-controller<sup>95</sup> es una implementación de un controlador SDN OpenFlow sencillo para gestionar switches remotos. Forma parte de Open vSwitch y corre sobre Linux incluyendo la distribución Ubuntu.

### 6.6.3 FlowN

FlowN [80] es una solución de virtualización escalable y eficiente que utiliza una base de datos para almacenar y manipular el mapeo entre redes virtuales y físicas lo cual ofrece un mejor desempeño para un número grande de redes virtuales. Es una extensión del controlador NOX y puede soportar más de 100 redes virtuales. FlowN permite a cada usuario contar con su propio espacio de direcciones virtuales y aislamiento de ancho de banda.

### 6.6.4 RouteFlow

RouteFlow [81] es un controlador especial de código abierto que provee enrutamiento IP virtualizado sobre hardware OpenFlow. Está conformado por

---

<sup>94</sup> <https://github.com/OPENNETWORKINGLAB/flowvisor/wiki>

<sup>95</sup> <http://manpages.ubuntu.com/manpages/trusty/man8/ovs-controller.8.html>

una aplicación controlador OpenFlow, un servidor independiente, y un ambiente de red virtual que reproduce la conectividad de una infraestructura física y corre máquinas de enrutamiento IP. Las máquinas de enrutamiento generan la tabla FIB (Forwarding Information Base) en las tablas IP Linux de acuerdo a los protocolos de enrutamiento configurados (ejemplo OSPF, BGP).

## 6.7 Herramientas de Gestión SDN

OpenFlow cuenta con las siguientes herramientas de gestión: YANG [37], NETCONF [31] y OF-CONFIG 1.2 [40].

## 6.8 Herramientas de Monitoreo SDN

Una arquitectura de red SDN puede utilizar herramientas de monitoreo de red propietarias tradicionales, como Netflow [82] de Cisco, sFlow<sup>96</sup> de InMon, o JFlow [83] de Juniper Networks, o utilizar herramientas de monitoreo especializadas con poco *overhead* y alta precisión. A continuación se describen algunas arquitecturas de monitoreo disponibles:

### 6.8.1 PayLess

PayLess [84] es un *framework* de monitoreo basado en consultas para redes SDN que provee un API RESTful flexible para recopilar estadísticas a diferentes niveles de agregación, como flujos, paquetes y puertos. PayLess ejecuta la recopilación de información con alta precisión en tiempo real sin incurrir en un alto *overhead* de red. Utiliza un algoritmo de planificación adaptativo que permite alcanzar el mismo nivel de precisión del estándar OpenFlow sin tener que consultar continuamente a los switches. PayLess tiene una sobrecarga de 6,6 mensajes de monitoreo por segundo en promedio, comparado con las consultas periódicas de un controlador general, de 13,5 mensajes de monitoreo por segundo promedio.

### 6.8.2 OpenTM

OpenTM [85] es una arquitectura de monitoreo que lleva la traza de los flujos activos en una red OpenFlow. Adicionalmente obtiene la información de enrutamiento de la aplicación de enrutamiento del controlador y sondea periódicamente los contadores de cantidad de bytes y número de paquetes de los flujos activos en los switches a lo largo del camino del camino de datos. Para reducir la sobrecarga en la red se pueden sondear de manera aleatoria un subconjunto de los switches seleccionados cuidadosamente para no afectar la precisión de las estadísticas recopiladas por la herramienta.

---

<sup>96</sup> <http://www.sflow.org/sFlowOverview.pdf>

### 6.8.3 FlowSense

FlowSense [86] es una arquitectura de monitoreo que permite estimar el desempeño de una red OpenFlow a un bajo costo. Utiliza un método pasivo que captura y analiza el intercambio de los mensajes de control entre los switches y el controlador de una red OpenFlow asociados a cambios en el tráfico de la red, como ocurre con los mensajes *PacketIn* y *FlowRemoved*. FlowSense utiliza los mensajes *PacketIn* que notifican la llegada de un nuevo flujo y los mensajes *FlowRemoved* que notifican la expiración de un flujo, para estimar la utilización de un enlace por un flujo.

## 6.9 Herramientas de Depuración OpenFlow

Las herramientas de depuración son de suma importancia al momento de hacer implementaciones SDN ya que nos permiten efectuar validaciones del comportamiento de la red en desarrollo con respecto a los resultados esperados en el estudio y ayudan a identificar bugs o errores en la programación de las aplicaciones. En la Tabla 6.4 se enumeran algunas de las herramientas de depuración OpenFlow disponibles:

| Herramienta                                 | Descripción                                                                                                                                                                                                                                                                                                               |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NICE [87] (No bugs In Controller Execution) | Herramienta de pruebas automatizada utilizada para ayudar a descubrir bugs en programas OpenFlow a través del chequeo del modelo para explorar el espacio del estado del sistema completo: controlador, switches y hosts a través de la ejecución simbólica de manejadores de eventos                                     |
| oftrace <sup>97</sup>                       | Herramienta de trazas y análisis OpenFlow que toma como entrada un archivo con formato libpcap generado por tcpdump, wireshark u otro programa de análisis de red y genera como salida estadísticas útiles sobre la sesión OpenFlow                                                                                       |
| Anteater [88]                               | Herramienta de depuración OpenFlow que chequea invariantes de red en el plano de datos, como aspectos de conectividad o consistencia. La herramienta es agnóstica de los protocolos y puede capturar errores adicionales como fallas en el firmware del switch o inconsistencias con la comunicación del plano de control |
| VeriFlow [89]                               | Herramienta de verificación OpenFlow que reside entre el controlador y los switches siendo capaz de detener reglas erróneas que pudieran provocar un comportamiento anómalo antes de alcanzar la red                                                                                                                      |

---

<sup>97</sup> <http://archive.openflow.org/wk/index.php/Liboftrace>

|                   |                                                                                                                                                                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OfRewind [90]     | Herramienta de depuración que permite registrar eventos de red asociados a los planos de control y datos para reproducirlos en una etapa posterior y poder identificar y resolver los eventos que causaron alguna anomalía en la red                                                          |
| ndb [91]          | Herramienta que implementa puntos de chequeo y trazas hacia atrás de paquetes en ambientes SDN mostrando la secuencia de acciones de reenvío por las que atraviesa un paquete                                                                                                                 |
| STS <sup>98</sup> | Simulador de resolución de problemas de redes SDN escrito en Python y dependiente de POX. La herramienta simula los dispositivos de la red permitiendo generar casos de pruebas y examinar el estado de la red de manera interactiva para encontrar las entradas responsables de generar bugs |

Tabla 6.4: Herramientas de Depuración OpenFlow

## 6.10 Casos de Uso de SDN

La ONF organiza los casos de usos SDN para redes de campus empresariales en 6 dimensiones [92]: (1) virtualización de red (slicing/aislamiento de tráfico), (2) mejora en la seguridad y aplicación de políticas, (3) movilidad transparente y BYOD, (4) redes con conciencia de aplicación, (5) simplificación de la gestión y (6) video *streaming* y colaboración.

### 6.10.1 Virtualización de Red

El caso de uso de la virtualización de red en un ambiente SDN se descompone en tres bloques o conceptos: (1) superposición de la red, (2) slicing y (3) aislamiento de tráfico. En la siguiente sección se mencionan estos conceptos de virtualización.

**Superposición de Red:** Consiste en virtualizar switches de hardware por switches de software (*vSwitch*) que permiten la comunicación entre VMs en un ambiente de virtualización de servidores. Un *vSwitch* se implementa en software y corre en el ambiente de un hipervisor huésped. En una red SDN OpenFlow virtual el plano de datos de los *vSwitch* SDN es gestionado y programado desde un controlador SDN OpenFlow. La implementación de la red virtual sigue un modelo de despliegue SDN Overlay en el cual una red lógica corre sobre una infraestructura de red física subyacente mediante el establecimiento de túneles virtuales como VXLAN [50], NVGRE [51] o STT [52].

**Slicing:** Consiste en la segmentación de una red en segmentos o *slices* diferentes las cuales tienen asignados sus propios recursos de ancho de banda, topologías y espacio de direcciones. De esta manera un investigador puede correr sus propios protocolos con controladores y switches bajo su dominio, independientemente de los controladores y switches de otro investigador sobre la misma red. Para implementar la virtualización mediante *slicing* se utilizan controladores especiales

<sup>98</sup> <http://ucb-sts.github.io/sts>

que actúan como un proxy de control SDN para los diferentes *slices* de una red, como FlowVisor [42].

**Aislamiento de Tráfico:** Consiste en la separación del tráfico de clientes en base a alguna política o regla. El aislamiento de tráfico de la red se puede conseguir mediante la utilización del campo VLAN ID de la cabecera de paquetes OpenFlow, a través de *slicing* de red o mediante técnicas de mapeo de redes virtuales a redes físicas a través de bases de datos, como es el caso de FlowN [80].

La selección del método de virtualización dependerá de diversos factores, como el tipo de problema a resolver, las características de la red y el número de redes virtuales a definir. Si se trata de un escenario con pocas redes virtuales se recomienda una solución de *slicing* bajo la plataforma de virtualización FlowVisor. Si por el contrario, se trata de un ambiente donde se requiere un gran número de redes virtuales superior a 100 se recomienda la plataforma de virtualización de red FlowN.

En la Figura 6.6 (tomada de [92]) se muestra la aplicación de un caso de uso de virtualización de un campus universitario. En este caso se crean 4 slices para los departamentos: Medical, Student, PCI-BSS y FacultyStaff y 1 slice para la infraestructura. Las políticas de acceso pueden ser aplicadas desde un controlador central basado en el tipo de departamento o el tipo de acceso, inalámbrico o cableado. Adicionalmente SDN permite aplicar políticas por tipo de aplicación, la cual a su vez permite el acceso a grupos específicos de recursos.

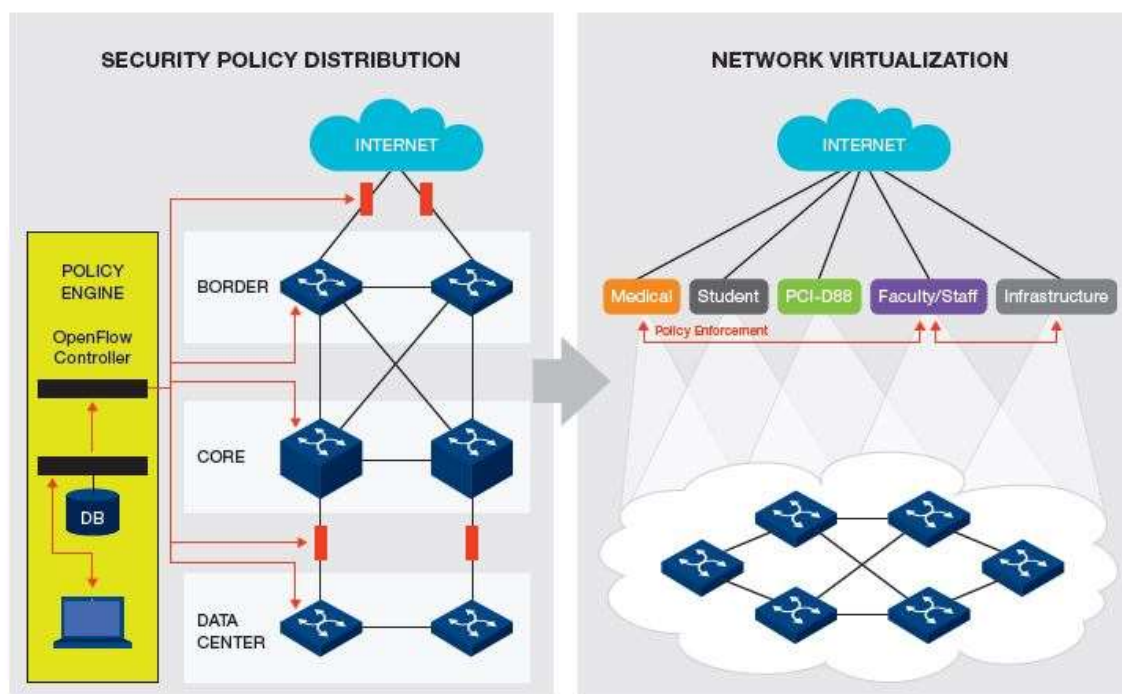


Figura 6.6: Slicing en un Campus Universitario

### 6.10.1 Mejoras en la Seguridad y Aplicación de Políticas

Las redes SDN OpenFlow permiten implementar mecanismos de seguridad y aplicación de políticas mediante la incorporación de módulos de software al controlador de la red. En la siguiente sección se describen algunos ejemplos de casos de usos de estos mecanismos.

#### **Seguridad**

SDN posee cualidades importantes que facilitan la implementación de mecanismos de seguridad: (1) visión global del estado de la red, (2) inteligencia de control centralizada y (3) APIs abiertas para programar el comportamiento de la red. Con estas facilidades SDN se convierte en el punto central para desarrollar aplicaciones de seguridad inteligentes [93] incluyendo filtrado de paquetes [94], control de acceso [95], detección de intrusos y gestión de SLAs [96].

#### **Aplicación de Políticas**

Una gestión de políticas adecuada es de suma importancia en escenarios de redes corporativas y SDN puede ser utilizada para hacer cumplir políticas de red mediante la programación, el monitoreo y la entonación del desempeño de la red. El caso de uso de aplicación de políticas consiste en hacer cumplir políticas de red a usuarios, aplicaciones o dispositivos en base a directrices del grupo IT de una empresa. Un grupo IT puede definir políticas de uso de los servicios de la red como la tasa pico de ancho de banda de un enlace permitida a una aplicación o la hora del día en la cual se pueden acceder a determinadas VLANs o servicios. La aplicación de políticas se puede efectuar mediante un modelo de control similar a Procera [20].

**Procera** es un modelo de control conducido por eventos que intenta resolver tres problemas de gestión fundamentales: (1) permitir cambios frecuentes al estado y las condiciones de la red, (2) proveer el soporte para la configuración de la red en un lenguaje de alto nivel y (3) proveer mayor visibilidad y control sobre tareas para ejecutar diagnósticos y resolución de problemas. Las tecnologías basadas en Procera [20] permiten a los administradores de la red implementar un rango amplio de políticas de red en un lenguaje de políticas de alto nivel e identificar las fuentes de problemas de desempeño. El lenguaje de políticas y el modelo de control de Procera están basados en la programación funcional reactiva FRP [97]. Procera permite a los administradores expresar políticas de alto nivel con este lenguaje y traducir políticas en un conjunto de reglas de reenvío, las cuales son utilizadas para hacer cumplir la política en la infraestructura de red subyacente a través de OpenFlow.

Para expresar las políticas de red conducidas por eventos, Procera ofrece un conjunto de dominios de control que pueden ser utilizados por los administradores de la red para definir ciertas condiciones y asignar acciones de reenvío de paquetes apropiadas a cada condición correspondiente (Ver Tabla 6.5). Actualmente Procera utiliza la especificación OpenFlow versión 1.0.0.

| Dominios de Control | Ejemplos                                                                                                                                                            |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tiempo              | Horas de tráfico pico                                                                                                                                               |
| Uso de los Datos    | Cantidad de uso de datos, tasa de tráfico                                                                                                                           |
| Estatus             | Identidad del dispositivo/usuario, grupo de política, estatus de autenticación                                                                                      |
| Flujo               | Puerto de ingreso, ethernet origen/destino, tipo ethernet, VLAN ID, priority de la VLAN, IP origen/destino, protocolo IP, bits ToS, número de puerto origen/destino |

Tabla 6.5: Dominios de Control en Procera y Políticas de Alto Nivel

### 6.10.2 Movilidad Transparente

Varios esfuerzos han sido enfocados en la conectividad ubicua en el contexto de infraestructuras basadas en redes de acceso inalámbricas. SDN para Wireless provee mecanismos que permiten controlar ajustes de transmisión en base a clientes, slices o flujos y crear optimizaciones con conciencias de aplicaciones y tráfico. En la siguiente sección se revisan algunas propuestas de arquitecturas de redes WiFi SDN inalámbricas incluyendo: OpenRoad [98][99], CloudMac [100] y Odin [101].

### 6.10.3 Redes con Conciencia de Aplicaciones

El caso de uso de conciencia de aplicaciones trata los aspectos relacionados con identificar tráfico importante en la red y redirigir el tráfico a rutas o enlaces basado en requerimientos de desempeño o en el estado de la red. Este caso de uso se apoya en mecanismos de ingeniería de tráfico, balanceo de carga y QoS.

#### Ingeniería de Tráfico

Las aplicaciones de ingeniería de tráfico tratan los aspectos de medir, analizar, regularizar y predecir dinámicamente el comportamiento de los flujos de datos con el objeto de mejorar el desempeño de la red a nivel de tráfico y a nivel de recursos. Entre los requerimientos de desempeño se incluyen el retardo, la variación en el retardo, la pérdida de paquetes, y el rendimiento [102].

SDN provee varios mecanismos que pueden ser utilizados en la ingeniería de tráfico. Específicamente, SDN provee: (1) visibilidad global de la red incluyendo limitaciones de recursos y cambios dinámicos del estado de enlaces de la red, (2) conocimiento de las aplicaciones y sus requerimientos QoS, (3) capacidad de programar la red a través de APIs abiertas, (4) capacidad de reprogramar la red para evitar congestiones y ofrecer desempeño QoS mejorado, (5) capacidad para recopilar estadísticas de los dispositivos SDN, (6) soporte de múltiples tablas de flujos en el pipeline de los switches OpenFlow [103].

**QoS (Calidad de Servicio):** El caso de uso de QoS consiste en garantizar recursos de red incluyendo capacidad de enlaces y asignación de colas de

prioridades a paquetes de flujos con requerimientos estrictos de latencia, ancho de banda y retardo. Un caso común ocurre con la convergencia de servicios de colaboración de audio, videoconferencia y datos sobre una misma fábrica de red. Para soportar todos estos servicios se deben proveer mecanismos QoS que permitan brindar servicios de red diferenciados a usuarios o aplicaciones de red, basado en la prioridad de los paquetes de sus flujos.

El protocolo de comunicación OpenFlow ofrece algunos mecanismos para implementar aplicaciones QoS en la red. En la especificación OpenFlow 1.0 los paquetes pueden ser enviados a colas de puertos de salida mediante la acción *enqueue*, la cual es renombrada a *set\_queue* en la versión 1.3. En la especificación OpenFlow 1.3 se incorporan tablas de medidores que permiten implementar operaciones QoS simples, como limitar la tasa de tráfico a determinados flujos de paquetes. El mecanismo de asignación de flujos a colas se puede combinar con las tablas de medidores para implementar servicios diferenciados más complejos.

Kim y sus coautores [104] proponen un framework de control QoS para fábricas convergentes que permite programar automáticamente y de manera flexible los parámetros QoS de una red OpenFlow 1.0 [54]. El controlador de la red OpenFlow utiliza extensiones del API QoS, que incluye módulos limitadores de la tasa por flujos y módulos de asignación de prioridades dinámicas. El controlador QoS puede crear *slices* de red que permiten asignar tráfico de aplicaciones a *slices* diferentes y aprovisionar los slices dinámicamente para satisfacer los requerimientos de desempeño de todas las aplicaciones en su conjunto. Con la ayuda del controlador QoS los operadores de la red solo tienen que describir especificaciones simples de alto nivel y el controlador QoS se encarga de reservar automáticamente los recursos de red para garantizar el cumplimiento de los requerimientos de desempeño requeridos. Otras implementaciones de QoS para SDN se describen en el estudio de conceptos para mejoras de QoS en SDN [105].

#### 6.10.4 Simplificación de la Gestión

Un controlador SDN tiene la visibilidad completa de la red y es capaz de gestionar el reenvío del tráfico mediante reglas y políticas definidas desde un punto central. Esta facilidad simplifica la gestión de la red ya que solo se deben configurar las reglas de reenvío en un panel central y con el apoyo de herramientas GUI de configuración de políticas y sistemas de gestión centralizados se puede gestionar la red de manera sencilla y confiable sin tocar directamente los dispositivos SDN. Adicionalmente se pueden integrar al controlador herramientas de automatización incluyendo OpenStack<sup>99</sup>, Puppet<sup>100</sup>, Chef<sup>101</sup> o Ansible [36] que minimicen los tiempos de aprovisionamiento de la configuración facilitando la incorporación de nuevas funcionalidades, aplicaciones y servicios.

---

<sup>99</sup> <https://www.sdxcentral.com/cloud/open-source/definitions/openstack-networking>

<sup>100</sup> <https://www.sdxcentral.com/listings/puppet-labs>

<sup>101</sup> <https://www.sdxcentral.com/listings/opscode-inc>

### 6.10.5 Video Streaming/Colaboración

SDN permite construir un árbol multicast entre un proveedor multicast y sus abonados utilizando aplicaciones IP multicast que corren sobre un controlador SDN centralizado que tiene visibilidad completa de la red. La red SDN puede ser programable y ofrece capacidades de despliegue, escalabilidad, adaptabilidad y actualizaciones inmediatas. Noghani y Sunay [107] proponen un *framework* IP multicast para *video streaming* basado en SDN OpenFlow que permite a un controlador SDN implementar IP multicast entre un proveedor multicast y sus abonados.

### 6.10.6 Virtualización de Funciones de Red (NFV)

**Virtualización de Funciones de Red (NFV)** [108][109]: Es la consolidación de funciones de red en servidores estándar de la industria, switches y hardware de almacenamiento localizado en los centros de datos incluyendo funciones de firewall, balanceo de cargas, optimizadores de tráfico, sistemas de monitoreo y otros. A través de NFV se puede reducir o eliminar el hardware propietario específico-a-aplicaciones de la infraestructura de red y ahorrar costos de operación en el despliegue y operación de los servicios de red. NFV provee la misma funcionalidad de un hardware dedicado utilizando equipos virtuales que corren sobre servidores commodities x86. En la Figura 6.7 tomada de [110] se presenta una comparación del modelo de red clásico con respecto al nuevo modelo de servicios de red virtuales.

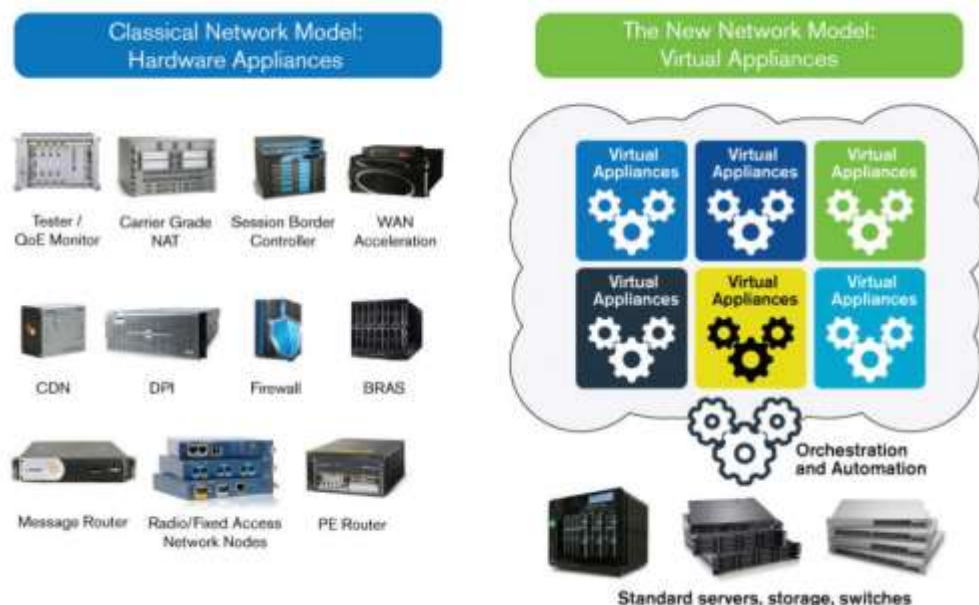


Figura 6.7: Comparación Modelos de Red Clásicos y Equipos Virtuales.

Un caso de uso de SDN y NFV puede ser un banco regional con sucursales interconectadas mediante una red IP-Ethernet que requiere servicios de enrutamiento de borde, encriptamiento y switching Ethernet entre su campus

corporativo y sus sucursales locales y regionales. Bajo un modelo de red tradicional el banco necesitaría adquirir, implementar y operar tres equipos distintos. Bajo un esquema NFV se puede instalar un solo servidor en el campus del banco y descargar y ejecutar tres aplicaciones en el servidor: (1) enrutamiento de borde, (2) encriptamiento y (3) switching Ethernet. En este escenario el controlador SDN se encarga de reenviar el tráfico saliente hacia las sucursales a los puertos de los switches que se conectan al segmento de red de servidores que hospedan los servicios virtuales.

## 7. Marco Metodológico

### 7.1 Metodología

La investigación desarrollada para la realización de esta tesis, se circunscribe al análisis documental [111] que analiza casos de estudio e investigaciones ya realizadas y unifica diversos criterios que se manejan en el campo de las redes SDN y que son relativas a los campus empresariales.

#### 7.1.1 Estudio Teórico

- Análisis del funcionamiento de los planos de control y datos en una red de campus empresarial tradicional.
- Análisis sobre el funcionamiento de los planos de control y datos en una red SDN de campus empresarial.
- Investigación de los aspectos de integración de los componentes de red SDN en una red de campus empresarial.
- Análisis de los principales desafíos para la implementación de redes SDN.
- Investigación de los esquemas de migración hacia redes SDN.
- Estudio de los factores claves para el despliegue de redes SDN en campus empresariales.

#### 7.1.2 Propuesta

Exponer los criterios de diseño para la implementación de redes SDN en campus empresariales.

#### 7.1.3 Diseño de la Investigación

El diseño de la investigación es bibliográfico ya que a través de la revisión de material documental de manera sistemática, se ha llegado al análisis de las consideraciones de diseño para el despliegue de redes SDN y se han determinado sus características y relación de variables.

El plan de acción para alcanzar los objetivos de esta investigación se presentan en el cuadro siguiente:

| Objetivos                                                                   | Acciones                                                                                                                                                               |
|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hacer un análisis general de la forma en que operan las redes tradicionales | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, y libros relacionados con el funcionamiento de las redes de campus tradicionales |

|                                                                                                                                         |                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Analizar el modo de operación de un plano de control centralizado en ambientes SDN                                                      | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, reportes, presentaciones, videos, libros, eventos, y portales Web relacionados con la tecnología objeto del estudio: SDN y OpenFlow      |
| Identificar los componentes necesarios para implementar una arquitectura de red SDN                                                     | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, reportes, presentaciones, videos, libros, eventos, y portales Web relacionados con la tecnología objeto del estudio: SDN y OpenFlow      |
| Describir el funcionamiento y el estado actual de las especificaciones del protocolo de comunicación OpenFlow                           | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, libros y especificaciones técnicas OpenFlow por parte de la ONF                                                                          |
| Investigar el estado actual de los componentes de la arquitectura SDN                                                                   | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, reportes, hojas técnicas, presentaciones, videos, libros, eventos, y portales Web relacionados con la arquitectura de redes SDN OpenFlow |
| Analizar casos de uso de SDN en campus empresariales                                                                                    | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, reportes, presentaciones, videos, libros, eventos, y portales Web relacionados con los casos de uso de SDN y OpenFlow                    |
| Estudiar casos de estudio de implementaciones SDN en campus empresariales                                                               | Recolección de datos bibliográficos mediante consultas a publicaciones, white papers, reportes, presentaciones, y portales Web relacionados con la implementación de redes SDN en campus empresariales                         |
| Determinar las consideraciones y aspectos claves necesarios para garantizar el éxito de una implementación SDN en un campus empresarial | Análisis y Correlación de la información recopilada sobre los desafíos, informes técnicos y lecciones aprendidas en los casos de estudio de despliegues SDN                                                                    |

Tabla 7.1: Diseño de la Investigación

#### 7.1.4 Alcance

Se pretende profundizar el estudio de las consideraciones de diseño para desplegar redes SDN, con la finalidad de precisar los factores claves para implementar despliegues SDN en campus empresariales y migrar de manera parcial o total redes de campus tradicionales a SDN.

#### 7.1.5 Métodos y Técnicas de Recolección de Información

- **Objetivos:** Describe la recopilación de información del negocio del caso de estudio:
  - Visión, metas y objetivos de la nueva red.
- .

- **Información Técnica Pre-Despliegue:** Incluye la recopilación de información técnica del caso de estudio en base a reportes, informes técnicos y publicaciones del grupo IT responsable del despliegue SDN:
  - Cantidad de usuarios.
  - Cantidad de dispositivos de red.
  - Cantidad de puertos y velocidad de puertos.
  - Número de vlans.
  - Direccionamiento IP.
  - Número de aplicaciones y tipo.
  - Identificación de políticas de red: QoS, seguridad y políticas de uso.
  - Recopilación de información sobre el perfil de conocimientos, habilidades y competencias del grupo de administradores IT del caso de estudio.
- **Planificación del Despliegue:** Contempla el levantamiento de información sobre la planificación del despliegue de la nueva red del caso de estudio:
  - Tipo de implementación: Greenfield/Brownfield.
  - Duración del proyecto y fases.
  - Identificación de Prioridades en las aplicaciones, servicios y a aplicaciones a migrar.
- **Información Técnica del Despliegue:** Se enumeran los aspectos técnicos de la red objetivo:
  - Cantidad de usuarios.
  - Cantidad de dispositivos de red.
  - Tipo de tráfico.
  - Tipo de medio de transmisión.
  - Cantidad de puertos, velocidad de puertos.
  - Número de vlans.
  - Direccionamiento IP.
  - Número de aplicaciones y tipo.
- **Información Técnica OpenFlow:** Se indican las características OpenFlow del despliegue:
  - Tipo de despliegue OpenFlow: SDN Basado en Dispositivos/Overlay/Híbrido.
  - Tipo de switches OpenFlow: Solo OpenFlow, Híbrido.
  - Tipo de Implementación de switch OpenFlow: Basado en hardware/Basado en Software.
  - Selección del controlador OpenFlow: Descripción de las características de hardware y software del controlador.
  - Selección de los switches OpenFlow: Descripción de las características de hardware y software de los switches OpenFlow.
  - Selección de funcionalidades OpenFlow: Selección de la versión y características OpenFlow a implementar.
  - Selección de herramientas de monitoreo y métricas.
  - Desempeño del plano de control de la red.
  - Desempeño del plano de datos de la red.
  - Escalabilidad del plano de control OpenFlow/Tamaño de las Tablas de Flujos.

- Políticas QoS de la red.
- Políticas de seguridad de la red: Reglas de firewall,
- Servicios de seguridad de la red: Firewall, aislamiento de tráfico, autenticación de usuarios
- Confiabilidad de la red: Despliegue de un solo controlador o varios controladores.

#### **7.1.6 Métodos y Técnicas de Análisis de la Información**

- Clasificación y análisis cualitativo y cuantitativo de los datos del estudio.
- Analizar la metodología utilizada para implementar la nueva red.
- Analizar los factores de riesgos para implementar la solución.
- Identificar las mejores prácticas y las lecciones aprendidas del caso de estudio.

#### **7.1.7 Procedimientos de la Investigación**

- Captura de los datos del estudio.
- Clasificación de los datos de la investigación.
- Elaboración de informe con análisis de resultados del estudio.

## **8. Marco Referencial**

Entre las implementaciones de redes basadas en OpenFlow se incluyen las redes de campus universitarios, los bancos de pruebas, y las implementaciones efectuadas por la industria. En las siguientes secciones se describen los principales desafíos de SDN y el caso de estudio de la implementación SDN OpenFlow en el campus de la universidad de Stanford.

### **8.1 Desafíos de SDN**

El despliegue de redes SDN en campus empresariales presenta los siguientes desafíos: Desempeño, escalabilidad, disponibilidad y sobrevivencia y costos.

#### **8.1.1 Desempeño**

Los beneficios de las redes SDN vienen acompañados con ciertos desafíos de desempeño: La separación del plano de control y datos sugiere penalidades de desempeño en términos de retardos adicionales para las operaciones de control incluyendo el establecimiento de flujos, el descubrimiento de la topología y la recuperación a fallas [112].

#### **8.1.2 Escalabilidad**

La escalabilidad de una solución SDN puede verse afectada si el controlador SDN recibe muchas solicitudes simultáneas de los dispositivos OpenFlow y se convierte en un cuello de botella. Las restricciones en los recursos de cómputo de los switches OpenFlow también pueden afectar la escalabilidad de la red al contar con un límite en el número de flujos que pueden instanciar en las tablas de flujos y en la capacidad del procesador para instanciar nuevos flujos.

#### **8.1.3 Disponibilidad y Sobrevivencia**

Al desacoplar los planos de datos y control y centralizarlos en un controlador se presenta la condición de un solo punto de falla en la red en el controlador de la red. Si el diseño del despliegue SDN no cuenta con mecanismos de redundancia se puede ver afectado el funcionamiento y la continuidad de la red.

#### **8.1.4 Costos**

En la actualidad existen dos costos recurrentes en las implementaciones de redes: CAPEX y OPEX. Los costos CAPEX están asociados con la adquisición de hardware y software necesario para implementar la red y los costos OPEX están asociados con la operación y mantenimiento de la red. Los costos CAPEX dependen del alcance del despliegue y están relacionados con la cantidad de controladores y switches OpenFlow necesarios. Los costos OPEX están relacionados con el esfuerzo en horas hombres necesarias en las primeras

etapas de la migración o instalación de una nueva infraestructura de red SDN que incluyen: actividades de planificación de la red, ejecución de pruebas de concepto, integración de la red, pruebas y validaciones. Adicionalmente existe un costo asociado al monitoreo de la red y de la salud del controlador para garantizar la disponibilidad de los servicios de red.

## **8.2 Caso de Estudio SDN en el Campus de la Universidad de Stanford**

La universidad de Stanford con el patrocinio de GENI implementó un despliegue SDN en una parte de su campus en el periodo 2010-2013. La experiencia sirvió de base para replicar las mejores prácticas en otros campus universitarios en los Estados Unidos y para mejorar las especificaciones OpenFlow, las capacidades de los componentes de la arquitectura SDN por parte de los fabricantes y el crecimiento del ecosistema SDN de empresas, patrocinantes, desarrolladores de software e investigadores asociados.

### **8.2.1 Objetivos Específicos**

Los principales objetivos del despliegue OpenFlow en Stanford fueron:

- Demostrar las posibilidades de innovación de SDN.
- Permitir la ejecución de experimentos de investigación sobre la red de producción del campus universitario.
- Robustecer y mejorar la especificación OpenFlow y el crecimiento de la tecnología SDN.

### **8.2.2 Etapas de la Implementación SDN en Stanford**

La implementación de SDN en Stanford tuvo un ciclo de maduración de 3 años y contempló 4 fases: (1) Prueba de Concepto, (2) Slicing y Escalabilidad de Despliegues SDN, (3) Implementación Extremo-a-Extremo con Huella nacional y (4) Despliegue de Producción [112]. En la siguiente sección se analizan cada una de las fases y se destacan las lecciones aprendidas en la implementación.

### **8.2.3 Fase 1: Prueba de Concepto**

- **Objetivo:** Se presentaron los siguientes objetivos:
  - Crear un despliegue SDN pequeño utilizando el primer prototipo de switches y controladores OpenFlow disponibles.
  - Construir experimentos y aplicaciones para mostrar el potencial de SDN. Generar puntos de mejoras a la especificación OpenFlow.
  - Atraer más investigadores, operadores de red y vendedores para explorar y desplegar SDN.
- **Planificación del Despliegue:** Se propusieron 3 fases para este despliegue: (1) Construir una pequeña red SDN en laboratorio conformada por 3 switches y 2 APs inalámbricos OpenFlow, (2) ampliar la red SDN del

laboratorio con 3 switches OpenFlow adicionales y (3) interconectar 3 islas OpenFlow incluyendo Stanford, Internet2<sup>102</sup> y JGN2Plus<sup>103</sup>.

- **Información Técnica del Despliegue:**

Tipo de Implementación: Brownfield.

- **Información Técnica OpenFlow:** La implementación contempló los siguientes bloques de construcción:

**Version OpenFlow:** 0.8.1.

**Controlador:** Controlador NOX versión 0.4.

**Aplicaciones:** El controlador contó con aplicaciones internas, incluyendo: Enrutamiento Capa 2 del camino más corto, aprendizaje MAC Capa 2, descubrimiento de la topología basado en LLDP (Link Layer Discovery Protocol) y recolección de estadísticas del switch.

**Infraestructura de Red:** Se construyó una red SDN/OpenFlow pequeña en el laboratorio de la Universidad de Stanford. La red contempló 3 switches OpenFlow prototipo de HP y Cisco y 2 Access Points inalámbricos con software de referencia OpenFlow desarrollados por el grupo de investigación de la universidad (ver Figura 8.1 tomada de [112]). HP y NEC implementaron la segmentación de tráfico OpenFlow y no-OpenFlow mediante el contexto de VLANs. El tráfico OpenFlow es asignado a VLANs pre-especificadas y el tráfico legado es asignado a otras VLANs y se denominó a este enfoque como el modelo híbrido. Los switches OpenFlow fueron asociados al controlador NOX.

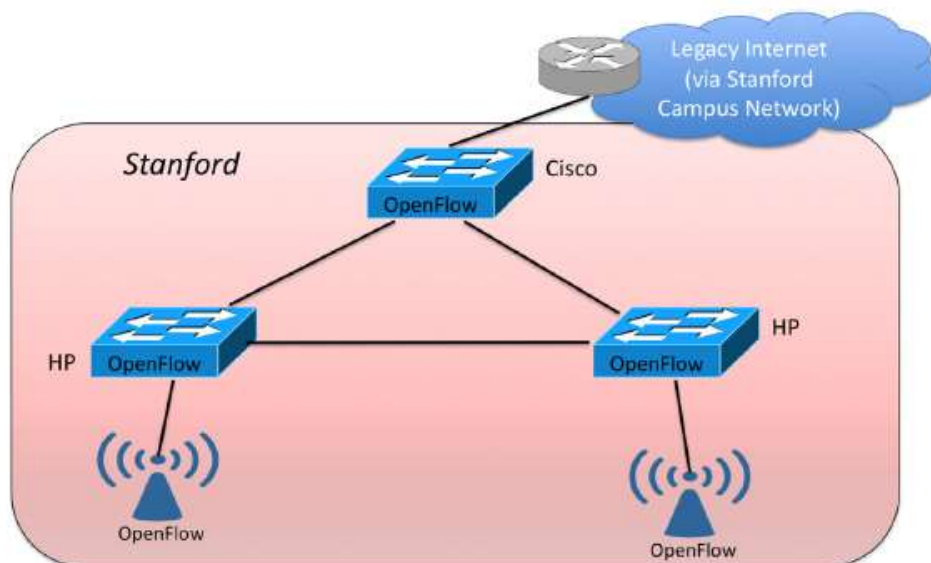


**Figura 8.1: Infraestructura de Red Stanford Fase 1 Prueba de Concepto**

El controlador NOX fue ampliado con módulos adicionales basados en los experimentos implementados durante esta fase.

Posteriormente se incorporaron 4 switches OpenFlow (ver Figura 8.2 tomada de [112]) y se implementaron 3 switches OpenFlow mediante cajas

<sup>102</sup> <https://www.internet2.edu/>

<sup>103</sup> [https://www.jgn.nict.go.jp/jgn2plus\\_archive/english/index.html](https://www.jgn.nict.go.jp/jgn2plus_archive/english/index.html)

NetFPGA<sup>104</sup> en los POPs (Point of Presence) del backbone de la red Internet2. Adicionalmente el grupo de investigadores de Stanford colaboró en la implementación de un switch Juniper MX OpenFlow-híbrido permitiendo crear una pequeña isla OpenFlow en la red Internet2. La red del banco de pruebas avanzadas de Japón para la investigación y el desarrollo JGN2plus también contó con una pequeña isla OpenFlow conformada por switches NEC y cajas NetFPGA. Las 3 islas fueron interconectadas por túneles punto-a-punto (ver Figura 8.2 tomada de [112]) permitiendo crear una red OpenFlow con una pequeña huella global conformada por Stanford, Internet2 y JGN2plus.

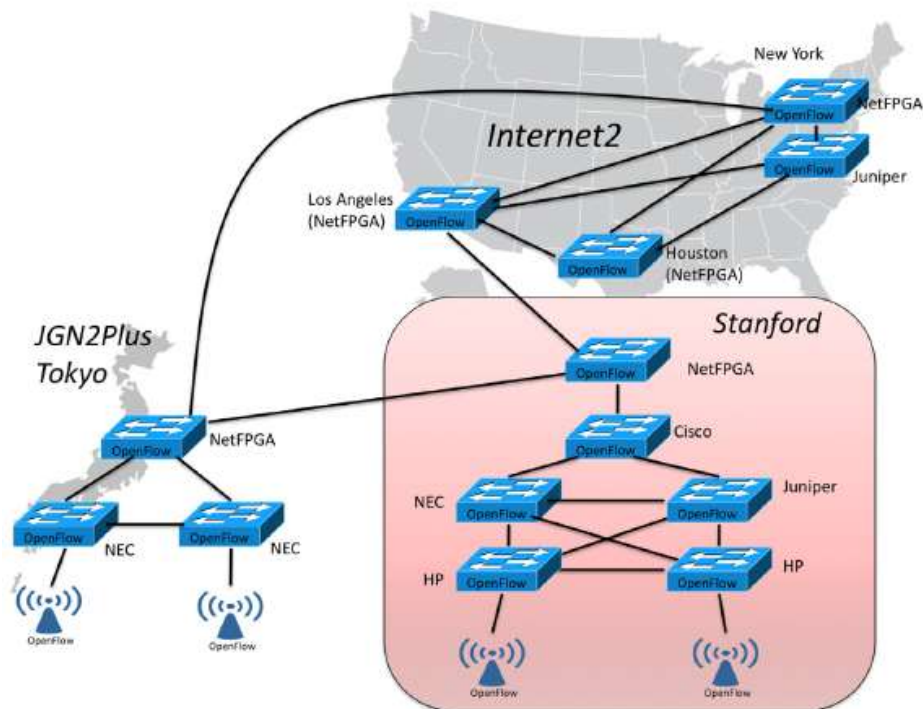


Figura 8.2: Infraestructura Etapa 1 Prueba de Concepto Extendida

**Aplicaciones de Usuario:** Se propuso el experimento movilidad de máquinas virtuales a través de fronteras capa 2 y capa 3 mediante la propiedad de independencia de capas de OpenFlow a través de la combinación de campos de cabecera Capa 2-Capa 4 y la visibilidad global de la red del controlador. La máquina virtual pudo moverse y el controlador pudo reenrutar los flujos a la nueva ubicación con independencia de la ubicación Capa 2 o Capa 3. La capacidad de movilidad de máquinas virtuales se demostró con una aplicación de un juego con múltiples jugadores móviles que se podían desplazar a sitios distintos sin interrumpir la sesión del juego. Para mejorar la experiencia del usuario se logró disminuir la latencia entre los clientes del juego y el servidor backend enrutando el tráfico del juego a través de rutas con el menor número de saltos y moviendo la máquina virtual del servidor del juego a la proximidad

<sup>104</sup> <https://netfpga.org>

del cliente. La movilidad de máquinas virtuales se demostró dentro del campus de Stanford y entre Stanford y Japón mostrando el potencial de SDN [113].

**Lecciones Aprendidas:** Se destacaron las siguientes lecciones aprendidas en términos de desempeño e integración a redes legadas:

**Desempeño SDN:** La métrica tiempo de establecimiento de flujo, definida como el tiempo desde la llegada del primer paquete de un flujo hasta el momento en el cual se ejecuta la acción de reenvío del paquete fue considerada una métrica de desempeño importante. En el modo de control reactivo utilizado frecuentemente en los experimentos académicos esta métrica indica la duración de tiempo que cada flujo necesita esperar en una red OpenFlow para que el tráfico comience a fluir. Esto puede representar una sobrecarga significativa para flujos de corta vida.

La mayoría de los switches de la implementación presentaron bajo desempeño de CPU lo cual representó un impacto negativo en el tiempo de establecimiento de flujos y en el tiempo de respuesta de los mensajes del plano de control que provocaron inclusive caídas de la red en algunas oportunidades. La baja capacidad del CPU puede obedecer a la naturaleza de los switches los cuales fueron concebidos para operar en ambientes de redes legadas con manejo de reenvío de paquetes en base al aprendizaje de direcciones MAC y reenvío IP tradicionales. Estos switches no fueron concebidos para manejar grandes volúmenes de mensajes OpenFlow. Los switches tampoco fueron diseñados para tolerar altos volúmenes de consultas de estadísticas *stats\_requests*.

El modo de control reactivo para el establecimiento de flujos también estresa el CPU afectando la operación del protocolo OpenFlow en general. Esta condición fue solventada en el corto plazo implementando reglas que limitaban la cantidad de mensajes enviados por el canal de control OpenFlow.

**Convivencia de red OpenFlow y red legada:** La implementación de una red conjunta OpenFlow y legada requirió el estudio de aspectos de implementación de los switches, mecanismos de túneles, descubrimiento de topologías basadas en LLDP, overlay routing y retardos TCP. También fue esencial disponer de switches OpenFlow-híbridos.

La interconexión entre las islas OpenFlow Stanford, Internet2 y JGN2Plus se logró mediante mecanismos de túneles. La implementación de túneles en las cajas NetFPGA excedieron el MTU (Maximum Transmission Unit) en varias ocasiones por efectos del encapsulamiento del plano de datos lo cual ocasionó el descarte de paquetes en nodos intermedios de la red. Este problema fue resuelto mediante un mecanismo de túnel basado en software desarrollado en Stanford denominado *Capsulator*<sup>105</sup> y utilizando

---

<sup>105</sup> <https://github.com/peymank/Capsulator>

fragmentación IP a expensas de la penalidad de desempeño por el establecimiento de túneles basado en software.

El descubrimiento de dispositivos de la red se llevó a cabo mediante el protocolo LLDP con una dirección multicast estándar para el dominio no-OpenFlow y una dirección multicast no-estándar para el dominio OpenFlow. El uso de una dirección multicast distinta obedeció a que la implementación OpenFlow de algunos switches descartaban la dirección multicast LLDP estándar.

La creación de enlaces virtuales entre islas OpenFlow mediante VLANs punto-a-punto requirió tomar la precaución de que el enrutamiento Overlay que corre en la aplicación de control no permita a los paquetes atravesar un mismo enlace Capa 2 no-OpenFlow en direcciones diferentes y así evitar que el aprendizaje Capa 2 ocasione una condición de *flap* (puertos continuamente apagados y encendidos) en el switch. Esta situación ocurrió en la Internet2 y fue resuelta desactivando la funcionalidad de aprendizaje MAC en algunos switches legados y haciendo reingeniería de la topología de red.

Se pudo notar que el algoritmo TCP Nagle, el cual está habilitado por defecto, interfirió con el intercambio de mensajes entre el switch OpenFlow y el controlador creando instancias cuando el tiempo del establecimiento de flujos era muy grande. Este aspecto fue resuelto deshabilitando el algoritmo de Nagle utilizando una opción de socket TCP para el canal de control OpenFlow. El algoritmo de Nagle fue concebido para agrupar un número de mensajes pequeños en un buffer antes de enviarlos a una misma conexión incrementando la eficiencia en la transmisión al disminuir el número de paquetes que deben ser enviados, sin embargo, no aplica para el caso del intercambio de mensajes de control OpenFlow ya que los mensajes OpenFlow son generalmente pequeños y críticos en el tiempo y no tiene sentido encolar los paquetes hasta que se alcance el tamaño máximo de un segmento TCP.

▪ **Análisis de Resultados:**

- Se demostró el potencial de SDN para innovar en nuevas aplicaciones y servicios en la aplicación de movilidad de usuarios en la red OpenFlow del campus de Stanford y entre islas OpenFlow.
- Se comprobó la flexibilidad para desplegar una red SDN empresarial en un entorno pequeño y su interconexión con otras islas OpenFlow a través de mecanismos de túneles.
- Se verificó la convivencia de tráfico de producción y experimental en una misma subred SDN/OpenFlow mediante la separación de VLANs para tráfico legado y tráfico no-OpenFlow.
- Se evidenció la influencia de la capacidad de cómputo del CPU de los switches en el funcionamiento de la red OpenFlow. Es conveniente que los fabricantes de red incorporen buenas

capacidades de CPU en los switches para soportar grandes volúmenes de instanciación de flujos y mejorar el tiempo de establecimiento de flujos.

- Se justificó la necesidad de evaluar aspectos de integración con redes legadas incluyendo las condiciones de flapping por efectos del enrutamiento Overlay y la influencia negativa del algoritmo de Nagle.

#### 8.2.4 Fase 2: Slicing y Escalabilidad de Despliegues SDN

- **Objetivo:** Esta etapa estuvo orientada a fomentar el crecimiento del despliegue SDN/OpenFlow y a soportar capacidades de segmentación de red. Se destacaron los siguientes objetivos:
  - Soportar múltiples experimentos concurrentes y tráfico de producción sobre la misma infraestructura física.
  - Evaluar el desempeño de la arquitectura SDN en despliegues reales.
  - Mejorar el software y las herramientas disponibles para el despliegue SDN.
- **Planificación del Despliegue:** Contempló la actualización de los controladores a la implementación de referencia OpenFlow 0.8.9 y la implementación de una capa de virtualización de control.
- **Información Técnica del Despliegue:**

Tipo de Implementación: Brownfield. Contempló la ampliación de la red de laboratorio del edificio William Gates.
- **Información Técnica OpenFlow:**

**Versión OpenFlow:** 0.8.9 liberada en diciembre 2008.

**Controlador:** NOX versión 0.4 para experimentación e investigación y SNAC versión 0.4 para tráfico de producción OpenFlow.

**Funcionalidades:** Se incorporaron nuevas funcionalidades incluyendo soporte de extensiones para vendedores, expiración fuerte de reglas de flujos en la cache, permitir acciones de reemplazo de reglas definidas previamente, proveer concordancia de campos *Code* e *ICMP Type* y actualizaciones menores a la especificación OpenFlow.

**Capa de Virtualización:** Se incorporó una capa de virtualización de control entre los controladores y los dispositivos de red creando redes virtuales denominadas slices. Cada slice se rige por un controlador específico y está asociado a un espacio de flujos definido. Cada experimento fue asociado a un slice independiente que pudo operar con independencia del resto. Se utilizó el controlador Flowvisor 0.3 y 0.4 para la implementación de la segmentación mediante slices.

**Infraestructura de Red:** Se expandió el despliegue del laboratorio de la fase previa incluyendo el sótano y el tercer piso del edificio William Gates utilizando switches NEC, HP y NetFPGA con el nuevo firmware OpenFlow 0.8.9. La infraestructura contó con 4 switches NEC, 2 Cajas NetFPGA y 1 switch OpenFlow-híbrido HP (ver FiguraFigura 8.3 tomada de [112]).

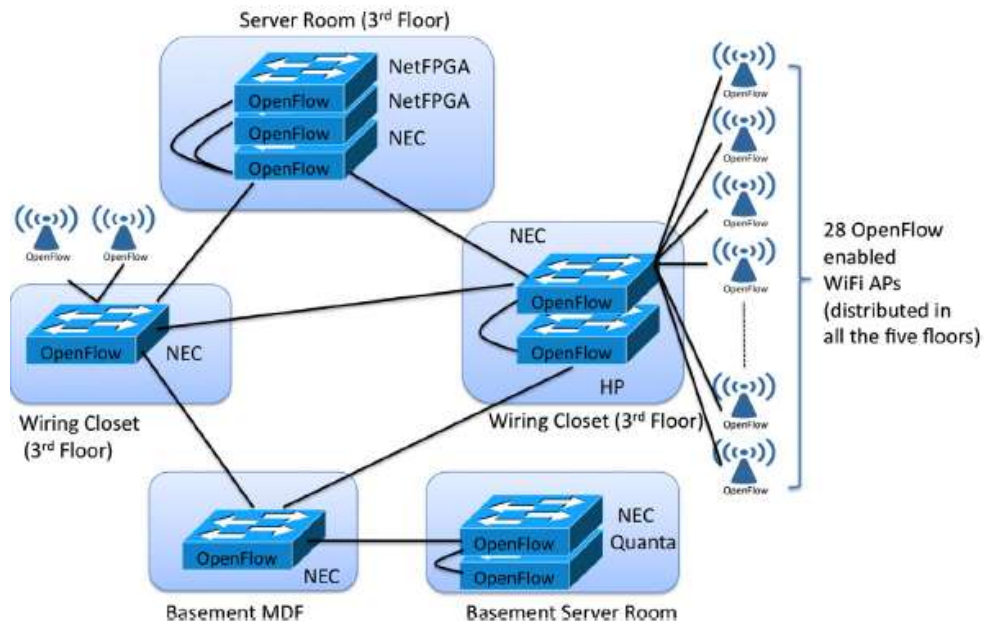


Figura 8.3: Infraestructura de Red Stanford Edificio William Gates Fase 2

En los switches HP y NEC se crearon múltiples instancias de switches OpenFlow mediante la definición de VLANs. Se asoció cada VLAN a un controlador de tal forma que el controlador interpretara que estaba conectado a múltiples switches distintos.

**Aplicaciones de Usuarios:** Se implementaron los siguientes experimentos: (1) *Plug-N-Serve* [114], permite el balanceo de carga web tomando en consideración características de caminos y niveles de congestión en servidores destino. Se utilizó el espacio de flujo conformado por todo el tráfico con número de puerto TCP 80 para la asignación del slice asociado a este experimento. (2) *OpenRoads* [98][99], permite un *handover* sin interrupción entre nodos inalámbricos WiFi y WiMax para una aplicación de streaming de video. El slice tomó el control de todo el tráfico con origen y destino de access points inalámbricos. (3) *Aggregation*, permite agregar múltiples flujos TCP en un solo flujo reprogramando una entrada de la tabla de flujos conformada por flujos componentes que transitan por la red. El slice para este experimento estuvo asociado a la dirección MAC de las estaciones de trabajo del experimento. (4) *OpenPipes*, encapsula tráfico de frames de video en paquetes Ethernet sin procesar y los canaliza a través de varios filtros de video que se ejecutan en nodos de red distribuidos. La política del slice estuvo basada en el campo Ethertype de la cabecera Capa 2. El software SDN se encargó de construir el mapa de red en una estructura de datos, manipular el mapa de red para su control lógico y enviar eventos de actualización de flujos al controlador NOX.

- **Análisis de Resultados:**

- Se evidenció la flexibilidad de SDN para expandir la topología de red y el número de aplicaciones. Se evidenció la capacidad para correr

nuevos experimentos sobre la infraestructura de red y se desarrollaron herramientas de orquestación y depuración para gestión y análisis de indicadores de red.

- Se construyó la implementación de referencia OpenFlow 0.8.9 que permitió a los vendedores desarrollar firmware de switches rápido y estable.
- Se demostró que la segmentación de la red basada en slices provee suficiente flexibilidad para delegar control a diferentes experimentos o entidades administrativas.
- Se justificó la necesidad de implementar un límite a la tasa de flujos por slice para no afectar el tiempo de respuesta en el procesamiento de flujos de experimentos de otros slices. Esta funcionalidad se implementó en la versión 0.4 de Flowvisor.
- Se demostró que pueden ocurrir conflictos en las acciones a ejecutar por 2 controladores distintos cuando existe un solapamiento del mismo espacio de flujos. Esta condición de solapamiento ocurre en las versiones 0.4 y 0.6 de Flowvisor.
- La métrica principal de desempeño de red SDN fue definida como el tiempo de establecimiento de flujo y se recomendaron valores aceptables de 5 a 10 ms para brindar buena experiencia del usuario.

### 8.2.5 Fase 3: Implementación SDN Extremo-a-Extremo

#### ▪ **Objetivos:**

- Apoyar en el despliegue de SDN en 8 universidades de los Estados Unidos, incluyendo: Universidad de Clemson, Instituto de Tecnología de Georgia, Universidad de Indiana, Universidad de Princeton, Universidad de Rutger, Universidad de Stanford, Universidad de Washington y Universidad de Wisconsin) y en GPO (GENI Projects Office), NLR e Internet2.
- Construir una infraestructura SDN extremo-a-extremo funcional con virtualización mediante slices e interfaces GENI.
- Permitir a investigadores experimentar en entornos SDN con huella local y nacional.
- Fomentar el crecimiento de la arquitectura y ecosistema de SDN.

#### ▪ **Información Técnica OpenFlow:**

**Versión OpenFlow:** 1.0

**Funcionalidades destacadas:** (1) Segmentación mediante slices basado en colas conformado por un mecanismo QoS para aislamiento de tráfico en redes OpenFlow, (2) concordancia de direcciones IP en paquetes ARP, (3) cookies de flujos para identificar flujos, (4) estadísticas selectivas de puertos y (5) concordancia de bits ToS en la cabecera IP.

**Controlador:** NOX versión 0.4.

**Capa de Virtualización:** Flowvisor versión 0.6.

**Infraestructura de Red:** Se implementó una pila OpenFlow con virtualización de red mediante slices y un framework de gestión de políticas

conformado por los componentes Expedient<sup>106</sup> y Opt-in Manager para integrarse al framework de control de GENI (ver Figura 8.4 tomada de [112]).

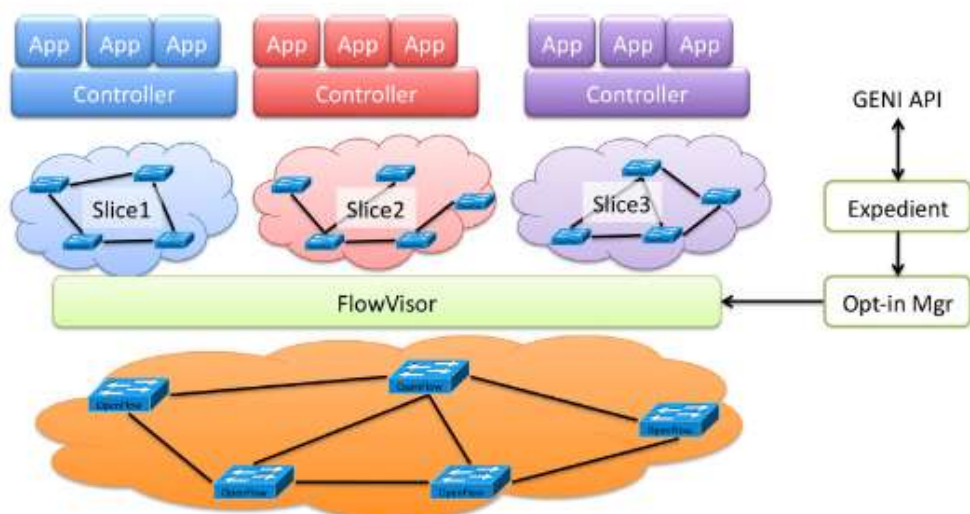


Figura 8.4: Implementación de Capa de Virtualización Stanford Fase 2

**Expedient**, centraliza la gestión de slices y experimentos y la autenticación y autorización de usuarios. Está implementado como un servidor web de tres niveles tradicional con una base de datos relacional como backend y soporta las siguientes APIs: PlanetLab, GENI y Opt-in Manager.

**Opt-in Manager**, interfaz de usuario y base de datos para almacenar y controlar el espacio de flujos de cada usuario y llevar el control de la lista de experimentos y el espacio de flujos de cada experimento.

La infraestructura SDN estuvo conformada por 11 islas OpenFlow interconectadas entre sí incluyendo campus universitarios, la oficina de proyectos GPO y el backbone NLR (ver Figura 8.5 tomada de [112]).

La implementación estuvo basada en una sola infraestructura de cómputo y red programable virtualizable en slices. El despliegue original consistió de una topología de red en estrella utilizando VLANs punto-a-punto sobre el backbone NLR, con Stanford como raíz. Posteriormente se establecieron conexiones de VLANs punto-a-punto directamente al backbone NLR desde otras islas OpenFlow. La Universidad de Princeton se conectó a la Universidad de Stanford mediante el establecimiento de un túnel Capa 2-en-Capa 3 basado en el software de túneles Capsulator.

La infraestructura de interconexión de islas OpenFlow representó una sola red OpenFlow Capa 2 a los investigadores. El tráfico en cada isla se manejó mediante reglas OpenFlow que incluyó campos de cabecera Capa 2-Capa

<sup>106</sup>

<http://yuba.stanford.edu/~jnaous/expedient/docs/api/expedient.clearinghouse.manage-pysrc.html>

4 dirigidas por controladores y las interconexiones entre las islas se efectuó mediante switches legados.

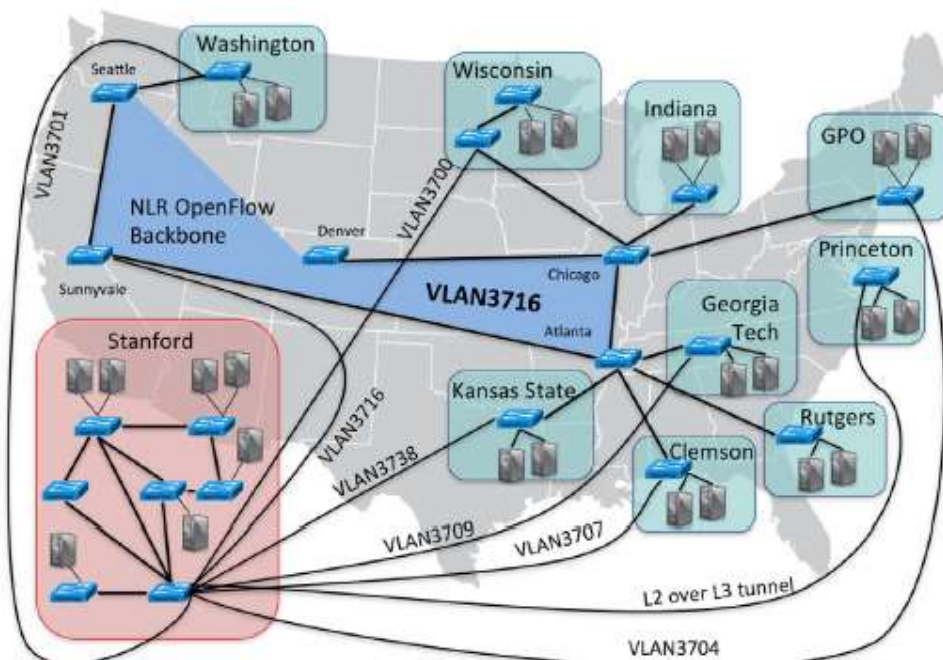


Figura 8.5: Infraestructura de Red Stanford Fase 3

Los investigadores pudieron utilizar la isla OpenFlow global y crear sus propios slices conformados por nodos de cómputo y elementos de red interconectados a través del componente Expedient en Stanford o a través del software Omni [115]. Tanto Expedient como Omni interactuaron con el gestor de agregados Expedient de cada campus para crear slices y el administrador de red local inspeccionó los requerimientos de cada slice con el gestor Opt-in Manager seleccionando los recursos de cada experimento. Se crearon las reglas en Flowvisor mapeando el espacio de flujo al controlador asignado a cada experimento.

**Aplicaciones de Usuario:** Se probó el potencial de OpenFlow corriendo experimentos sobre una infraestructura de red con alcance nacional. Los experimentos más destacados fueron: (1) *Aster\*x*, un balanceador de carga basado en el sistema Plug-N-Serve [114] para operar sobre una infraestructura conformada por servidores y clientes localizados en todas las islas SDN/OpenFlow del despliegue a excepción de la Universidad de Rutgers. (2) *Pathlet*, arquitectura de enrutamiento escalable basada en políticas con enrutamiento controlado desde el origen [116]. La funcionalidad fue implementada sobre el controlador OpenFlow y permitió ejecutar enrutamiento multicamino en dispositivos de borde tomando en consideración el desempeño extremo-a-extremo de la red. (3) *OpenRoads*, experimento basado en el sistema OpenRoads de la fase previa, involucró la transmisión de un stream de video en vivo desde un carrito de golf conduciendo a una velocidad de 5 a 10 mph. El dispositivo pudo utilizar múltiples redes inalámbricas y SDN facilitó la implementación del handover

entre access points WiFi y una estación base WiMax (4) *SmartRE*, framework que permite escalar las capacidades de servicio y red removiendo contenido redundante en la transmisión [117]. El tráfico se generó por un sistema de video en demanda y el sistema utilizó un esquema de coordinación en los dispositivos de red para eliminar la redundancia en las transferencias de red. El controlador OpenFlow y la aplicación facilitaron la gestión de los routers con la funcionalidad de eliminación de redundancia. (5) *Plastic Slices*, experimento para correr 10 slices GENI simultáneos. En cada slice se efectuó una transferencia de archivos entre computadoras virtuales variando la planificación.

▪ **Lecciones Aprendidas:**

**Experimentos con alcance nacional:** Fue necesario un esfuerzo considerable de coordinación y logística entre grupos multidisciplinarios que involucró decenas de personas.

La diversidad de fabricantes de switches OpenFlow-híbrido utilizados requirió de un esfuerzo técnico importante para brindar estabilidad y robustez a la red. La ejecución de las pruebas del proyecto Plastic Slices en una etapa previa hubiese ahorrado un esfuerzo importante en la tarea de estabilidad de la red.

La actualización de la pila de software OpenFlow del despliegue que involucró a Expedient, Opt-in Manager y Flowvisor fue un proceso complicado porque el ciclo de vida de desarrollo de estos componentes era distinto, lo cual no es recomendable. Esto representó mucho esfuerzo de coordinación entre desarrolladores y staff de soporte. Esta tarea requiere de personal especializado en el desarrollo y operación de las herramientas.

**Operación de la red:** Se presentaron varios puntos de mejoras en términos de escalabilidad, ubicación del controlador y definición de espacios de flujos basados en subredes IP.

**Escalabilidad:** La capacidad de procesamiento de la pila de software OpenFlow fue impactada inicialmente. Se corrieron 10 experimentos con un promedio de 1000 reglas OpenFlow. Esto originó la caída del controlador Flowvisor en varias ocasiones. El tiempo de establecimiento de flujos fue demasiado alto en varias oportunidades lo cual requirió de muchas iteraciones de desarrollo en el software del controlador Flowvisor.

El desempeño del CPU de los switches estuvo por debajo de lo esperado para el caso de uso de OpenFlow y se tomaron medidas que involucraron el establecimiento de políticas de límite en la tasa de flujos del plano de datos para mantener la carga del CPU en umbrales aceptables.

Un solo servidor hospedando un controlador para un despliegue de slices a nivel nacional fue suficiente y no presentó cuellos de botella de congestión. Sin embargo, el tiempo del establecimiento de flujos en Princeton basado en un controlador en Stanford fue sub-óptimo. El problema de la ubicación del controlador es un factor clave para brindar desempeño y confiabilidad en estos casos [118].

**Solapamiento de espacios de flujos:** Se debe restringir la asignación libre de espacios de flujos de subredes IP a los investigadores para evitar la condición de solapamiento. En su lugar, es preferible asignar a cada experimento una subred IP particular para evitar el solapamiento de espacio de flujo entre experimentos.

**Formación de bucles de red:** Se pueden presentar bucles en la red cuando una red OpenFlow se conecta a dos redes legadas ya que la especificación OpenFlow utilizada no soporta el protocolo STP. Se debe evitar la formación de bucles en el diseño de la redundancia e la red.

▪ **Análisis de Resultados:**

- El tiempo de establecimiento de flujos constituyó una métrica de desempeño clave en esta etapa y estuvo asociado a la capacidad de CPU de los switches OpenFlow del despliegue.
- Un solo controlador fue suficiente para el manejo de los 10 experimentos del despliegue y no presentó cuellos de botella.
- Para experimentos sensibles a la latencia se puede presentar el problema de la ubicación del controlador el cual puede influenciar y afectar el tiempo de establecimiento de flujos.
- Se debe evitar la formación de bucles de red en la interconexión con redes legadas en la etapa de diseño del despliegue de la red.

## 8.2.6 Fase 4: Despliegue de Producción

La fase de producción contempló el edificio William Gates CS (Computer Science) y el edificio Paul Allen CIS/CIX (Center for Integrated System).

- **Objetivos:** Se plantearon las siguientes metas de red:
  - Disponibilidad de red > 99.9%.
  - Proveer esquema de reversión a red legada en caso de falla.
  - Desempeño de red aproximado al presentado en la red legada.
  - Transparencia en la experiencia del usuario.
- **Información Técnica Pre-Despliegue:** Se contó con la siguiente red inicial:  
**Edificio William Gates CS:** Red legada distribuída en 2 gabinetes conformada por switches HP Procurve. Existencia de VLANs y subredes IP /24 asignadas a grupos de investigación. No existía soporte OpenFlow en el hardware.  
**Edificio Paul Allen CIS/CISX:** Contó con switches HP Procurve desplegados en 6 gabinetes abarcando 4 pisos incluyendo el sótano. No existía soporte OpenFlow previo.  
No existió redundancia a nivel de gabinetes. Los switches se conectaron a 2 switches de distribución en el sótano. Los switches de distribución en el sótano se conectaron a 2 routers de Core Cisco del campus. Todos los switches corrieron STP para evitar bucles. La red estuvo gestionada por el software de código abierto Zenoss<sup>107</sup> junto con configuraciones CLI.

---

<sup>107</sup> <https://www.zenoss.com/product/unified-monitoring/network-monitoring>

- **Planificación del Despliegue:** Se establecieron 3 fases.
  - Fase 1: Crear una red OpenFlow WiFi en el edificio Gates.
  - Fase 2: Construir una red de producción de prueba para un grupo de investigación pequeño (Grupo McKeown).
  - Fase 3: Construir una red de producción en el edificio Paul Allen CIS.
- **Información Técnica del Despliegue:** Se designó la VLAN 74 para la red OpenFlow del edificio William Gates CS con 25 usuarios y la VLAN 98 para la red OpenFlow del edificio Paul Allen CIS/CISX con 50 usuarios.

▪ **Información Técnica OpenFlow:**

**Fase 1, Red WiFi OpenFlow en el edificio William Gates CS.**

Versión OpenFlow: 0.8.9.

Controlador: NOX.

Capa de Virtualización: Flowvisor.

Infraestructura de red: Se construyó la red WiFi OpenFlow en el edificio Gates que formó parte del experimento OpenRoads que formó parte de la fase 2. El despliegue contó con 30 access points WiFi OpenFlow y una estación base WiMax OpenFlow distribuidos en 6 pisos del edificio Gates incluyendo el sótano (ver Figura 8.6 tomada de [119]). Cuatro access points WiFi se conectaron a un switch OpenFlow en el sótano y a 2 switches OpenFlow en el tercer piso. El resto se conectó mediante túneles en software Capsulator a un switch OpenFlow-híbrido. La red WiFi se abrió a usuarios invitados quienes pudieron tener acceso a la Internet pública. Los usuarios podían cambiarse a la WiFi legada cambiando el SSID. Esta red se utilizó para pruebas piloto y depuración de componentes SDN hasta lograr estabilidad. La red WiFi fue madurando en su construcción y solo tuvo caídas planificadas al momento de efectuar actualizaciones de software. La red contó con una capa de virtualización mediante Flowvisor y estuvo dirigida por un controlador NOX.

**Fase 2, Red cableada OpenFlow en el edificio William Gates CS.**

Versión OpenFlow: 1.0

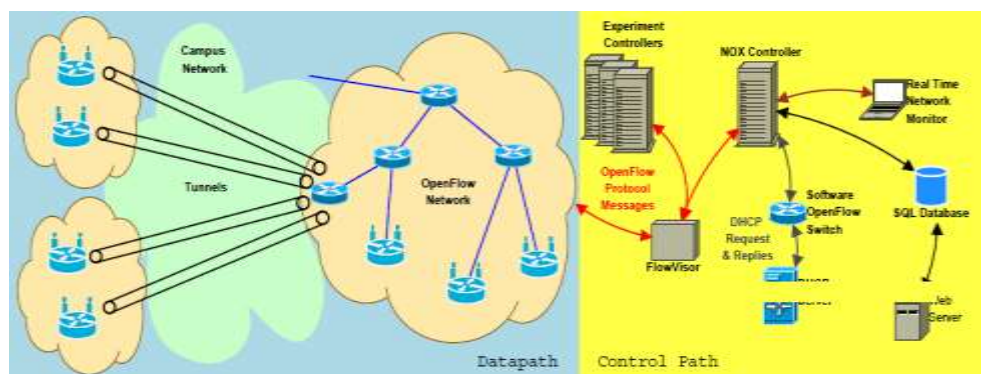


Figura 8.6: Red WiFi OpenRoads en el edificio William Gates

**Controlador:** Inicialmente se utilizó SNAC versión 0.4, luego se migró al controlador comercial BigSwitch.

**Capa de Virtualización:** Flowvisor versión 0.6.

**Infraestructura de red:** Seis switches OpenFlow-híbrido brindaron conectividad al edificio (ver Figura 8.7 tomada de [120]).

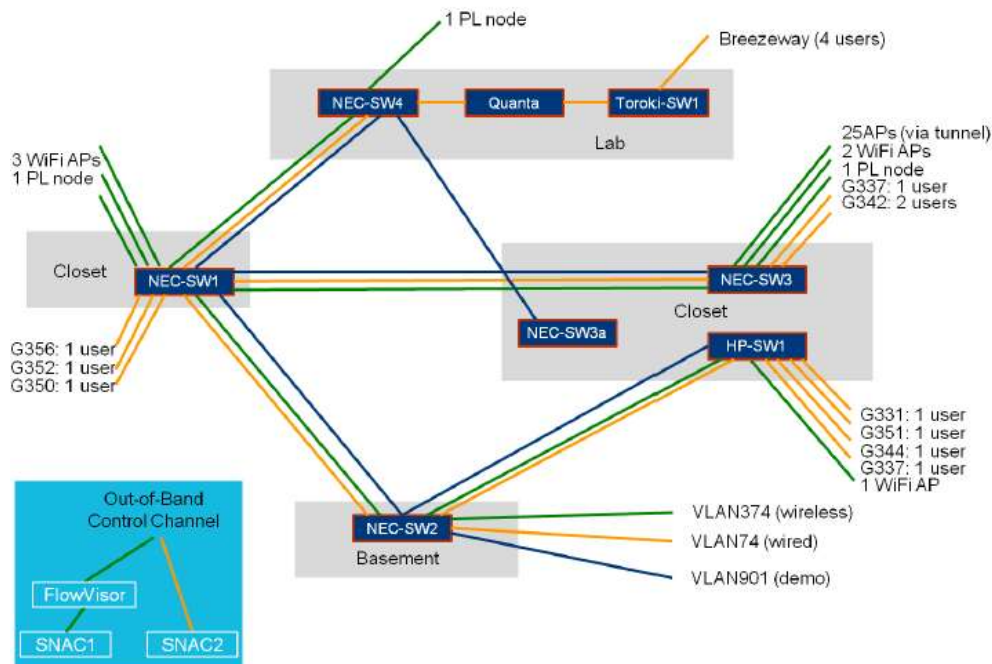


Figura 8.7: Despliegue OpenFlow del Edificio William Gates

Se implementaron los siguientes modelos de switches: HP (ProCurve 5406ZL), NEC (IP8800), Toroki (LS4810) y Pronto (3240 and 3290). Dos switches ubicados en gabinetes separados del piso 3 se conectaron al switch de agregación del sótano. El tráfico de la red legada fue aislado del tráfico OpenFlow mediante la asignación de VLANs separadas.

### Fase 3, Red cableada OpenFlow en el edificio Paul Allen CIS/CISX.

Versión OpenFlow: 1.0

**Infraestructura de red:** Se habilitó el control OpenFlow a la VLAN 98 que abarca todo el edificio. La red contó con 6 switches OpenFlow-híbrido de 48 puertos 1 GE y 14 access points WiFi OpenFlow (ver Figura 8.8 tomada de [120]). La red brindó conectividad a servidores ubicados en salones de clases y acceso a Internet a usuarios móviles. Se incluyeron los siguientes modelos de dispositivos: (1) switches OpenFlow-híbridos de HP (ProCurve 5406ZL y ProCurve 5412ZL) con una VLAN OpenFlow, (2) access points WiFi basados en cajas ALIX PCEngine con interfaces 802.11g.

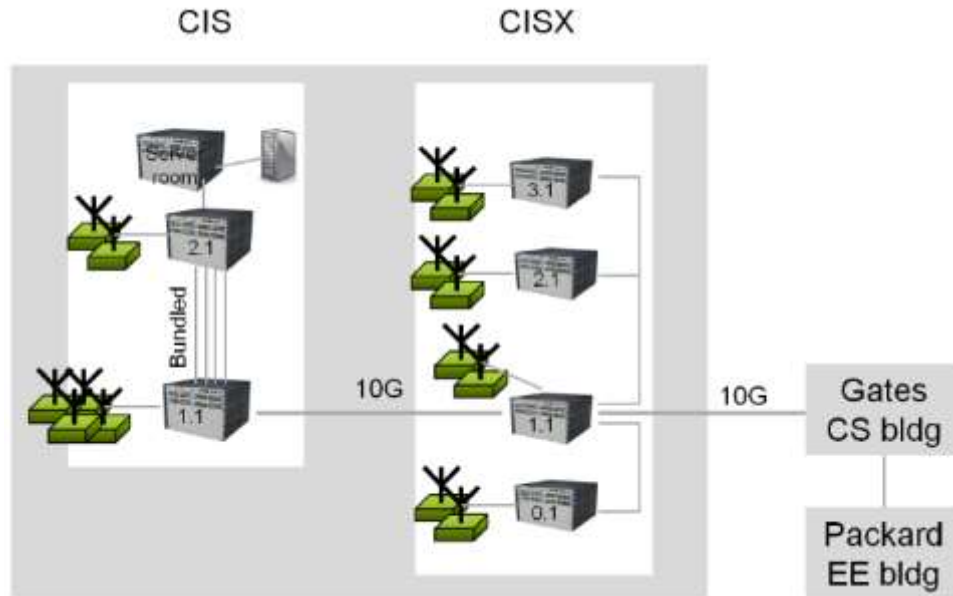


Figura 8.8: Despliegue OpenFlow en el Edificio Paul Allen CIS/CISX

- **Esquema de Migración:** El enfoque adoptado fue mover usuarios seleccionados y VLANs gradualmente al control OpenFlow. La migración se desarrolló en cuatro fases:

**Fase 1: Añadir soporte OpenFlow al hardware.** Se actualizó el firmware de los equipos HP ProCurve, NEC IP8800 y Pronto para proveer soporte OpenFlow.

**Fase 2: Verificar el soporte OpenFlow en los switches.** Se añadió una VLAN experimental gestionada por un controlador externo.

**Fase 3: Migrar usuarios a la nueva red.** Se migraron los usuarios a la subred no-OpenFlow con las siguientes actividades: (1) Crear una subred de Producción, (2) añadir/Mover usuarios gradualmente a la nueva subred y (3) Verificar conectividad dentro de la nueva subred.

**Fase 4: Habilitar OpenFlow en la nueva subred.** Se validó el funcionamiento de la subred y se pasó al control OpenFlow configurando el controlador. Se validaron el funcionamiento, el desempeño y la estabilidad de la red utilizando herramientas de monitoreo y se recopiló información de la experiencia de los usuarios a través de encuestas. La VLAN de producción cableada contempló redes OpenFlow y redes no-OpenFlow y la red WiFi fue gestionada exclusivamente mediante OpenFlow.

**Infraestructura de Monitoreo:** Se implementó una infraestructura de monitoreo de red que constituyó un factor clave en el éxito de la migración (ver Figura 8.9 tomada de [112]). La infraestructura estuvo conformada por nodos con sondas de monitoreo, repositorios de datos y software para gráficos de estadísticas.

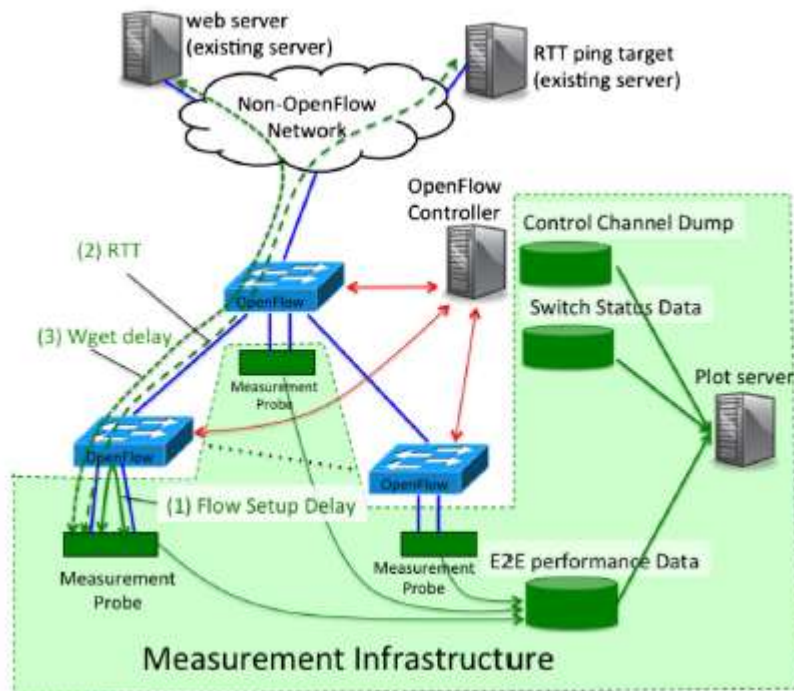


Figura 8.9: Infraestructura de Monitoreo Migración a OpenFlow en Stanford

**Herramientas:** Se utilizaron en su mayoría herramientas de monitoreo estándar y herramientas de depuración, incluyendo:

- **ping:** Utilidad de diagnóstico de una red IP que comprueba el estado de la comunicación entre una máquina local y un dispositivo de red IP externo por medio del envío de paquetes de solicitud *ICMP Echo Request* y recepción de paquetes de respuesta *ICMP Echo Reply*.
- **tcpdump**<sup>108</sup>: Herramienta de línea de comandos para analizar el tráfico que circula por la red capturando y mostrando los paquetes transmitidos y recibidos entre una máquina origen y una máquina destino
- **wget**<sup>109</sup>: Paquete de software libre para recuperar archivos HTTP, HTTPS, FTP y FTPS mediante línea de comandos.
- **oftrace**<sup>110</sup>: Librería de traza y/o analizador del tráfico de control OpenFlow.
- **wireshark dissector para OpenFlow**<sup>111</sup>: Analizador de paquetes de código abierto para identificar información de control del protocolo OpenFlow.
- **mininet**: Paquete de emulación de red que utiliza espacios de nombres de red para ejecutar pruebas de concepto de escenarios de red OpenFlow.
- **ofrewind**: Replicador de eventos de red que reproduce tráfico del plano de datos y de control.

<sup>108</sup> <http://www.tcpdump.org>

<sup>109</sup> <https://www.gnu.org/software/wget>

<sup>110</sup> <https://github.com/capveg/oftrace>

<sup>111</sup> <https://wiki.wireshark.org/OpenFlow>

- **Hassel y NetPlumber**<sup>112</sup>: depurador y verificador de políticas de red en tiempo real.
- **ATPG**<sup>113</sup>: generador de paquetes de pruebas automático para depurar la red.

**Métricas de desempeño:** Se recopilaron 3 conjuntos de datos para el análisis: (1) Plano de control, (2) desempeño extremo-a-extremo y (3) métricas de estado del switch.

**Métricas de desempeño extremo-a-extremo:** Se analizó el siguiente conjunto de métricas:

**Tiempo de establecimiento de flujos.** Se hicieron pruebas de ping entre un par de sondas conectadas a un switch en intervalos de 10 segundos. El intervalo de 10 segundos permitió la expiración de las entradas en la tabla de flujos y cada paquete ICMP generó un mensaje *packet\_in* en el controlador OpenFlow.

**Retardo RTT (Round Trip Time).** Se midió el retardo de un ping desde un nodo origen hasta un nodo objetivo localizado fuera de la red OpenFlow.

**Retardo wget recursivo.** Se midió el retardo para descargar una sola página web, incluyendo sus imágenes y enlaces, utilizando la opción recursiva del comando *wget*.

**Métricas de análisis de switches:** Se estudiaron las siguientes métricas:

**Uso de CPU del switch.** Se estudió el nivel de utilización del CPU de switches seleccionados.

**Cantidad de entradas de flujo.** Se analizó el tamaño de la tabla de flujos de switches seleccionados.

**Tasa de llegada de flujos.** Se observó la tasa de llegada de nuevos flujos.

- **Análisis de Resultados de desempeño:** Se estudió el rendimiento de la nueva red OpenFlow durante una semana en el edificio CIS y se evidenció que el desempeño de la red legada es equivalente al desempeño de la red OpenFlow a excepción del tiempo incurrido en el establecimiento de flujos. Una vez los switches OpenFlow tienen almacenados en su memoria cache las acciones de flujos, el desempeño de ambas redes es similar.

**Tiempo de establecimiento de flujos:** Se observa que el tiempo de establecimiento de flujos está cercano a los 10 ms en promedio lo cual es un buen indicador ya que se encuentra por debajo de un indicador límite de referencia de 100 ms para una buena experiencia de usuario (ver Figura 8.10 tomada de [112]).

---

<sup>112</sup> <https://bitbucket.org/peymank/hassel-public/wiki/Home>

<sup>113</sup> <http://eastzone.github.io/atpg>

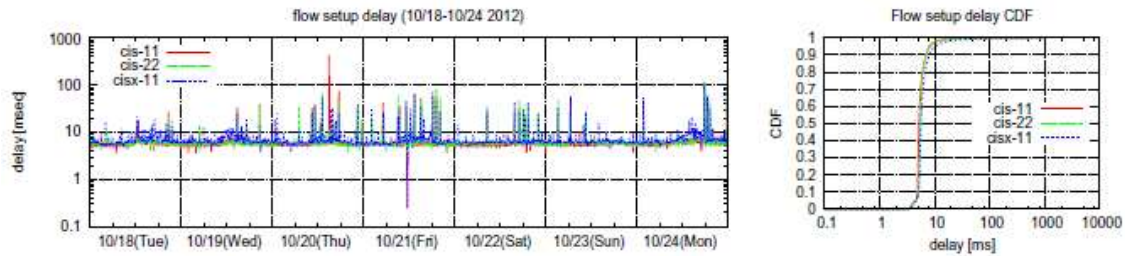


Figura 8.10: Medición Tiempo de Establecimiento de Flujos.

**Retardo RTT:** Se observan valores RTT en el orden de los sub-milisegundos, lo cual es un RTT bajo y buen indicador (ver Figura 8.11 tomada de [112]).

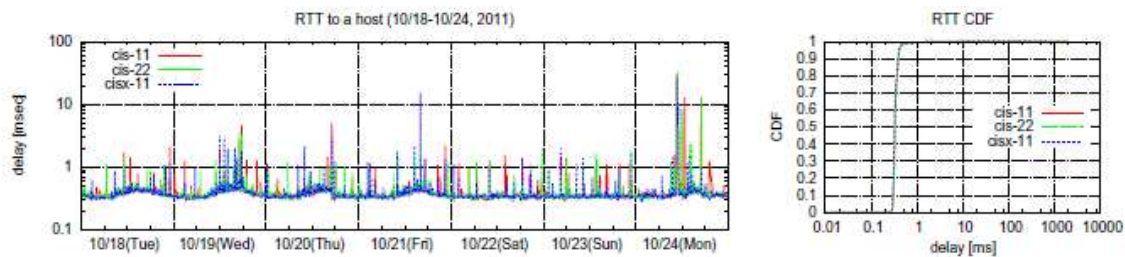


Figura 8.11: Medición Retardo RTT.

**Pérdida de paquetes:** Se muestra un pérdida de paquetes cercana al 0% como se indica en la gráfica CDF (Cumulative Distribution Function) de los valores RTT la cual alcanzó un tope de 1 (ver Figura 8.11 tomada de [112]).

**Retardo wget:** Se aprecia un retardo inferior a 1 segundo para el 80% de las solicitudes web al portal [www.stanford.edu](http://www.stanford.edu) con más de 50 objetos, manifestando un buen indicador en términos de la experiencia de usuario.

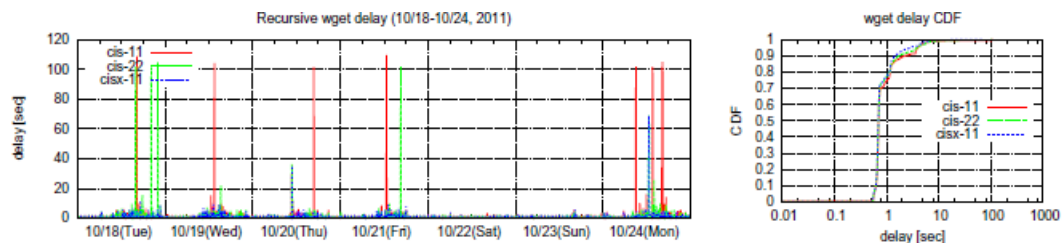


Figura 8.12: Medición Retardo wget.

**Utilización de CPU del switch:** No existe sobrecarga ya que se observa una utilización de CPU por debajo de 70% (ver Figura 8.13 tomada de [112]).

**Cantidad de flujos activos:** Se percibe que el controlador maneja un techo de 320 flujos por switch, lo cual es aceptable considerando la capacidad de los switches HP del despliegue (ver Figura 8.14 tomada de [112]).

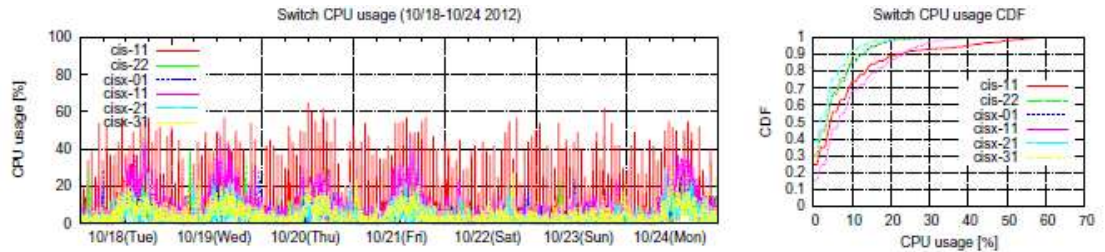


Figura 8.13: Medición Uso de CPU.

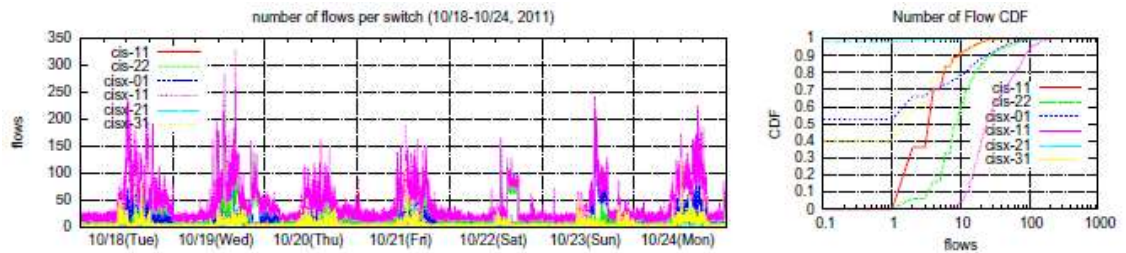


Figura 8.14: Medición Número de Flujos Activos.

**Tasa de flujos:** Se observa la existencia de una tasa de 60 flujos por segundo, la cual es una cantidad manejable por el controlador (ver Figura 8.15 tomada de [112]).

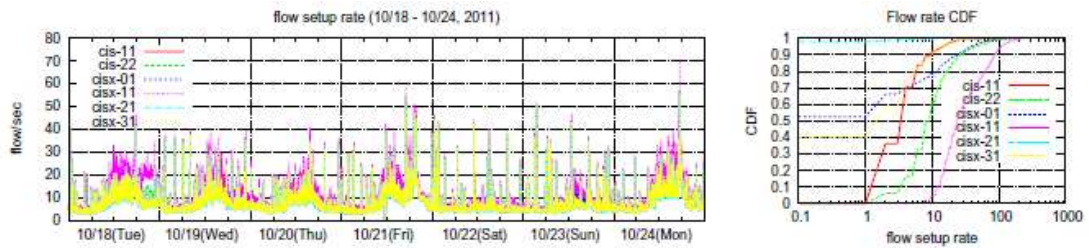


Figura 8.15: Medición Tasa de Llegada de Flujos.

**Cantidad de usuarios WiFi:** Se observa un patrón diurno en la cantidad de usuarios y se obtuvo un tope de 18 usuarios en horas picos, lo cual representa un indicador normal para el tamaño de la red del estudio (ver Figura 8.16 tomada de [112]).

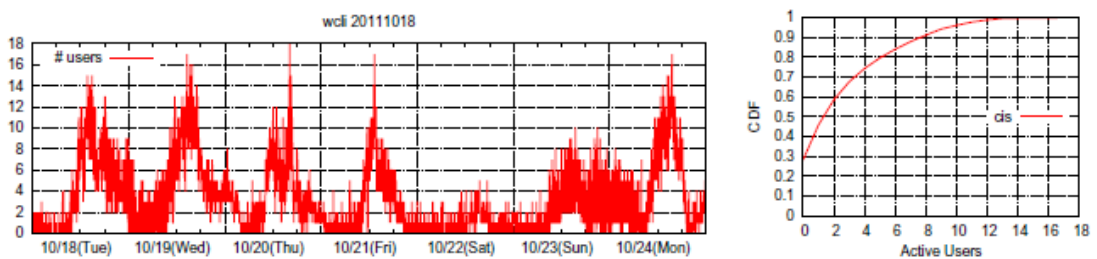


Figura 8.16: Medición Número de Usuarios.

### 8.2.7 Método de diagnóstico de desempeño de la red

De la experiencia en el despliegue SDN en el campus de la Universidad de Stanford quedó un procedimiento de diagnóstico muy útil para implementaciones futuras. El método determina la causa raíz de un problema de red OpenFlow en base a las métricas obtenidas en un análisis de desempeño (ver Tabla 8.1 tomada de [112]).

| Caso | Síntoma |      |      |         |                | Posible Causa Raíz                   |
|------|---------|------|------|---------|----------------|--------------------------------------|
|      | FST     | RTT  | CPU  | #Flujos | Tasa de Flujos |                                      |
| 1    | Alto    | *    | Bajo | *       | Bajo           | Controlador                          |
| 2    | Alto    | Bajo | Alto | *       | Bajo           | El switch está sobrecargado          |
| 3    | Alto    | Bajo | Alto | *       | Alto           | Llegada de flujos en ráfaga          |
| 4    | Alto    | Bajo | Bajo | *       | *              | Software del switch                  |
| 5    | Alto    | Alto | *    | Alto    | Alto           | Desbordamiento de la tabla de flujos |
| 6    | Bajo    | Alto | *    | *       | *              | Software del switch/Controlador      |

Tabla 8.1: Método de Diagnóstico de Desempeño de la Red

- **Caso #1:** Si se observan grandes tiempos de establecimiento de flujos FST (flow setup times) por un período de tiempo prolongado, entonces se debe chequear el gráfico de RTTs de la red no-OpenFlow para determinar si en realidad es un problema de OpenFlow. Si el RTT de la red no-OpenFlow está normal, entonces la causa raíz debe ser la parte OpenFlow y pudiera estar asociado a retardos de procesamiento del switch o del controlador. En la Tabla 8.1 tomada de [112] se muestra el caso #1 donde las métricas uso de CPU y tasa de establecimiento de flujos es baja mientras el tiempo de establecimiento de flujos es alta, por lo cual se determina la causa raíz a un tiempo de procesamiento lento en el controlador.
- **Caso #2:** Si el uso de CPU del switch es alto y la tasa de establecimiento de flujos es baja, entonces el CPU del switch debe estar sobrecargado procesando alguna tarea distinta a mensajes *packet\_in*. La causa raíz puede ser que el controlador se encuentre enviando muchos mensajes al switch o que el switch se encuentre ejecutando otra tarea.
- **Caso #3:** Si el uso de CPU del switch y la tasa de establecimiento de flujos es alta, entonces la causa raíz pudiera obedecer a una tasa de establecimiento de flujos alta. Para identificar el paquete que causa el problema, se debe observar el mensaje *packet\_in* en el canal de control y ubicar los paquetes que generan ráfagas de mensajes *packet\_in*.

- **Caso # 4:** Si el tiempo de establecimiento de flujos es alto y el retardo RTT y el uso de CPU es bajo, entonces la causa raíz del problema debe ser un problema en el software del switch.
- **Caso #5:** Si todas las métricas son altas, la causa raíz obedece a un problema de sobrecarga en la tabla de flujos del switch.
- **Caso #6:** Si el retardo RTT es alto mientras las otras métricas son buenas, entonces esto es un indicador de que una regla no ha sido instalada en la tabla de flujos o que el switch está ejecutando el reenvío en software. Para determinar la causa raíz exacta se debe analizar con mayor detalle el canal de control para verificar si la instalación de la tabla de flujos se hizo correctamente.

### 8.2.8 Depuración de la red

La arquitectura SDN provee información muy útil para la depuración de la red incluyendo: (1) Información de enrutamiento de la tabla de flujos, (2) estadísticas de tráfico a nivel de flujos, (3) secuencia de eventos en la red basado en el intercambio de mensajes de control y (4) topología de red completa. Esta información fue complementada con el análisis del canal de control entre los switches y el controlador y con el desarrollo de herramientas de depuración de red, tales como el conector OpenFlow para la herramienta wireshark.

### 8.2.9 Análisis de Resultados

El primer despliegue SDN en el campus de la Universidad de Stanford tuvo una duración de 3 años y conllevó 4 fases importantes. El despliegue requirió el esfuerzo conjunto de grupos multidisciplinarios incluyendo administradores de red, desarrolladores de software, fabricantes de equipos e investigadores.

El despliegue combinado con las aplicaciones y los experimentos lograron varios hitos importantes que incluyeron la viabilidad de experimentar sobre una red de producción y que culminó con la migración a una red SDN/OpenFlow en una red de producción. El despliegue brindó grandes aportes a la tecnología SDN demostrando la factibilidad de implementar soluciones SDN en entornos de campus universitarios que incluso alcanzaron una dimensión de alcance nacional. El despliegue representa un marco de referencia para implementar soluciones SDN en campus empresariales. El análisis de brechas y los resultados más relevantes se enumeran a continuación:

- En la migración se detectaron los siguientes inconvenientes de interoperabilidad entre la red OpenFlow y la red no-OpenFlow: (1) Los controladores no soportaron STP, (2) el controlador no pudo descubrir los switches no-OpenFlow en la topología de red, (3) los switches no operaron bien con el protocolo LACP (Link Agregation Control Protocol) y (4) no se

pudo obtener visibilidad completa de los flujos y usuarios de ambas redes OpenFlow y no-OpenFlow. Estas limitaciones constituyeron un punto de entrada a las mejoras de la especificación y a nuevos desarrollos que evolucionen la arquitectura.

- Se demostró la factibilidad de correr tráfico de experimentación y de producción sobre una misma red. La implementación de experimentos y demostraciones logró la aceptación de la comunidad en general.
- Se superaron las limitaciones generales que existen al momento de correr experimentos sobre redes de gran escala incluyendo: la capacidad de crear topologías de red arbitrarias, la escalabilidad del experimento y la posibilidad de utilizar tráfico mixto. Estas limitantes fueron superadas con el apoyo de la herramienta de emulación Mininet que permitió ejecutar pruebas de concepto y facilitó a los investigadores y desarrolladores interactuar con una red emulada bajo una pila de control SDN y posteriormente correr experimentos y pruebas sobre una red física real.
- El éxito del despliegue estuvo acompañado de una infraestructura de monitoreo de red que veló por el buen desenvolvimiento de la red en todas las fases del proyecto y aportó la realimentación necesaria para efectuar las mejoras correspondientes a la especificación OpenFlow y a la fabricación de los equipos de red.
- La infraestructura de red fue madurando con el tiempo a través de la interacción del grupo de investigación del proyecto en la Universidad de Stanford y los desarrolladores y fabricantes de equipos de red.
- El resultado final fue un despliegue SDN estable que contó con el aporte de los siguientes contribuyentes:
  - **Investigadores del proyecto:** Llevaron a cabo la implementación, monitoreo y mejoras continuas a la especificación OpenFlow. Se ofrecieron implementaciones de referencia de software a los vendedores, se probaron switches y controladores, se entregaron reportes de fallas, se sugirieron nuevas funcionalidades y se entregó un método para diagnóstico de desempeño.
  - **Investigadores y Experimentadores:** corrieron demostraciones y experimentos importantes sobre redes físicas.
  - **Desarrolladores de software:** Implementaron las mejoras al software SDN.
  - **Fabricantes:** Interactuaron con la Universidad ofreciendo mejoras a sus productos. Participaron fabricantes de switches incluyendo Cisco, HP, Juniper, NEC y Pronto y fabricantes de controladores que incluyeron a Nicira, NEC y BigSwitch. HP ofreció switches OpenFlow-híbridos desde el inicio del despliegue. NEC dispuso switches OpenFlow de alto desempeño y controladores para investigación y desarrollo. Pronto ofreció switches de bajo costo para experimentación. Nicira aportó controladores SDN abiertos, incluyendo NOX y SNAC. BigSwitch suministró un controlador SDN comercial para su uso en la fase de producción.



## 9. Consideraciones de Diseño SDN

En esta sección se presentan las consideraciones de diseño para implementar o migrar redes SDN en campus empresariales, como resultado de las mejores prácticas y las lecciones aprendidas recogidas del caso de estudio y de las fuentes bibliográficas de consulta en el área de investigación de este trabajo. El éxito de una implementación SDN se encuentra adscrito a los siguientes lineamientos generales:

### 9.1.1 Consideración 1: Factibilidad Técnica y Comercial

El primer factor a considerar para desplegar una red SDN en un campus empresarial es la evaluación técnica y comercial de la solución. Es importante definir el valor que aporta la solución al cliente en términos de beneficios operacionales y económicos. La solución debe ajustarse al presupuesto y debe cumplir con aspectos de interoperabilidad a nivel de red, aplicaciones y servicios existentes y futuros. También es de suma importancia considerar la experiencia del usuario tanto del departamento IT como de los usuarios finales de la solución.

El estudio de la factibilidad se inicia con un levantamiento de información de los requerimientos del cliente y un análisis del estado de la infraestructura de red actual. El análisis de requerimientos permitirá definir las metas del negocio y su crecimiento a corto y largo plazo. El análisis de la infraestructura determina la salud actual de la red y la naturaleza de las aplicaciones. Se debe auditar la red en términos de tipos de aplicaciones, volumen de tráfico, y comportamiento.

Desde el punto de vista comercial es importante evaluar el impacto de SDN en los costos de adquisición CAPEX (Capital Expenditure) y en los costos operacionales OPEX (Operational Expenditure). Es conveniente efectuar un análisis de costos y beneficios para determinar si la solución se encuentra alineada con la visión, metas y recursos del negocio. En un entorno SDN actual los principales costos están asociados a los aspectos operacionales de la red y a los costos de adiestramiento en el uso de la tecnología. Los costos CAPEX son relativamente bajos ya que se pueden adquirir switches *comodities* estándar de bajo costo con firmware OpenFlow y hacer un esfuerzo mayor de inversión en el hardware y software de los controladores que rigen el comportamiento de la red. Otro costo asociado consiste en el desarrollo o adquisición de nuevas aplicaciones en aquellos casos de uso donde se requiera la implementación de alguna funcionalidad no nativa a los controladores.

La factibilidad técnica y comercial debe seguir al menos los siguientes puntos:

- La disponibilidad de la red debe ser superior a un objetivo definido. Por ejemplo, en la red de la Universidad de Stanford, la disponibilidad esperada era superior a 99.9%.
- Debe existir un mecanismo de reversión a la red legada en caso de fallas durante la migración.

- El desempeño de la red debe estar cercano o ser superior al desempeño de la red legada.
- La experiencia de los usuarios no debe ser afectada durante la migración.
- La red objetivo debe ser programable mediante APIs abiertos y extensibles.
- La instalación de la red debe ser rápida y sencilla.
- El acceso, uso y gestión de la red debe ser sencillo y ágil. La red objetivo debe ser administrable con software, herramientas y simuladores disponibles.
- La arquitectura de red debe soportar actualizaciones automatizadas de software con mínima interrupción de servicio.
- La red objetivo debe ser interoperable con dispositivos de red de diferentes fabricantes.
- La red de arranque puede requerir la preparación y transformación a un estado intermedio seguro desde el cual se pueda proceder a una migración total de la solución.
- La red final debe ser validada con respecto a los requerimientos y expectativas del negocio.
- Los costos de la solución deben estar dentro del presupuesto asignado al proyecto.

### 9.1.2 Consideración 2: Modelo de Despliegue

La selección del modelo de despliegue es un factor determinante en los costos y la gestión de riesgo de la solución. El modelo Basado en Dispositivos SDN debe implementarse en despliegues Greenfield, mientras que los modelos SDN Overlay o SDN Híbridos deben implementarse en despliegues mixtos. Los despliegues Mixtos requieren mayor cantidad de personal especializado y mayor coordinación de personal de redes legadas y OpenFlow en aspectos de integración, implementación, gestión y seguridad de red. El modelo Basado en Dispositivos SDN requiere de una mayor inversión de recursos en la etapa inicial en adiestramiento de personal y monitoreo de desempeño, sin embargo, a medida que evoluciona el funcionamiento de la red y el dominio de la tecnología, disminuyen paulatinamente los costos de operación.

**Tipos de Despliegues SDN:** Existen 2 modalidades de despliegue dependiendo del tipo de infraestructura del cliente. Para implementaciones de una nueva infraestructura se considera un despliegues Greenfield y para implementaciones con equipos legados se consideran despliegues Brownfield.

**Despliegue Greenfield:** La ONF denota al modelo Basado en Dispositivos SDN con el nombre de despliegue Greenfield (ver Figura 9.1 tomada de [120]). En este caso la red existente es actualizada a OpenFlow y la red es controlada en base a las reglas de reenvío definidas en un controlador. Los dispositivos SDN pueden ser switches OpenFlow nativos o dispositivos que aceptan un conector OpenFlow. La implementación consiste en establecer la conexión de red entre el controlador y los dispositivos de red. El controlador se encarga de hacer el descubrimiento de la red y de publicar las reglas de las tablas de flujos a los dispositivos de red.

Posteriormente se definen las reglas de filtrado y conectividad específicas en el controlador en base a las políticas de red de seguridad y calidad de servicio recopiladas durante la fase de preparación.

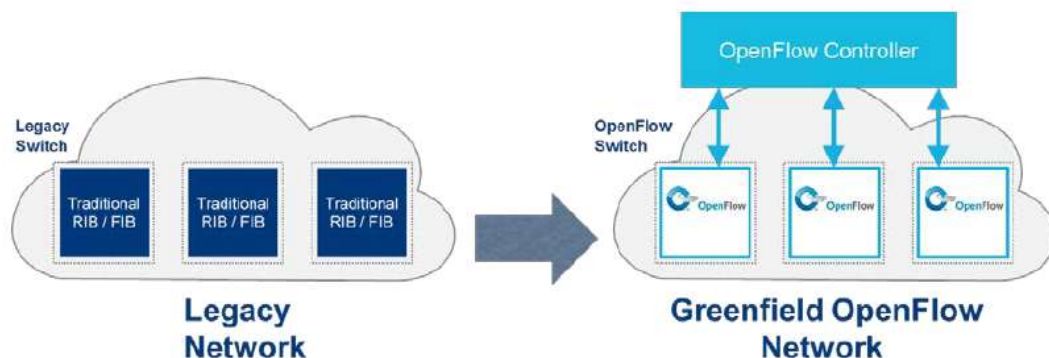


Figura 9.1: Despliegue de Migración Greenfield

**Despliegue Mixto (Mixed):** La ONF denota el modelo SDN Híbrido con el nombre Despliegue Mixto (ver Figura 9.2 tomada de [120]) en el cual conviven dispositivos OpenFlow nativos con dispositivos legados. En este caso el controlador SDN/OpenFlow y los dispositivos legados intercambian información de enrutamiento entre sí a través de protocolos de red tradicionales.

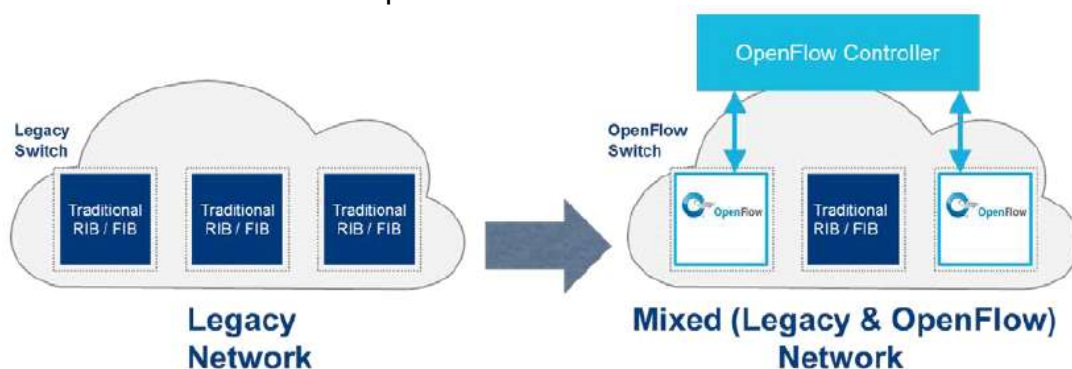


Figura 9.2: Despliegue de Migración Mixto

**Despliegue Híbrido:** Es una variante del despliegue Mixto en el cual conviven nodos legados con conectores OpenFlow con nodos legados (ver Figura 9.3 tomada de [120]).

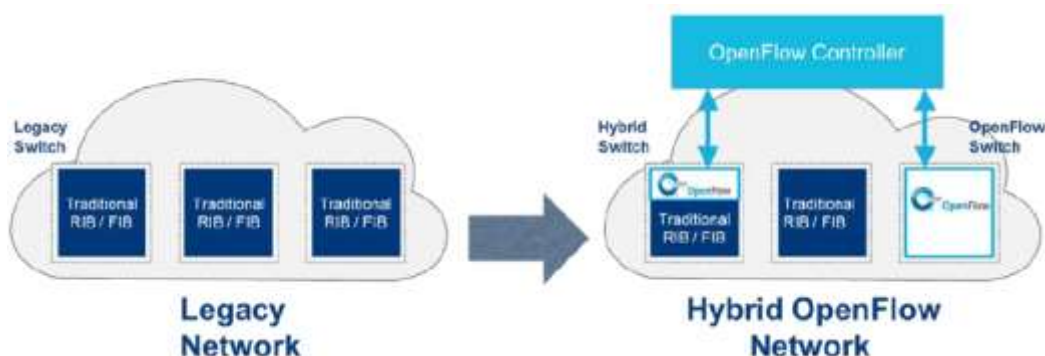


Figura 9.3: Despliegue de Migración Híbrido

### 9.1.3 Consideración 3: Hoja de Ruta

Es recomendable diseñar una hoja de ruta o plan estratégico que establezca las metas o resultados deseados e incluya los principales pasos o hitos necesarios para alcanzarlos. En general un despliegue SDN consta de tres fases importantes: (1) Preparación, (2) Implementación y (3) Validación.

- **Preparación:** La fase de preparación contempla: (1) el análisis de los requerimientos del negocio, (2) el diseño de alto nivel de la solución, (3) la preparación de las pruebas de concepto, (4) el diseño de bajo nivel de la solución y (4) la planificación de la implementación.

**Análisis de Requerimientos:** Esta etapa consiste en la identificación de las metas y objetivos del negocio y el reconocimiento del estado actual de la red. Particularmente se debe recopilar y analizar la siguiente información: (1) metas del negocio, (2) caracterización de las aplicaciones del negocio y de los servicios de red incluyendo: tipo de aplicación, volumen, prioridades, requerimientos de ancho de banda y reglas de acceso, (3) topología de red, incluyendo usuarios, sistemas, servidores, equipos de red, tipo y velocidad de enlaces de conexión, (4) identificación de segmentos de red, incluyendo definiciones de VLANs, direccionamiento IP y protocolos de red, (5) identificación de políticas del negocio, políticas de seguridad, políticas de calidad de servicio y normas de cumplimiento como regulaciones y (6) identificación de los sistemas de gestión de la red.

La evaluación de la red se puede hacer levantando la información con informes del cliente y corriendo herramientas de monitoreo y análisis de tráfico que permitan identificar métricas de utilización de la red y las aplicaciones, usuarios y servicios con mayor uso de servicios de red. Esta etapa también contempla la identificación de riesgos, cuellos de botella de la red y problemas de desempeño.

**Diseño de Alto Nivel:** Se plantea un diagrama de alto nivel de la solución identificando los componentes de la arquitectura de red incluyendo: controladores, dispositivos de red, aplicaciones y sistemas de gestión. La selección de los controladores y los dispositivos de red se realizan en base a los resultados del análisis de los requerimientos y considera aspectos como costos, facilidad de uso, desempeño, soporte del fabricante y de la comunidad y APIs soportadas. Se debe seleccionar el tipo de switch OpenFlow entre OpenFlow nativo, híbrido o virtual, dependiendo del modelo de despliegue de la solución.

**Pruebas de Concepto:** Consiste en la preparación de un banco de pruebas para determinar si la solución a implementar satisface las metas del negocio. La implementación del banco de pruebas se puede realizar mediante la utilización de herramientas de simulación de código libre

incluyendo Mininet [65], VT-Mininet y OFNet [70] o herramientas comerciales como EstiNet [67]. La herramienta OFNet permite hacer emulaciones gráficas de red y provee capacidades de depuración visual, capacidades de monitoreo de desempeño y capacidades de generación de tráfico sintético que facilitan la depuración de la red y la optimización del diseño de la solución en un ambiente controlado.

**Diseño de Bajo Nivel:** Consiste en la definición de las reglas de control de tráfico que serán implementadas en el controlador SDN/OpenFlow en base a los requerimientos y las políticas del negocio. También incluye la activación de servicios de red en el controlador, la configuración de la capa de virtualización de la red y la habilitación de la funcionalidad OpenFlow en los dispositivos de red.

**Planificación:** Corresponde al establecimiento de la línea de tiempo de las fases de ejecución del despliegue y los entregables e informes de cada fase.

- **Implementación:** La fase de implementación consiste en la migración de la red legada, los usuarios y servicios a la red SDN/OpenFlow. El esquema de migración dependerá del análisis de requerimientos donde se determina el modelo de despliegue SDN. En el caso de una red OpenFlow totalmente nueva, se implementa el modelo Basado en Dispositivos SDN. En el caso de escenarios de migración de una red legada se implementa el modelo SDN Híbrido.

**Enfoque de Migración:** El enfoque de migración recomendado por la ONF consiste en mover usuarios y VLANs individuales a la red de control OpenFlow para gestionar el riesgo asociado en desplegar la nueva tecnología. Se consideran 4 fases fundamentales:

- Añadir soporte OpenFlow al hardware: Se hace una actualización del firmware a los dispositivos de red para soportar OpenFlow.
  - Verificar el soporte OpenFlow en los dispositivos de red: Se verifica la funcionalidad OpenFlow incorporando una VLAN experimental gestionada por el controlador SDN.
  - Migración de usuarios: Esta tarea involucra las siguientes actividades: (1) Añadir una nueva subred de producción, (2) añadir o mover usuarios gradualmente a la nueva subred, (3) verificar la conectividad dentro de la nueva subred y (4) habilitar OpenFlow en la nueva subred. Una vez verificado el funcionamiento de la nueva subred, se habilita el control OpenFlow a la subred configurando las reglas correspondientes en el controlador.
- **Validación:** Representa una fase fundamental en el proceso ya que permite verificar el cumplimiento de las metas del negocio y los objetivos del diseño. Para efectuar la validación, el despliegue debe contar con una infraestructura de monitoreo que permita recopilar, analizar y verificar el

correcto funcionamiento OpenFlow y la conectividad, desempeño y estabilidad de la nueva red.

#### 9.1.4 Consideración 4: Selección del Controlador

El controlador es el componente fundamental en una arquitectura de red SDN ya que tiene la responsabilidad de dirigir y vigilar el comportamiento de la red. La selección del controlador tiene un impacto directo en la operación, escalabilidad, confiabilidad y desempeño de la red. La selección del controlador dependerá de los costos y de los requerimientos planteados y debe estar alineada con las metas del negocio en términos de servicios, funcionalidades y crecimiento.

La Figura 9.4 tomada de [121] muestra aspectos de evaluación de controladores SDN separados en cuatro bloques: Funcionalidades, eficacia, casos de uso y redes de aplicación. El bloque funcionalidades considera las características de implementación del controlador e influencia directamente a los otros bloques incluyendo la escalabilidad, extensibilidad, programabilidad e interoperabilidad [121]. El bloque eficacia compara aspectos del funcionamiento del controlador en términos de desempeño, confiabilidad, escalabilidad y seguridad.

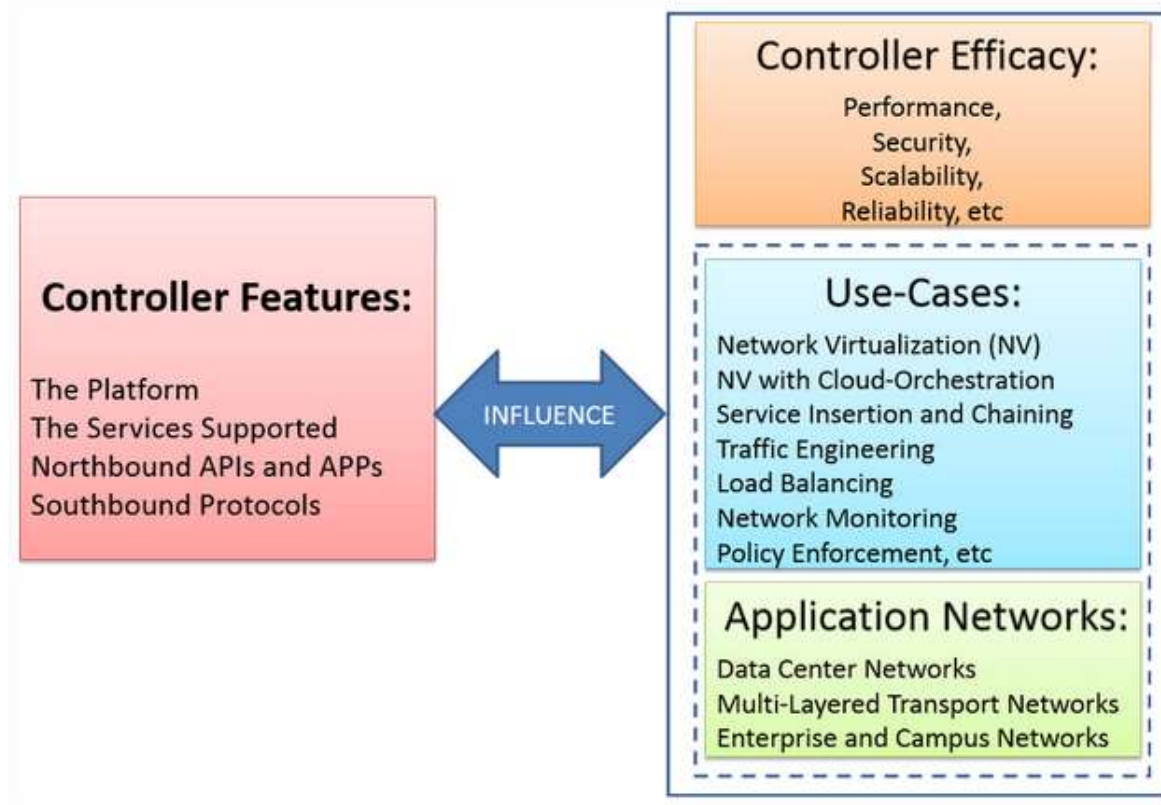


Figura 9.4: Diferentes Aspectos para Seleccionar Controladores SDN

El bloque casos de uso establece una relación entre el caso de uso y el controlador que lo soporta. El bloque redes de aplicación ubica al controlador en

un dominio de aplicación que puede ser centros de datos, redes de transporte para operadoras de servicio o redes de campus empresariales.

**Esquemas de Selección de Controladores SDN:** De la La Figura 9.4 tomada de [121] se desprenden tres esquemas de selección de controladores:

- Selección de Controladores basado en Aspectos de Implementación.
- Selección de Controladores basado en Aspectos de Eficacia.
- Selección de Controladores basado en Casos de Uso.

**Selección de Controladores basado en Aspectos de Implementación:** Toma en consideración las características técnicas del controlador. En diciembre del 2013 Khondoker, Zaalouk, Marx y Bayarou, desarrollaron una plantilla línea base (ver Tabla 9.1) para seleccionar controladores SDN [74]. La plantilla evalúa aspectos técnicos relacionados con la facilidad de uso, lenguaje de desarrollo, plataforma soportada y documentación. La plantilla fue revisada en el estudio de Centeno y sus coautores [75] y actualizada en este documento (ver Tabla 9.1). La plantilla limita el estudio a los siguientes controladores: Beacon, Floodligh, NOX, POX, Trema, Ryu y OpenDaylight.

|                        | Beacon                           | Floodlight             | NOX                    | POX                    | Trema                                | Ryu                                                  | ODL                    |
|------------------------|----------------------------------|------------------------|------------------------|------------------------|--------------------------------------|------------------------------------------------------|------------------------|
| Soporte OpenFlow       | OF v1.0, v1.3                    | OF v1.0, v1.3          | OF v1.0                | OF v1.0                | OF v1.3                              | OF v1.0, v1.2, v1.3, v1.4, v1.5 y extensiones Nicira | OF v1.0, v1.3          |
| Virtualización         | Mininet y Open vSwitch           | Mininet y Open vSwitch | Mininet y Open vSwitch | Mininet y Open vSwitch | Incluye un emulador de red integrado | Mininet y Open vSwitch                               | Mininet y Open vSwitch |
| Lenguaje de desarrollo | Java                             | Java                   | C++                    | Python                 | Ruby/C                               | Python                                               | Java                   |
| Provee API REST        | No                               | Si                     | No                     | No                     | Si (Básica)                          | Si (Básica)                                          | Si                     |
| Interfaz Gráfica       | Web                              | Web                    | Python+. QT4, Web      | Python+. 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 OpenStack | No     | Si     | No     | No     | Si     | Si     | Si     |
| Multiprocesos        | Si     | Si     | Si     | No     | Si     | No     | Si     |
| Código Abierto       | Si     | Si     | Si     | Si     | Si     | Si     | Si     |
| Tiempo en el Mercado | 6 años | 4 años | 8 años | 5 años | 7 años | 3 años | 3 años |
| Documentación        | Buena  | Buena  | Media  | Pobre  | Media  | Media  | Media  |

Tabla 9.1: Plantilla Línea Base para Seleccionar Controladores

- **Soporte OpenFlow:** Los administradores de la red necesitan entender las funcionalidades soportadas en la versión OpenFlow del controlador (ver Tabla 5.9) para obtener mayor provecho de la arquitectura de red y mayor conocimiento de las opciones y extensiones disponibles que puedan aplicarse al caso de uso del negocio. También es importante considerar la hoja de ruta del vendedor en el soporte de nuevas versiones y funcionalidades para garantizar el tiempo de vida de la solución [122].
- **Virtualización de Red:** Contempla la capacidad de virtualizar la red física en redes virtuales que pueden estar asociadas a usuarios, tráfico o servicios particulares. La virtualización de red permite asignar políticas a segmentos de red específicos permitiendo el aprovisionamiento de recursos diferenciado a usuarios. El controlador puede contar con mecanismos de virtualización nativos o integrarse a controladores de virtualización como Flowvisor. Otra característica relacionada con la virtualización de red es la capacidad de integrarse a herramientas de emulación de red como Mininet. Esta funcionalidad permite efectuar pruebas de concepto en ambientes de laboratorio controlados con diferentes topologías y evaluar el funcionamiento de la red en una etapa previa al despliegue de la red final.
- **Lenguaje de desarrollo:** El hecho de que el controlador se encuentre construido en software confirma la importancia del lenguaje de programación en la selección del controlador. Java es un lenguaje predilecto debido a su capacidad multihilo y a su portabilidad. Python presenta problemas con el soporte multihilo en su nivel de desempeño, C y C++ tiene problemas con la gestión de memoria y los lenguajes .NET son dependientes de la plataforma y no soportan sistemas Linux [123]. Los controladores basados en Java son multiplataforma y presentan buena modularidad, los controladores basados en C proveen alto desempeño pero carecen de buena modularidad, buena gestión de memoria y buena interfaz de usuario y los controladores basados en Python carecen de un manejo multihilo real. Java es un lenguaje de programación fácil de codificar, mantener, depurar y asegurar. Además puede soportar la carga dinámica de aplicaciones y estructuras de datos sin recompilación ni reinicio del sistema completo. Otra característica fundamental de Java es su portabilidad ya que se puede migrar con facilidad a diferentes plataformas que soporten máquinas virtuales Java [124].

- **Soporte API REST:** El soporte de APIs hacia el norte constituye una característica importante de un controlador ya que determina la facilidad en el desarrollo de nuevas funcionalidades en el controlador y la integración con aplicaciones de terceros. El API REST es generalmente el API más recomendado por sus características de simplicidad, flexibilidad, extensibilidad y seguridad. A través de invocaciones simples HTTP y HTTPS se pueden acceder con facilidad a los recursos de la red.
- **Interfaz Gráfica:** La interfaz gráfica desarrollada para un controlador determina la facilidad de uso de la solución SDN e impacta los costos de operación y mantenimiento de la red. Una interfaz gráfica simple, intuitiva que permita la definición rápida de reglas y el monitoreo de estadísticas OpenFlow y la visualización gráfica de la topología de red desde un panel central favorecen la opción de un controlador particular.
- **Soporte de plataformas:** La plataforma de red sobre la cual está soportado un controlador puede definir la selección de un controlador específico. Se tiene preferencia por controladores más universales que puedan correr sobre múltiples plataformas y sistemas operativos. Esta característica también depende del conocimiento y uso de la plataforma actual en la cual se encuentra operando una red ya que esto impacta en el tiempo de adopción de la nueva tecnología.
- **Soporte de OpenStack:** El sistema operativo para nube OpenStack permite gestionar recursos de cómputo, red y almacenamiento desde un panel central. La capacidad de integración del controlador con OpenStack permite soportar funcionalidades y servicios orientados a la nube que facilitará la gestión de la red en entornos de nubes basados en este sistema operativo.
- **Soporte Multiprocesos:** La incorporación del soporte multihilo en la arquitectura de un controlador impacta su desempeño de manera directa. En la Figura 9.5 tomada de [125] se muestra la arquitectura de un controlador multihilo que corre una instancia de la aplicación en cada hilo independiente. La capacidad multihilo mejora significativamente el desempeño del controlador. En la medida que aumenta la cantidad de hilos se pueden crear más instancias de controlador y atender mayor cantidad de solicitudes de mensajes OpenFlow de los switches.

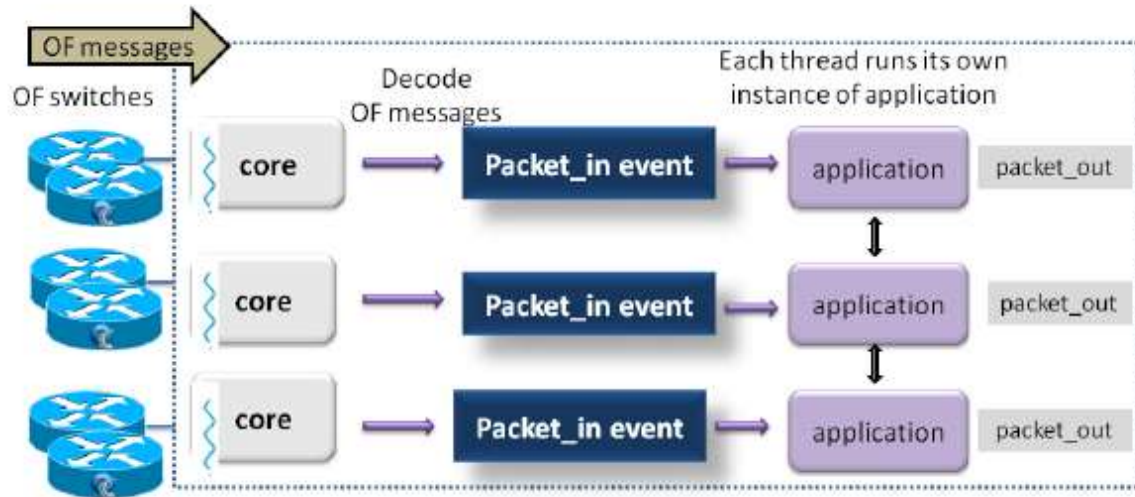


Figura 9.5: Arquitectura de un Controlador con Soporte Multihilo

La Tabla 9.2 tomada de [125] muestra el resultado de una evaluación de desempeño de un controlador propuesto con respecto al controlador Beacon. Se puede apreciar una escalabilidad casi lineal en flujos/segundo en ambos controladores a medida que se incrementa la cantidad de hilos soportados.

|                        | Threads |      |      |
|------------------------|---------|------|------|
|                        | 1       | 2    | 4    |
| Proposed OF controller | 1.36    | 2.62 | 4.50 |
| Beacon                 | 1.35    | 2.12 | 4.01 |

Tabla 9.2: Escalabilidad de Flujos (en Millones de Flujos/Segundo)

- **Código Abierto:** Esta funcionalidad determina la capacidad de ampliar y mejorar las funcionalidades implementadas en el controlador al disponer del código de desarrollo correspondiente.
- **Edad:** El tiempo de vida del controlador es un indicador de la vigencia, actualización y soporte de nuevas funcionalidades. A medida que aumenta la edad del controlador se amplían sus funcionalidades, se aumenta la pericia en su operación, depuración y mantenimiento aumentando la madurez y confianza en su selección y uso.
- **Documentación:** Considera la disponibilidad de información técnica de implementación y guías de usuario en el sitio web del fabricante o de la comunidad.

**Selección de Controladores basado en Aspectos de Eficacia:** Evalúa características de desempeño, confiabilidad, disponibilidad, escalabilidad y seguridad.

**Desempeño:** El desempeño de un controlador se mide en términos de la cantidad de solicitudes de flujos que puede manejar por segundo y de la rapidez para responder a cada solicitud [126]. NOX es un controlador popular que exhibe las siguientes métricas de desempeño: Puede manejar alrededor de 30 mil solicitudes de flujos por segundo mientras mantiene un tiempo de instalación de flujos de sub-10 milisegundos [127]. NOX-MT es un sucesor multihilo de NOX que utiliza técnicas de optimización como el procesamiento de entrada/salida en lotes para mejorar el desempeño línea base. Estas optimizaciones le permiten a NOX-MT superar a NOX por un factor de 33 y alcanzar métricas de 1.6 millones de solicitudes por segundo con un tiempo de respuesta promedio de 2 ms en una máquina de ocho núcleos con CPU de 2Ghz [126][128]. La capacidad de un controlador para un entorno dado depende en gran medida de la cantidad de servidores o usuarios que puede atender y de la cantidad de switches de la red. Un entorno de 1500 servidores recibe 100 mil flujos por segundo en promedio [129]. Una red con 100 switches puede en el peor caso resultar en 10 millones de flujos por segundo [130]. Adicionalmente al retardo de establecimiento de flujos de 10 milisegundos de un controlador SDN se incluye un retardo de 10% de retardo a un número grande de flujos de corta vida en una red. Estas condiciones sugieren que un solo controlador no puede satisfacer la demanda de red en términos de capacidad y tiempo de respuesta en entornos de redes grandes. Las métricas de desempeño tienen un impacto directo en el diseño de una solución SDN. Por ejemplo, si el número de flujos iniciados por los switches supera la capacidad soportada por el controlador se sugiere la incorporación de más controladores al despliegue [122].

Un controlador SDN puede establecer flujos en la red en dos modos de operación: modo proactivo y modo reactivo. El modo proactivo consiste en pre-poblar las tablas de flujos de los switches antes de la llegada del primer paquete a la red. En el modo reactivo las tablas de flujos se van alimentando en base a la llegada de nuevos paquetes. El enfoque proactivo minimiza la latencia de flujos, mientras que el enfoque reactivo permite al controlador tomar decisiones en una base flujo-por-flujo, tomando en consideración requerimientos de calidad de servicio y condiciones de carga. El tiempo de establecimiento de flujos en modo reactivo es la suma de los siguientes tiempos: (1) tiempo que le lleva al switch en enviar el paquete al controlador SDN, (2) tiempo en procesar el paquete en el controlador, (3) tiempo en enviar la respuesta del controlador al switch y (4) tiempo en poblar la entrada de flujo en el switch.

En la siguiente sección se presentan las herramientas de comparación de desempeño SDN/OpenFlow disponibles en el mercado.

**Herramientas para el Análisis de Desempeño de Controladores:** En la actualidad Cbench [59] es la herramienta de evaluación de desempeño de controladores OpenFlow con mayor uso y su variante WCBench<sup>114</sup> se utiliza para efectuar pruebas al controlador OpenDaylight. HCProbe<sup>115</sup> es otra herramienta flexible que evalúa varias características de un controlador incluyendo desempeño, confiabilidad y seguridad.

- **Cbench:** Emula un número configurable de switches OpenFlow que se comunican con un controlador. Cada switch envía un número configurable de mensajes de flujos nuevos (mensajes OpenFlow *packet-in*) al controlador OpenFlow, espera por el establecimiento de flujo (respuesta OpenFlow *flow-mod* o *packet-out*) y registra la diferencia de tiempo entre la solicitud y la respuesta. La herramienta permite seleccionar el tipo de prueba bajo dos modos de operación: modo throughput y modo latencia. En el modo throughput cada switch envía la mayor cantidad posible de paquetes para calcular el número de paquetes manejados por el controlador y se mide la máxima tasa de establecimiento de flujos que un controlador puede mantener. En el modo latencia cada switch envía una solicitud de flujo y espera la respuesta del controlador antes de enviar la próxima solicitud y se mide el tiempo de procesamiento del controlador bajo condiciones de carga baja.
- **WCBench:** Conjunto de scripts con Cbench en el núcleo que permiten medir el desempeño del controlador OpenDaylight. Es una mejora de Cbench para OpenDaylight que automatiza los pasos de instalación y configuración y añade funcionalidades para obtener resultados más confiables y soportar análisis estadísticos.
- **HCProbe** [131]: Es una herramienta desarrollada en el lenguaje Haskell que evalúa varios aspectos de un controlador OpenFlow. Incluye una implementación de referencia en software de un switch OpenFlow y un lenguaje específico de dominio para construir nuevos switches personalizados y escribir programas sobre estos. La herramienta genera correctamente paquetes OpenFlow y permite especificar patrones específicos de mensajes incluyendo mensajes OpenFlow malformados para efectuar pruebas de seguridad. Los usuarios pueden crear sus propios escenarios de pruebas mediante un API.

**Comparativa de Desempeño de Controladores:** Los factores claves que afectan el tiempo de establecimiento de flujos están asociados con el poder de procesamiento de los switches SDN, la capacidad de procesamiento del controlador y el desempeño entrada/salida del controlador. El desempeño del controlador depende de factores claves asociados a su arquitectura

---

<sup>114</sup> [https://wiki.opendaylight.org/view/CrossProject:Integration\\_Group:WCBench](https://wiki.opendaylight.org/view/CrossProject:Integration_Group:WCBench)

<sup>115</sup> <https://github.com/ARCCN/hcprobe>

interna [125] y al lenguaje de programación de su implementación [123]. Generalmente los controladores basados en el lenguaje C experimentan un mejor desempeño en modo throughput (ver Figura 9.6 tomada de [123]).

**Comparativa de Desempeño en Modo Throughput:** En la Figura 9.6 tomada de [123]) se muestra el resultado de unas pruebas de desempeño en modo throughput variando la cantidad de switches. Se puede observar que los controladores desarrollados en C arrojaron un mejor desempeño: Mul y Libfluid\_msg seguido por los controladores codificados en Java: Beacon, Iris y Maestro.

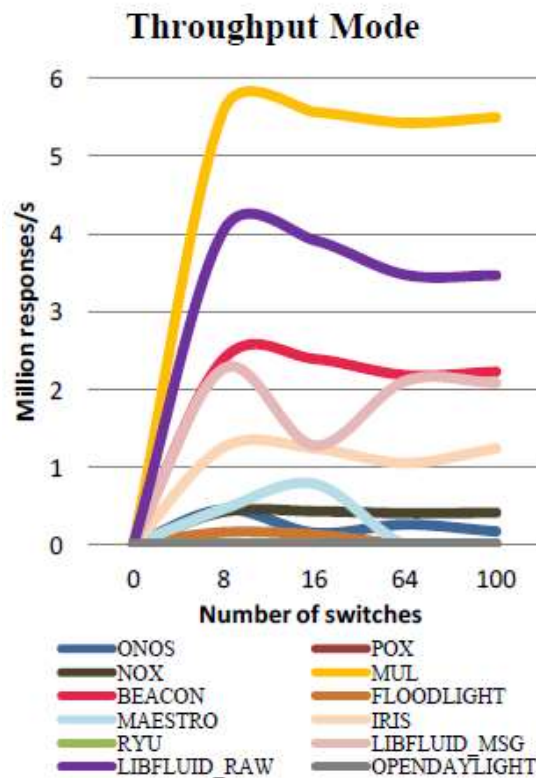


Figura 9.6: Pruebas Cbench - Variación Cantidad de Switches

En la Figura 9.7 tomada de [123] se muestra el resultado de pruebas de desempeño de controladores con la herramienta Cbench variando la cantidad de hilos. En esta se puede apreciar que los controladores basados en Java y C (MUL, Beacon, LiBfluid Raw, Iris) ofrecieron un mayor desempeño con respecto a los controladores codificados en otros lenguajes. POX no mostró una diferencia significativa ya que está codificado en el lenguaje Python donde el soporte multihilo no es muy eficiente.

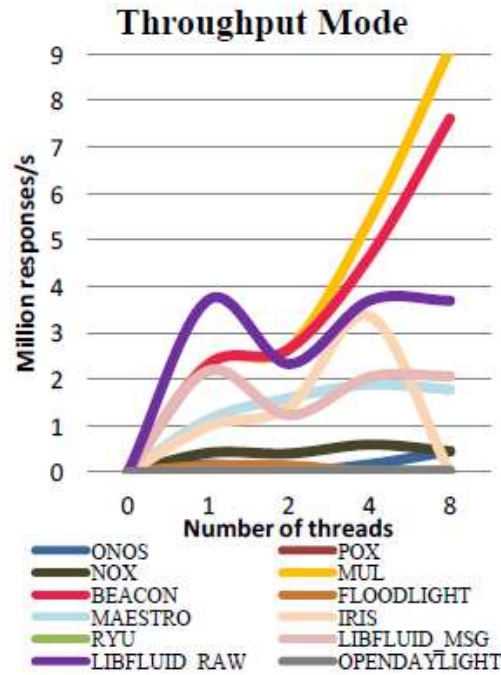


Figura 9.7: Pruebas Cbench - Variación Cantidad de Hilos

**Comparativa de Desempeño de Controladores en Modo Latencia:** En la Figura 9.8 tomada de [123] se muestra el resultado de pruebas de desempeño con la herramienta Cbench en modo latencia. En este caso el controlador Maestro construido bajo una arquitectura en modo de procesamiento por lotes adaptativo [125] presentó una menor latencia.

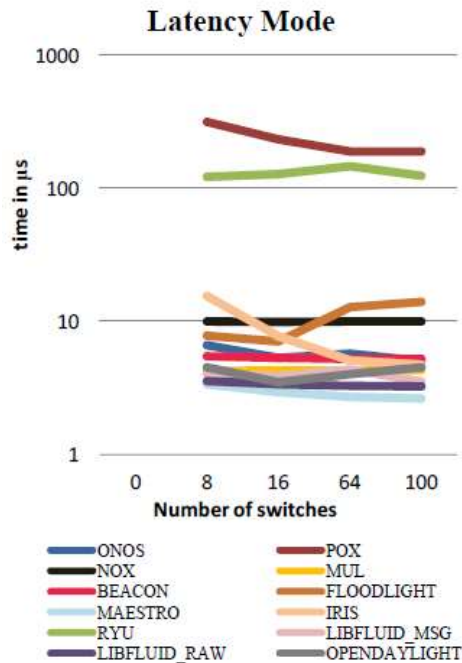


Figura 9.8: Pruebas Cbench - Variación Cantidad de Switches

- **Confiabilidad:** Se entiende por confiabilidad la habilidad de un controlador para operar por un largo periodo de tiempo soportando cargas de trabajo promedio sin cerrar accidentalmente conexiones con los switches o descartar mensajes OpenFlow.

En [131] se efectuaron pruebas de confiabilidad de controladores midiendo el número de fallas durante un largo período de tiempo bajo un perfil de carga dado. El perfil de tráfico se generó mediante la herramienta HCPProbe [131]. Los autores utilizaron la tasa de establecimiento de flujos registrada en el campus de Stanford en el año 2011 como perfil de la carga de trabajos. El perfil aplica para una carga de trabajos típica en un campus o red de oficinas durante un periodo de 24 horas con mayor cantidad de solicitudes en las horas del mediodía y una disminución en la cantidad de solicitudes en la noche. Para las pruebas se utilizaron cinco switches enviando mensajes *packet-in* con una tasa variando de 2000 a 18000 solicitudes por segundo. Se corrió la prueba durante 24 horas y se registró la cantidad de errores, entendiéndose por error el cierre de una sesión o una falla para recibir una respuesta de un controlador. Los experimentos demostraron que la mayoría de los controladores soportaron la carga de la prueba, a pesar que dos controladores: MuL y Maestro comenzaron a descartar mensajes *packet-in* después de varios minutos de operación. MuL descartó 660.271.177 mensajes y cerró 214 conexiones y Maestro descartó 463.012.511 mensajes sin cerrar conexiones. Para el controlador MuL las fallas fueron causadas por problemas con el módulo Learning Switch que no pudo añadir nuevas entradas a la tabla, lo cual condujo a una pérdida de paquetes y al cierre de conexiones con los switches.

- **Disponibilidad:** El controlador debe proveer mecanismos de alta disponibilidad basados en redundancia que le permitan delegar el control a otros controladores de manera automática en caso de falla en alguno de sus componentes o enlaces. Se recomienda disponer de un servicio de monitoreo del estado de los controladores y de un esquema de redundancia de controladores con recuperación automática. Se deben cumplir tres premisas para proveer mecanismos de alta disponibilidad:
  - **Redundancia de controladores.** La plataforma de controladores debe ser redundante para continuar el funcionamiento de la configuración de políticas en caso de que falle el controlador principal a cargo de la red.
  - **Datos de controladores en espejo.** La base de datos utilizada por el controlador principal para efectuar la toma de decisiones debe ser replicada y estar disponible para su uso por otros controladores de la red, de tal forma, que los controladores puedan configurar los dispositivos de la red de una manera consistente.
  - **Redundancia de caminos hacia los controladores.** Deben existir enlaces y caminos redundantes hacia los controladores para

asegurar que exista al menos un camino de comunicación disponible entre un switch y un controlador.

**Condición de un Solo Punto de Falla:** En una red SDN el controlador es el único responsable del funcionamiento de la red. En caso de ocurrirle una falla al controlador, la red puede seguir operando con las reglas predefinidas en los dispositivos SDN, pero no se pueden efectuar cambios en las tablas de flujos de los dispositivos. Esta situación se conoce como condición de un solo punto de falla. En una red legada cuando ocurre una falla en un nodo o en un enlace, los dispositivos pueden eventualmente recuperarse gracias a la inteligencia distribuida del plano de control. En la Figura 9.9 tomada de [47] se muestra un escenario donde falla un nodo en una red legada. En el primer diagrama de la izquierda se muestra la red operando bajo condiciones normales y dos flujos F1, y F2 siguen el mismo camino para salir al exterior. En el diagrama del medio ocurre una falla en el nodo central del camino que seguían los flujos F1 y F2 y se pierde momentáneamente la comunicación y el transporte del tráfico de los flujos F1 y F2. En el diagrama de la derecha los dos flujos F1 y F2 son re-enrutados por la inteligencia distribuida de la red hacia un camino alternativo para superar la falla del nodo central y siguen la misma ruta hacia su destino.

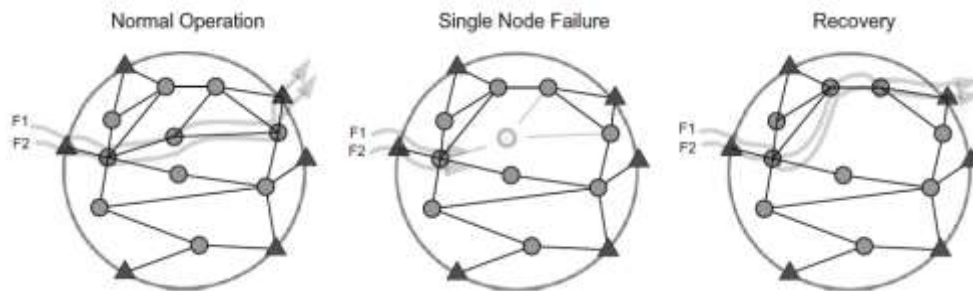


Figura 9.9: Esquema de Recuperación a Fallas en una Red Tradicional

En la Figura 9.10 tomada de [47] se muestra el caso de falla de un nodo de la red del ejemplo anterior en una red SDN. Bajo este escenario el controlador al percatarse de la falla en el nodo central de la red, selecciona las mejores rutas para los flujos F1 y F2 basado en el estado de la red y en las políticas QoS, como se indica en el diagrama de la derecha.

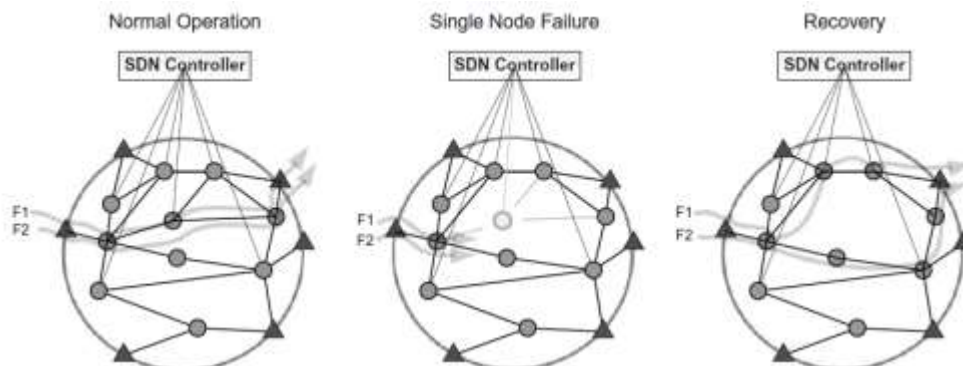


Figura 9.10: Esquema de Recuperación a Fallas en una Red SDN

Cuando la falla ocurre a nivel del controlador, la continuidad de la operación de la red se ve afectada y no se pueden actualizar las políticas ni el comportamiento de la red. En la Figura 9.11 tomada de [47] la pérdida del controlador deja a la red en un estado de funcionalidad reducida, que no puede adaptarse a la falla de otros componentes o a cambios operacionales en la red.

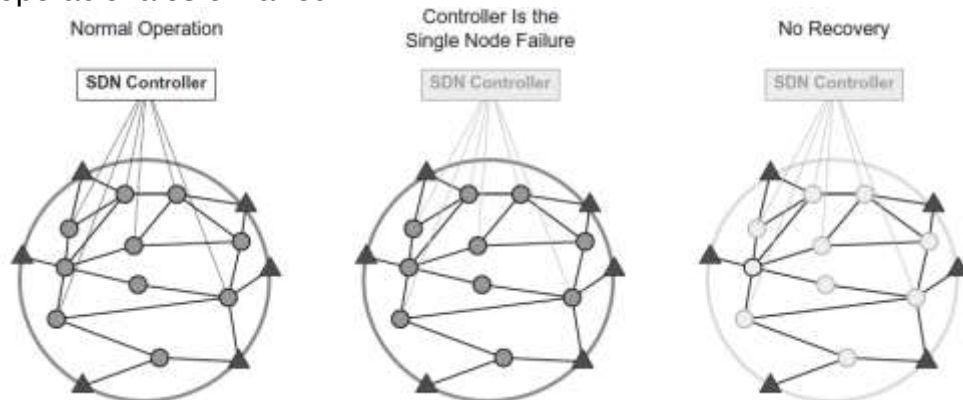


Figura 9.11: Condición Solo Punto de Falla de un Controlador

Para superar la condición de un solo punto de falla de un controlador, se pueden disponer varios controladores cooperando entre sí para mantener una visión consistente de la red.

**Controladores en Alta Disponibilidad:** Los controladores SDN deben utilizar técnicas de alta disponibilidad y/o redundancia para garantizar la continuidad del servicio en caso de falla de un componente de hardware, software o una sobrecarga en el servicio. En la Figura 9.12 tomada de [47] se muestra un arreglo de controladores en alta disponibilidad con redundancia a nivel del plano de control y de las bases de datos de control.

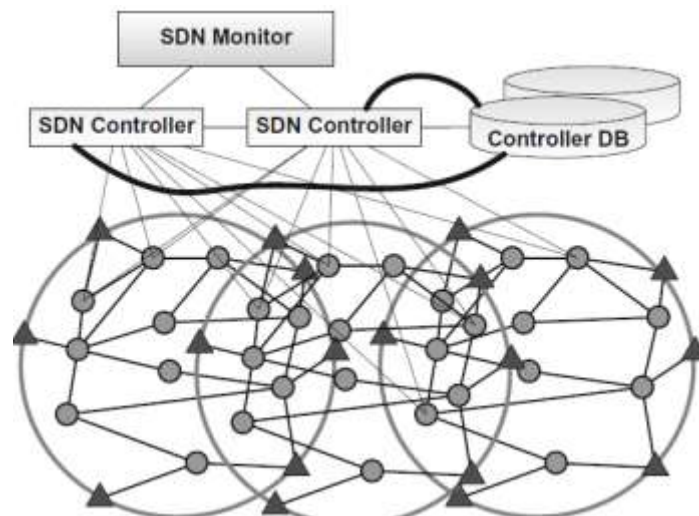


Figura 9.12: Esquema de Alta Disponibilidad de Controladores SDN

En este ejemplo la plataforma de controladores está conformada por dos controladores cooperando entre sí, una base de datos replicada en espejo y un monitor de red que se encuentra censando el estado de los controladores para tomar acciones correctivas en caso de fallas.

El esquema de redundancia de alta disponibilidad puede ser complementado con un esquema de redundancia a nivel de componentes físicos del controlador incluyendo arreglos de discos en espejo, memorias redundantes y varias fuentes de alimentación.

- **Escalabilidad:** El criterio de escalabilidad contempla la capacidad de crecimiento de la arquitectura de red para atender incrementos en el volumen de usuarios, aplicaciones y servicios. Los controladores seleccionados en la fase de diseño deben soportar crecimiento a nivel de recursos de cómputo para atender la demanda de servicios actual y un crecimiento esperado. La incorporación de más dispositivos de red impacta directamente la capacidad de cómputo del controlador y se puede generar una condición de congestión en la cual no se pueden atender nuevos requerimientos y ocasionar retardos que afecten el desempeño global de la red. El controlador debe ser capaz de escalar con el crecimiento de los usuarios, aplicaciones y dispositivos de la red. En caso de superarse la capacidad, se debe contar con un plan de acción para soportar las nuevas cargas de tráfico, incluyendo mecanismos de redundancia o segmentación de la red en dominios de control que compartan alcanzabilidad entre sus controladores. Un mecanismo para conseguir la escalabilidad de la red consiste en la implementación de una jerarquía de controladores con fronteras bien definidas.

**Jerarquía de Controladores:** Una forma de aumentar la escalabilidad de la red consiste en desplegar una jerarquía de controladores [132] que soporte la sobrecarga de eventos frecuentes que son enviados al controlador central de la red. Se puede establecer una jerarquía de árbol de dos niveles. Bajo este esquema las hojas del árbol conforman controladores que atienden requerimientos locales en base a la proximidad geográfica de los dispositivos de red y la raíz del árbol conforma el controlador central que mantiene el estado de la red y gestiona a las hojas de la jerarquía. En la Figura 9.13 tomada de [47] se muestra una red ejemplo basada en una jerarquía de controladores de dos niveles.

En la jerarquía, los controladores hoja atienden los eventos de mayor frecuencia y de carácter local, mientras que el controlador raíz atiende eventos inusuales o de menor frecuencia. Con este diseño un operador puede replicar controladores locales bajo demanda y aliviar la carga en el controlador maestro.

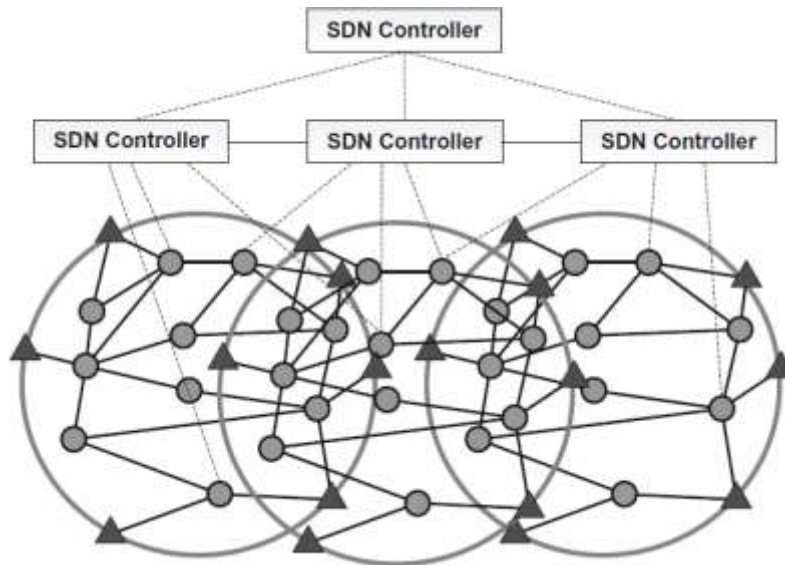


Figura 9.13: Jerarquía de Controladores

**El Problema de la Colocación del Controlador:** Trata el problema de diseño de definir cuantos controladores de red son necesarios en un despliegue y donde deben ubicarse para ofrecer tolerancia a fallas, disponibilidad y buen desempeño en una red SDN [133]. El retardo entre controladores y entre controladores y switches conducen a tiempos de respuesta grandes e influyen su habilidad para responder a eventos de red [126]. Heller et al. caracterizan formalmente el problema de la colocación de controladores para entornos WAN. Los autores reportan que la latencia desde cada nodo a un solo controlador pueden satisfacer las metas de tiempo de respuesta de las tecnologías existentes en muchas redes de tamaño medio. Sin embargo, el crecimiento de la red debe estar soportado por dos o más controladores cooperando entre sí para gestionar la red completa. Jimenez et al.[134] definen los principios para diseñar un plano de control escalable desde el punto de vista del problema de la colocación de controladores. Los autores utilizan un algoritmo llamado k-Critica para encontrar el número mínimo de controladores y su ubicación para construir una topología de red robusta con capacidad de gestión de fallas y balanceo de cargas entre varios controladores. Hu et al. [135] concluyen que el número de controladores y su ubicación pueden afectar la confiabilidad de una red SDN.

En la Figura 9.14 tomada de [126] se muestra un ejemplo que demuestra el efecto en la confiabilidad y desempeño al colocar uno o dos controladores en una red SDN de cinco switches. En la gráfica las líneas solidas representan enlaces físicos y las líneas punteadas representan el camino más corto entre switches o entre switches y controladores. La figura 9.14(a) muestra como la presencia de un solo controlador determina una condición de un solo punto de falla mientras que las figuras 9.14(b), 9.14(c) y 9.14(d) son más confiables.

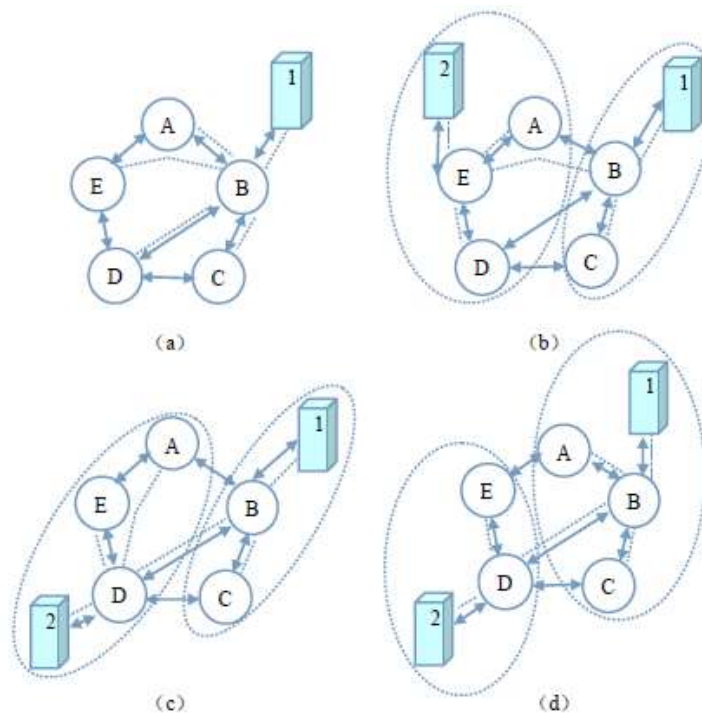


Figura 9.14: Ejemplo de Opciones de Colocación de Controladores SDN

En la figura 9.14(b), si se cae el enlace entre los switches A y B, se pierde la comunicación entre el switch A y su controlador y se rompe la comunicación entre los controladores. Si la misma falla ocurre en la figura 9.14(c) el camino de comunicación entre los controladores no se ve afectado. Las figuras 9.14(b) y 9.14(c) demuestran que la ubicación de los controladores afectan la confiabilidad de la red. La figura 9.14(d) representa un escenario de despliegue aún más confiable que el mostrado en las figuras anteriores. En este caso cuando ocurre una falla en el enlace entre los switches D y E solamente el switch E pierde la comunicación con su controlador (controlador 2). Por el contrario, en la figura 9.14(c) se nota como los switches E y A no pueden comunicarse con el controlador 2. En resumen el rango de control de cada controlador influencia la confiabilidad de la red [126].

- **Seguridad:** SDN enfrenta nuevos retos al introducir nuevos componentes a la red: APIs, aplicaciones y controladores. La complejidad para brindar seguridad en una red SDN crece, el control central se convierte en un objetivo de ataque y la apertura de interfaces dificulta la definición y aplicación de políticas de seguridad. El controlador es el componente fundamental de la arquitectura de red SDN y un ataque al controlador puede afectar el funcionamiento de la red completa [136]. Los ataques a SDN pueden variar desde la denegación de servicio que busca socavar la disponibilidad de las operaciones de red hasta los ataques de hombre-en-el-medio que buscan modificar las reglas enviadas a los dispositivos del plano de datos para tomar control de los caminos de la red.

Otra variante de ataque consiste en comprometer el controlador explotando vulnerabilidades e instalando aplicaciones maliciosas para tomar el control total de la infraestructura de red. La mitigación de riesgos en SDN requiere un enfoque de seguridad por diseño que provea protección adecuada a la infraestructura de ataques de red y otros vectores involuntarios como bugs de software o errores de configuración de dispositivos [136]. Hayward et al. [139] presentan un estudio con los avances de la industria y de la investigación académica en el área de la seguridad de SDN. El estudio destaca siete categorías de problemas de seguridad: acceso no autorizado, fuga de datos, modificación de datos, aplicaciones maliciosas y compromiso de aplicaciones, denegación de servicio y problemas de configuración y seguridad SDN a nivel de sistemas. Los autores proponen soluciones a estos problemas. Algunas soluciones contemplan la replicación de controladores y aplicaciones para proveer alternativas de gestión o control en el caso de fallas de hardware o software, la diversidad de controladores para robustecer el plano de control en caso de bugs de software y vulnerabilidades de un solo controlador y componentes de seguridad para proteger la confidencialidad de datos sensibles. Hori et al. [137] consolida los vectores de ataque indicados por Kreutz et al. [138] con la lista de ataques de SDNSecurity.org<sup>116</sup> y construye una hoja de verificación ampliada para evaluar la seguridad de redes SDN/OpenFlow junto con sus contramedidas de protección.

**Vectores de Ataques SDN:** SDN tiene dos propiedades que pueden convertirla en un centro de ataque: (1) Una arquitectura de red basada en software que puede contener bugs de aplicación y otras vulnerabilidades y un punto de control central que puede convertirse en un objetivo de ataque para tomar posesión y control de toda la red. Kreutz et al. [138] clasifica los problemas de seguridad de SDN en siete vectores de ataque (ver Figura 9.15 tomada de [138]):

**Vector de Amenaza 1, Tráfico forjado o falso:** Ataque dirigido a switches o controladores por falla en dispositivos o por usuarios maliciosos. El atacante puede utilizar un dispositivo de red como un switch, una estación de trabajo o un servidor y lanzar un ataque de denegación de servicio a switches OpenFlow saturando los recursos de memoria y los recursos del controlador.

**Solución:** Utilizar sistemas de detección de intrusión con soporte para análisis causa raíz que ayuden a identificar anomalías de flujos. Esta solución puede estar acompañada de mecanismos que limiten la tasa de solicitudes al plano de control para restringir la dimensión del ataque.

**Vector de Amenaza 2, Ataques de vulnerabilidades en los switches:** Consiste en la utilización de algún switch para descartar o ralentizar paquetes.

---

<sup>116</sup> [http://sdnsecurity.org/project\\_SDN-Security-Vulnerability-attack-list.html](http://sdnsecurity.org/project_SDN-Security-Vulnerability-attack-list.html)

**Solución:** Implementar mecanismos de pruebas de software, incluyendo mecanismos de gestión de confianza autónoma para componentes de software [141]. También se pueden utilizar mecanismos para monitorear y detectar comportamientos con fallas en la red.

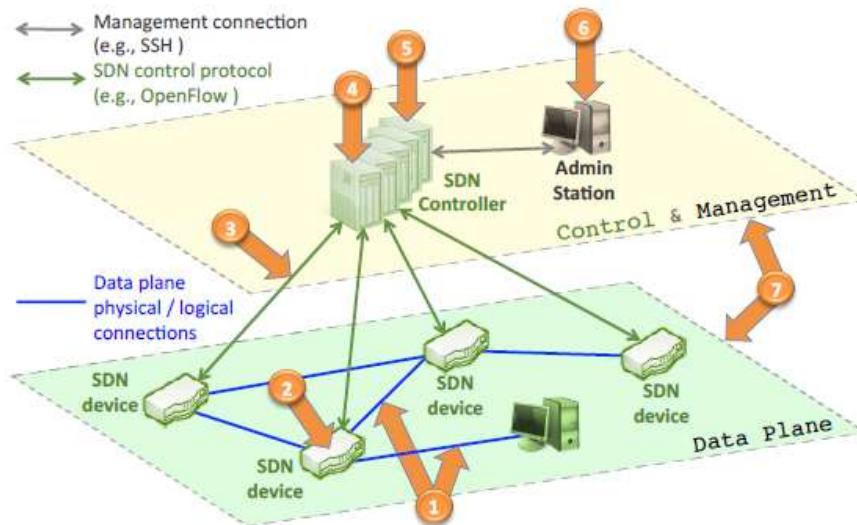


Figura 9.15: Vectores de Seguridad SDN

**Vector de Amenaza 3, Ataques en las comunicaciones del plano de control:** El plano de control puede ser utilizado por intrusos para generar ataques de denegación de servicio o para robo de datos. La arquitectura X.509 TLS/SSL utilizado en el canal de control ha sido víctima de ataques hombre-en-el-medio en la jerarquía de Autoridades Certificadoras [142]. La seguridad del canal de comunicación es tan fuerte como su enlace más débil, el cual puede ser un certificado auto-firmado, una Autoridad de Certificados comprometida o aplicaciones y librerías vulnerables. El modelo TLS/SSL no es suficiente para establecer y asegurar confianza entre controladores y switches. Una vez que el intruso gana acceso al plano de control, puede ampliar el nivel de ataque en el número de switches bajo su dominio para lanzar ataques de denegación de servicio distribuido.

**Solución:** Utilizar modelos de confianza oligárquicos con múltiples autoridades de certificación con anclas de confianza. Una posibilidad consiste en desplegar una autoridad por sub-dominio o por instancia de controlador. Otra opción es asegurar la comunicación con criptografía de umbrales a través de réplicas de controladores donde un switch necesita al menos  $n$  acciones para obtener un mensaje de control válido [143]. La criptografía de umbrales tiene como objetivo ofrecer tolerancia a fallas ya que es necesario comprometer la seguridad de varias entidades para romper la seguridad del sistema y la distribución de responsabilidades en la cual varios servidores deben cooperar para realizar una acción y poder descifrar un mensaje. Adicionalmente se pueden establecer mecanismos de

asociación de dispositivos para garantizar la confianza entre el plano de control y los dispositivos del plano de datos.

**Vector de Amenaza 4, Ataques a Vulnerabilidades del Controlador:** Esta es la amenaza más severa a una red SDN. Una falla en un controlador o un controlador malicioso compromete la operación de la red completa.

**Solución:** Se pueden emplear varias técnicas incluyendo mecanismos de replicación de controladores, diversidad de controladores, aplicaciones, lenguajes de programación, imágenes de software y otros, técnicas de recuperación que hacen un refrescamiento del sistema a un estado limpio y confiable, asegurar los elementos sensibles del controlador como llaves criptográficas, implementar políticas restringiendo cuales interfaces o aplicaciones pueden manipular reglas o restringir el alcance en el tipo de reglas que pueden generar para programar la red.

**Vector de Amenaza 5, Carencia de mecanismos para asegurar la confianza entre el controlador y las aplicaciones de gestión:** Los controladores y las aplicaciones carecen de mecanismos para establecer relaciones de confianza. Las técnicas para certificar dispositivos es diferente para certificar aplicaciones.

**Solución:** Proveer mecanismos de gestión autónoma para garantizar que la aplicación es confiable durante su ciclo de vida.

**Vector de Amenaza 6: Vulnerabilidades y ataques a las estaciones administrativas:** Tiene como objetivo atacar las consolas de administración de la red para reprogramar el comportamiento de la red desde un solo lugar.

**Solución:** Utilizar protocolos con verificación de credenciales dobles, requiriendo las credenciales de dos usuarios para acceder a un servidor de gestión. Adicionalmente se requieren mecanismos de recuperación que le permitan al servidor de gestión regresar a un estado confiable después de una reinicialización del sistema.

**Vector de Amenaza 7: Carencias de recursos confiables para actividades forenses y remediación:** Para investigar y determinar los hechos sobre algún incidente se requiere información confiables de todos los elementos y recursos de la red. Estos datos son útiles siempre y cuando sean confiables y hayan sido previamente autenticados y mantenido su integridad. La remediación requiere de instantáneas del sistema seguras y confiables para garantizar una recuperación rápida y correcta de los elementos de red a un estado conocido.

**Solución:** Implementar mecanismos de registro y rastreo en los planos de datos y control. Además los registros deben ser indelebles y deben tener respaldo en un repositorio de almacenamiento externo.

**Plataforma de Control Confiable y Segura:** Kreutz et al. [138] proponen una plataforma de control SDN basada en un conjunto de principios y mecanismos de seguridad:

**Replicación:** Consiste en la replicación de instancias del controlador y en la replicación de sus aplicaciones externas para aumentar la confiabilidad de la arquitectura de red (ver Figura 9.16 tomada de [138]).

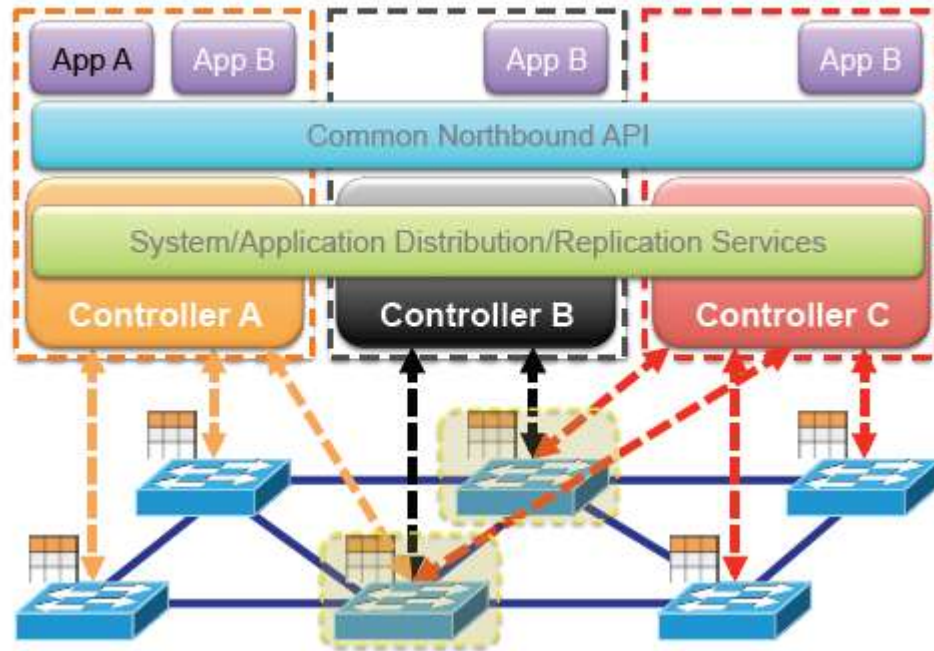


Figura 9.16: SDN Confiable y Segura

El controlador se ha replicado en tres instancias: Controlador A, Controlador B y Controlador C y la aplicación B también se ha replicado en tres instancias. Con esta técnica se asegura la tolerancia de hardware y software y se pueden aislar controladores o aplicaciones que presenten fallas en un momento determinado. Adicionalmente, la aplicación B con los algoritmos de consistencia adecuados puede continuar programando los switches de la red a diferencia de la aplicación A que pierde la características de continuidad en caso de fallar.

**Diversidad:** Plantea disponer diferentes plataformas de implementación de controladores con diferentes sistemas operativos, imágenes de software y aplicaciones de tal forma que si un controlador se ve afectado por algún bug o falla conocida el resto de los controladores no se ve afectado y continúan operando.

**Mecanismos de Auto-salud:** Contempla la disposición de mecanismos de recuperación proactivos y reactivos que permitan retornar el sistema a un estado seguro de manera automática reemplazando los componentes afectados en caso de fallas. Se recomienda hacer el reemplazo de

componentes con versiones diferentes para robustecer la defensa de la plataforma en contra de vulnerabilidades de sistemas específicos.

**Asociación dinámica de dispositivos:** Contempla la propiedad de resiliencia de los switches de la red al poder asociarse a diferentes controladores de manera dinámica en caso de una condición de falla en su controlador original. Adicionalmente a la mejora en la confiabilidad de la red se puede incrementar el rendimiento general de la red si los switches pueden estar asociados a diferentes controladores de manera simultánea balanceando la carga y reduciendo el retardo de control seleccionando el controlador que ofrezca respuestas más rápidas.

**Confianza entre controladores y dispositivos:** Consiste en establecer una relación de confianza entre un controlador y un conjunto de dispositivos confiables. Un mecanismo puede ser disponer de una lista blanca de dispositivos confiables en el controlador que restrinja la asociación a la red solo a los dispositivos incluidos en la lista. Otro mecanismo pudiera ser monitorear el comportamiento de los dispositivos de la red y calificar su confianza. En caso de presentar alguna desviación en el comportamiento normal de la red los dispositivos pudieran ser aislados y colocados en cuarentena por el resto de los dispositivos y controladores de la red.

**Confianza entre controladores y aplicaciones:** Se pueden implementar modelos de gestión de confianza autónomos en sistemas de software basado en componentes [141]. En estos modelos se provee una noción holística para que una entidad de confianza confíe en una entidad confiable mediante la observación de su comportamiento y la medición de atributos de disponibilidad, integridad, seguridad, mantenibilidad y confidencialidad.

**Dominios de seguridad:** Los dominios de seguridad en las plataformas de control SDN pueden aplicarse de manera equivalente a los esquemas de sandboxing y de virtualización implementados en los sistemas operativos restringiendo un conjunto mínimo de operaciones y comunicaciones entre diferentes dominios.

**Componentes seguros:** Representa un bloque de construcción fundamental en la plataforma de control SDN confiable y consta de dispositivos de almacenamientos a pruebas de manipulación confiables que respaldan información de seguridad como llaves privadas de cifrado o información de autenticación de usuarios y dispositivos y permiten la ejecución de operaciones confiable sobre estos. De esta manera si el sistema es comprometido la información de confidencialidad se encuentra respaldada.

**Actualización y parches de software rápido y confiable:** Los parches y actualizaciones son esenciales para reducir la ventana de vulnerabilidades.

Se debe disponer de una plataforma de control que permita hacer las actualizaciones de una manera segura [144].

Para complementar los mecanismos de seguridad descritos en esta sección se pueden seguir los lineamientos y principios de seguridad dictados por la ONF asociados al protocolo OpenFlow [140] cuya meta es mitigar el potencial de explotación del protocolo y evaluar y controlar los efectos negativos como la sobrecarga y las fallas de seguridad que pudieran ser introducidas en los despliegues de mecanismos de seguridad.

**Selección de Controladores basado en Casos de Uso:** Evalúa aspectos desde la perspectiva de su aplicación a casos de uso particulares<sup>117</sup> [121]. En la Tabla 9.3 tomada de [121] se especifican los casos de uso más comunes que son tomados en consideración para la selección de un controlador particular.

| Caso de Uso                                                                      | Descripción                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Interoperabilidad de Red Legada                                                  | Soporta arquitecturas y elementos de red legada para ofrecer automatización y servicios de red extremo a extremo                                                                                                                                                                                                       |
| Overlay de Borde Distribuido                                                     | Uso de túneles (generalmente capa 2 en capa 3) para crear una red overlay, para desacoplar un servicio de red de la infraestructura subyacente y para permitir a las redes de núcleo escalar y evolucionar independientemente de los servicios ofrecidos                                                               |
| Virtualización de Red Salto a Salto                                              | Ejemplo VLANs. Un controlador SDN "Reactivo" crea un camino de flujo desde un borde de la red a otro borde a través de OpenFlow mediante la instalación de flujos en cada switch del camino, incluyendo los switches de core y agregación                                                                              |
| Soporte OpenStack Neutron                                                        | Provee las abstracciones de red esperadas por OpenStack mediante conectores Neutron en el controlador para ofrecer funcionalidades de red como servicio                                                                                                                                                                |
| Inserción de Servicios Capa 4- Capa 7 y Encadenamiento de Funciones de Servicios | La inserción de servicios consiste en la incorporación dinámica de servicios en la infraestructura de red incluyendo firewalls, balanceadores de carga, sistemas de prevención de intrusos, caching y otras inspecciones de paquetes en profundidad en un base por clientes para cualquier flujo individual o agregado |
| Monitoreo de Red                                                                 | Permite recopilar datos de auditorías de los dispositivos de red en cualquier granularidad incluyendo estadísticas por IP, dirección MAC, aplicación u otros                                                                                                                                                           |
| Aplicación de Políticas                                                          | Hace cumplir una política consistente en un solo dominio o en múltiples dominios de red                                                                                                                                                                                                                                |

<sup>117</sup> <https://thenewstack.io/sdn-series-part-eight-comparison-of-open-source-sdn-controllers>

|                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Balanceo de Carga                                                                                               | Caracteriza tipos de tráfico y los reenvía a una variedad de recursos de red. También contempla el monitoreo de tráfico de la red y la programación de flujos basado en la demanda de servicios y la capacidad de los servidores                                                                                                                                                                                                                                |
| Ingeniería de Tráfico                                                                                           | Servicio de optimización de tráfico de la red mediante el análisis dinámico de tráfico de datos, la predicción y la regulación del comportamiento de los datos transmitidos en la red                                                                                                                                                                                                                                                                           |
| Taps dinámicos de red                                                                                           | Un tap de red es un dispositivo insertado en la red capaz de acceder los datos que fluyen a través de una red IP y proveen capacidades de visibilidad y resolución de problemas en cualquier puerto en un despliegue de switches                                                                                                                                                                                                                                |
| Optimización de Red Multicapa                                                                                   | Extiende SDN para soportar la interconexión de recursos IT incluyendo almacenamiento, máquinas virtuales y cómputo virtual, utilizando tecnologías de switching y redes de transporte óptico emergentes y redes de paquetes                                                                                                                                                                                                                                     |
| Redes de Transporte: NV (Network Virtualization), Re-enrutamiento de Tráfico, Interconexión de Centros de Datos | Servicio para reenviar grandes cantidades de datos confiables y evitar dispositivos de seguridad costosos. Crea interconexiones dinámicas en los intercambios de Internet entre enlaces empresariales o entre proveedores de servicios a través de switches de alto desempeño efectivos en costos. Ancho de banda dinámico: permite el control programático en los enlaces de las operadoras de servicio para solicitar ancho de banda extra cuando se necesite |
| Casos de Uso para Redes de Campus                                                                               | Incluye servicios de aislamiento de tráfico, movilidad transparente, aplicación de políticas de seguridad, gestión de ancho de banda y conciencia de aplicaciones                                                                                                                                                                                                                                                                                               |

**Tabla 9.3: Casos de Uso SDN**

En la Tabla 9.4 tomada de [121] se indica la aplicación de casos de uso de seis controladores SDN populares: Trema, Nox/Pox, RYU, Floodlight, ODL y ONOS.

| <b>Caso de Uso\Controlador</b>                                                  | <b>Trema</b> | <b>Nox/Pox</b> | <b>RYU</b> | <b>Floodlight</b> | <b>ODL</b> | <b>ONOS</b> |
|---------------------------------------------------------------------------------|--------------|----------------|------------|-------------------|------------|-------------|
| Interoperabilidad Red Legada                                                    | SI           | SI             | SI         | PARCIAL           | SI         | NO          |
| Overlay de Borde Distribuido                                                    | NO           | NO             | NO         | SI                | SI         | SI          |
| Virtualización de Red Salto a Salto                                             | NO           | NO             | SI         | SI                | SI         | NO          |
| Soporte OpenStack Neutron                                                       | NO           | NO             | NO         | NO                | SI         | PARCIAL     |
| Inserción de Servicios Capa 4-Capa 7 y Encadenamiento de Funciones de Servicios | NO           | NO             | PARCIAL    | NO                | SI         | PARCIAL     |
| Monitoreo de Red                                                                | PARCIAL      | PARCIAL        | SI         | SI                | SI         | SI          |

|                                                                                                                 |         |         |         |         |         |         |
|-----------------------------------------------------------------------------------------------------------------|---------|---------|---------|---------|---------|---------|
| Aplicación de Políticas                                                                                         | NO      | NO      | NO      | PARCIAL | SI      | PARCIAL |
| Balanceo de Carga                                                                                               | NO      | NO      | NO      | NO      | SI      | NO      |
| Ingeniería de Tráfico                                                                                           | PARCIAL | PARCIAL | PARCIAL | PARCIAL | SI      | PARCIAL |
| Taps dinámicos de red                                                                                           | NO      | NO      | SI      | SI      | SI      | NO      |
| Optimización de Red Multicapa                                                                                   | NO      | NO      | NO      | NO      | PARCIAL | PARCIAL |
| Redes de Transporte: NV (Network Virtualization), Re-enrutamiento de Tráfico, Interconexión de Centros de Datos | NO      | NO      | PARCIAL | NO      | PARCIAL | PARCIAL |
| Casos de Uso para Redes de Campus                                                                               | PARCIAL | PARCIAL | PAR     | PARCIAL | PARCIAL | NO      |
| Soporte de Enrutamiento                                                                                         | SI      | NO      | SI      | SI      | SI      | SI      |

**Tabla 9.4: Comparación de Controladores SDN Basado en Casos de Uso**

Existen 3 tipos de respuestas formuladas de acuerdo al siguiente esquema:

- SI:
  - Soporta completamente.
  - La mayoría de los elementos existen y puede soportar con cambios menores.
  - Existe una solución comercial/propietaria desarrollada en el tope del controlador de código abierto.
  - Soporta un número significativo de casos de usos dentro de un dominio aplicación de red.
- NO:
  - No soporta.
  - Solo existen elementos funcionales y necesitan cambios mayores.
  - No existe una solución comercial/propietaria en el tope del controlador.
- PARCIAL:
  - Soporta el caso de uso parcialmente.
  - Existe un número significativo de aplicaciones y servicios en el controlador que pueden ayudar en el caso de uso.
  - Soporta parcialmente un conjunto de casos de usos dentro de un dominio de aplicación de red.
  - El trabajo está en progreso.

Se puede apreciar que OpenDaylight soporta la mayoría de los casos de uso, razón por la cual ha alcanzado gran popularidad para el uso académico y comercial. Ryu y Floodlight soportan un conjunto de casos de usos similar, soportando Ryu dos casos de uso importantes que incluyen inserción y

encadenamiento de servicios y redes de transporte. También Trema, Nox y Pox soportan casos de usos similares.

### 9.1.5 Consideración 5: Integración e Interoperabilidad

La red SDN debe ser capaz de integrarse a la red actual y al resto de los elementos de la arquitectura SDN. Se deben tratar aspectos de integración e interoperabilidad entre las capas de control e infraestructura de la arquitectura, considerando la compatibilidad de versiones y funcionalidades soportadas en los controladores y switches de la red. También se deben analizar aspectos de integración y funcionalidades de las APIs soportadas por el controlador y las aplicaciones de red. La solución debe estar basada en estándares abiertos que permita la integración con un ecosistema SDN interoperable tanto a nivel de aplicaciones como a nivel de infraestructura para contar con una arquitectura sostenible en el largo plazo. La capacidad de integración con las redes legadas puede darse a través de la ejecución de pilas de red duales OpenFlow y tradicional que soporten protocolos de red como BGP y OSPF. El nivel de integración de un controlador con las aplicaciones, sistemas de gestión e infraestructuras de red externas se encuentra definido por las interfaces Northbound y Southbound soportadas.

- **Interfaces Northbounds:** Facilitan la integración con la capa de aplicación. En la actualidad el protocolo REST representa el API Northbound más utilizado y se encuentra implementado en la mayoría de los controladores.
- **Interfaces Southbounds:** Permiten la implementación de las reglas de reenvío, la gestión de la infraestructura de red y la comunicación con infraestructuras legadas a través de protocolos de red estándar. OpenFlow define el comportamiento de la red, NETCONF [31], RESTCONF [38] y OF-Config [40] gestionan la red y los protocolos IS-IS, OSPF y BGP se utilizan para la comunicación con redes tradicionales.

### 9.1.6 Consideración 6: Gestión y Monitoreo

Para garantizar el éxito de la implementación de la solución se debe contar con una infraestructura de monitoreo y gestión robusta capaz de dar seguimiento y control a los aspectos de conectividad, desempeño, estabilidad y correctitud del funcionamiento SDN/OpenFlow.

#### Requerimientos de Monitoreo de Red SDN

Una implementación SDN exitosa se encuentra respaldada por la visibilidad completa del comportamiento de la red [146]. La Figura 9.17 tomada de [145] muestra una arquitectura de recolección de métricas típica en una red SDN donde se recopila información de los planos de control y datos incluyendo los controladores y los dispositivos de red. Las métricas se almacenan, analizan y

reportan en un nodo de gestión central. Las métricas deben analizarse por un periodo de tiempo suficiente que permita identificar tendencias de comportamiento así como la identificación de umbrales máximos y mínimos de utilización. Las herramientas pueden correlacionar varias métricas y generar reportes más complejos.

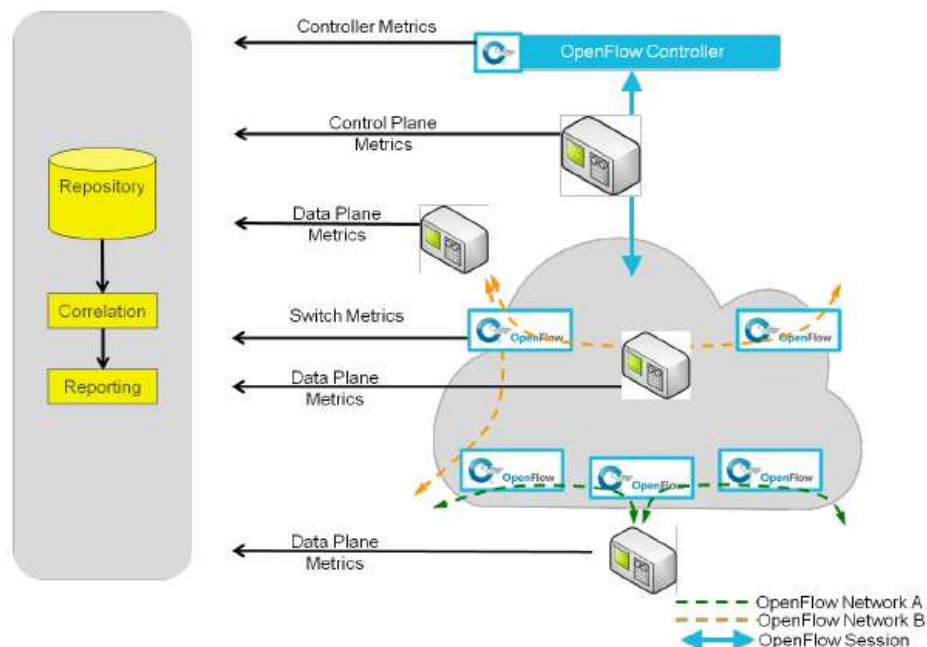


Figura 9.17: Infraestructura de Monitoreo SDN - Recolección de Métricas

**Infraestructura de Monitoreo SDN:** Contempla un conjunto de herramientas de captura, almacenamiento y análisis de tráfico. La infraestructura de monitoreo recopila información en 2 planos:

**Plano de Control:** Se recopila información a nivel de flujos basado en mensajes de control entrantes *packet-in* y *flow-exp*. Otras estadísticas importantes a recopilar son la tasa de arribo de flujos *flow\_arrival\_rate* y los flujos activos *active\_flows*.

**Plano de Datos:** Se pueden implementar nodos de monitoreo dedicados a la recolección de estadísticas ejecutando comandos ping y wget entre sí que permitan recopilar información sobre la utilización de CPU del switch, el tiempo de establecimiento de flujos, retardos RTT, retardos wget y tasa de pérdidas de paquetes.

#### Herramientas de Monitoreo SDN/OpenFlow Disponibles:

Entre las herramientas de monitoreo convencionales se encuentran ping [69], tcpdump y wget. Adicionalmente se encuentra herramientas más completas con interfaces gráficas y soporte de reportes como NetFlow [82], sFlow [147], JFlow [83] o herramientas de monitoreo especializadas que verifiquen el comportamiento de la red y la correctitud del funcionamiento OpenFlow. Algunas herramientas especiales de monitoreo disponibles son:

**PayLess** [84]: Framework de monitoreo basado en consultas para redes SDN que provee un API RESTful flexible para recopilar estadísticas a diferentes niveles de agregación, como flujos, paquetes y puertos. PayLess ejecuta la recopilación de información con alta precisión en tiempo real sin incurrir en una alta sobrecarga de red.

**OpenTM** [85]: Arquitectura de monitoreo que lleva la traza de los flujos activos en una red OpenFlow. Adicionalmente obtiene información de la aplicación de enrutamiento del controlador y sondea periódicamente contadores de cantidad de bytes y número de paquetes de los flujos activos en los switches a lo largo del camino de datos. Para reducir la sobrecarga en la red se pueden sondear de manera aleatoria un subconjunto de los switches seleccionados cuidadosamente para no afectar la precisión de las estadísticas recopiladas por la herramienta.

**FlowSense** [86]: Arquitectura de monitoreo que permite estimar el desempeño de una red OpenFlow a un bajo costo. Utiliza un método pasivo que captura y analiza el intercambio de los mensajes de control entre los switches y el controlador de una red OpenFlow asociados a cambios en el tráfico de la red, como ocurre con los mensajes *packet-in* y *flow-removed*. Los mensajes *packet-in* notifican la llegada de un nuevo flujo y los mensajes *flow-removed* reportan la expiración de un flujo, permitiendo estimar la utilización de un enlace por un flujo determinado.

**Retos del Monitoreo de Red SDN:** El monitoreo de red en SDN presenta los siguientes retos [146]:

- Se requiere el monitoreo de la red legada en términos de la utilización a nivel de enlaces, alcanzabilidad y disponibilidad de dispositivos y las métricas de salud de los dispositivos de red incluyendo el estado de la CPU, memoria, tarjetas y módulos de red.
- Localización de fallas de desempeño: Se debe determinar si la falla de una aplicación obedece a su funcionamiento o depende de aspectos relacionados a la red.
- Visibilidad de flujo de aplicaciones: Se requiere tener visibilidad de los caminos de red que toma una aplicación y el estado de los dispositivos de red y enlaces por donde viaja la aplicación. Se deben monitorear los caminos para detectar cambios en los saltos de red, métricas, retardos y ancho de banda.
- Monitoreo proactivo: Se deben especificar las aplicaciones impactadas en caso de ocurrir congestiones de tráfico en la red.
- Se debe reportar y llevar control de las notificaciones, verificaciones y detección de cambios en la configuración de la red.
- Se deben proveer mecanismos de control de acceso basado en roles en la red RBAC (Role Based Access Control) donde se delimite el alcance de la configuración y monitoreo que puede efectuar un administrador de red en base a su rol administrativo.
- La infraestructura de monitoreo debe soportar protocolos de red relevantes incluyendo enrutamiento VXLAN.

### **9.1.7 Consideración 7: Soporte y Adiestramiento**

Un criterio importante para garantizar la operación de la red de un despliegue SDN es el soporte del fabricante del controlador y de los equipos de red: switches, puntos de acceso inalámbricos y routers. Se debe evaluar el tiempo en el mercado, los niveles de SLA (Service Level Agreement) para asistencia técnica y en sitio ofrecidos: 8x5xNBD (8x5xNext Business Day), 24x7x4, 24x7x2, el soporte para el reemplazo de partes y piezas en caso de fallas la disponibilidad de portal web con documentación en línea y el fácil acceso a la información técnica de guías de diseño, implementación y resolución de problemas. El adiestramiento también constituye un aspecto importante ya que los administradores de red deben operar de manera autónoma la red y en la medida que adopten mayor dominio de la tecnología podrán brindar una mejor asistencia técnica a los usuarios de la red y podrán mejorar los tiempos de respuesta en la resolución de problemas y en la ampliación de la red. El adiestramiento debe considerar los siguientes aspectos: diseño, funcionamiento, operación, mantenimiento y gestión de redes SDN/OpenFlow, Configuración, actualización y validación de reglas en el controlador SDN, integración con aplicaciones externas, implementación de mecanismos de redundancia de controladores, (4) programación de controladores y (5) gestión de la red, monitoreo de métricas de desempeño y seguridad de la red.

## **10. Conclusiones y Trabajos Futuros**

Las implementaciones de SDN a través de OpenFlow proveen un enfoque poderoso para gestionar redes complejas con demandas dinámicas de datos. En una red SDN, se desacoplan los planos de control y datos, se centraliza la inteligencia y el estado de la red y se abstraen la infraestructura de red subyacente de las aplicaciones. Como resultado, las empresas obtienen agilidad, automatización y control de la red, que les permite construir redes flexibles y escalables capaces de adaptarse a las necesidades cambiantes de los servicios de los usuarios.

El despliegue de una nueva red SDN o la migración a un ambiente SDN requiere tomar en consideración aspectos importantes de diseño que permitan garantizar la operación de la red y proveen un ambiente escalable que pueda satisfacer el crecimiento en servicios y aplicaciones del campus empresarial. Para ello se deben tomar en cuenta las consideraciones descritas en este documento, en términos de factibilidad técnica/comercial, despliegue incremental, integración con los servicios y aplicaciones, y principios de resiliencia, escalabilidad, confiabilidad, gestión y seguridad.

Entre los trabajos futuros que complementen este estudio se encuentra la implementación de SDN en el campus universitario de la Universidad Central de Venezuela. Se puede considerar un escenario con un controlador central en el edificio del Rectorado que gobierne el plano de control de todas las facultades de la universidad o una jerarquía de controladores donde cada facultad disponga de su propio controlador que rijan su propio comportamiento y reporte su estado al controlador central. Otro Trabajo de interés es la implementación de Internet de las Cosas bajo una arquitectura SDN en un campus empresarial. Es de interés efectuar un estudio que amplíe las consideraciones descritas en este documento con las particularidades para estos ambientes.



## Referencias Bibliográficas

- [1] Open Networking Foundation. Software Defined Networking: The New Norm for Networks. ONF White Paper. Abril, 2012.
- [2] N. McKeown. OpenFlow: Enabling Innovation in Campus Networks. Stanford University. Marzo, 2008.
- [3] J. Case, M. Fedor, et al. A Simple Network Management Protocol (SNMP). RFC 1098. IETF. Mayo, 1990.
- [4] D. Tennenhouse, D. Wetherall. Towards an Active Network Architecture. ACM SIGCOMM Computer Communications Review, 26(2):5-18. Abril, 1996
- [5] B. Schwartz, A. Jackson, et al. Smart packets: Applying Active Networks to Network Management. Journal ACM Transactions on Computer Systems (TOCS) Volume 18 Issue 1. Febrero, 2000.
- [6] A. Campbell, I. katzela, K. Miki, J. Vicente. Open Signaling for ATM, INTERNET AND MOBILE NETWORKS (OPENSIG'98). University of Toronto, Ontario. Octubre, 1998.
- [7] A. Doria, J. Hadi, R. Hass, et al. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810, Marzo, 2010.
- [8] J. Schoenwaelder. Overview of the 2002 IAB Network Management Workshop. RFC 3535. IETF. Mayo, 2003.
- [9] J. Simeon, P. Wadler. The Essence of XML. ACM Symposium on Principles of Programming Languages. 2003.
- [10] N. Feamster et al. The Case for Separating Routing from Routers. AT&T Labs-Research. Septiembre, 2004.
- [11] D. Maltz et al. Network-Wide Decision Making: Toward A Wafer-Thin Control Plane. Carnegie Mellon University. Noviembre, 2004.
- [12] A. Greenber et al. A Clean Slate 4D Approach to Network Control and Management. Carnegie Mellon University. 2005.
- [13] H. Yang et al. Tesseract: A 4D Network Control Plane. Carnegie Mellon University. 2007.
- [14] M. Casado et al. Ethane: Taking Control of the Enterprise. Stanford University. October 2007.
- [15] M. Casado et al. NOX: Towards an Operating System for Networks. Nicira Networks. 2008.
- [16] Cisco Networking Academy. Connecting Networks Companion Guide. Cisco Press. Mayo, 2014.
- [17] J. Kurose, K. Ross. Computer Networking: A Top-Down Approach (6th ed.). Pearson. 2012.
- [18] R. Sundararajan. Software Defined Networking (SDN) - a definitive guide. Junio, 2013.

- [19] E. Haleplidis, K. Pentikousis, et al. Software-Defined Networking (SDN): Layers and Architecture Terminology. Internet Research Task Force (IRTF), RFC 7426. Enero, 2015.
- [20] H. Kim, N. Feamster. Improving network management with software defined networking. IEEE Communications Magazine. vol 51. no 2. pp. 114-119. 2013.
- [21] A. Porxas. Virtualization-enabled Adaptive Routing for QoS-aware Software-Defined (Master thesis). Universitat Politècnica de Catalunya. Diciembre, 2014.
- [22] D. Kreutz, F. Ramos, et al. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE, Vol.103, No. 1. Enero, 2015.
- [23] L. Richardson, S. Ruby. RESTful Web Services. Web Services for the Real World. O'Reilly Media, Mayo, 2007.
- [24] R. Fielding. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California. Irvine, 2000.
- [25] T. Hinrichs, N. Gude, M. Casado, et al. Practical declarative network management. Proceedings of the 1st ACM workshop on Research on enterprise networking. Agosto, 2009.
- [26] N. Foster et al. Frenetic: A network programming language. Proceedings of the 16th ACM SIGPLAN international conference on Functional programming. Septiembre, 2011.
- [27] C. Monsanto, N. Foster, R. Harrison, et al. A compiler and run-time system for network programming languages. Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. Enero, 2012.
- [28] J. Reich, C. Monsanto, N. Foster, et al. Modular SDN programming with pyretic. Usenix, The Advanced Computing Systems Association. Octubre, 2013.
- [29] B. Pfaff, B. Davie. The Open vSwitch Database Management Protocol. RFC 7047. IETF. Diciembre, 2013.
- [30] Y. Rekhter, T. Lid, S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271. Enero, 2006.
- [31] R. Enns, M. Bjorklund, J. Schoenwaelder, et al. Network Configuration Protocol (NETCONF). RFC 6241. Junio, 2011.
- [32] H. Yin, et al. SDNi: A message exchange protocol for software defined networks (SDNS) across multiple domains. Internet Engineering Task Force, Internet Draft. Junio, 2012.
- [33] JP. Vasseur, JL. Le Roux. Path Computation Element (PCE) Communication Protocol (PCEP). RFC 5440. Marzo, 2009.
- [34] T. Lakshman, T. Nandagopal, R. Ramjee, et al. The SoftRouter Architecture. Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking, 2004.

- [35] ITU. Management Framework For Open Systems Interconnection (OSI) For CCITT Applications. ITU Recommendation X.700. Septiembre, 1992.
- [36] F. Maccioni. Network Automation with Ansible 2.1 and beyond [PowerPoint slides]. Cisco. Septiembre, 2016.
- [37] M. Bjorklund. The YANG 1.1 Data Modeling Language. RFC 7950. Agosto, 2016.
- [38] A. Bierman, M. Bjorklund, K. Watsen. RESTCONF Protocol. RFC 8040. Enero, 2017.
- [39] GRPC. GRPC A High Performance, Open-Source Universal RPC Framework. <http://www.grpc.io>. 2015.
- [40] Open Networking Foundation. OF-CONFIG 1.2. OpenFlow Management and Configuration Protocol. ONF TS-016. 2014.
- [41] S. Milton, E. Kazmierczak, C. Keen. Data Modelling Languages: An Ontological Study. The 9th European Conference on Information Systems. Junio, 2001.
- [42] R. Sherwood, G. Gibb, K. Yap, et al. FlowVisor: A Network Virtualization Layer. Deutsche Telekom Inc. R&D Lab, Stanford University, Nicira Networks. Octubre, 2009.
- [43] A. Al-Shabibi, M. De Leenheer, et al. OpenVirteX: Make Your Virtual SDNs Programmable. HotSDN '14: Proceedings of the third workshop on Hot topics in software defined networking. ACM. Agosto, 2014.
- [44] Z. Bozakov, P. Papadimitriou. AutoSlice: automated and scalable slicing for software-defined networks. CoNEXT Student '12: Proceedings of the 2012 ACM conference on CoNEXT student workshop. ACM. Diciembre, 2012.
- [45] S. Gutz, A. Story, C. Schlesinger, et al. Splendid isolation: a slice abstraction for software-defined networks. Proceedings of the first workshop on Hot topics in software defined networks. 2012.
- [46] A. Voellmy, P. Hudak. Nettle: Functional Reactive Programming of OpenFlow Networks. Yale University. Enero, 2011.
- [47] P. Goransson, C. Black. Software Defined Networks, A Comprehensive Approach. Morgan Kaufmann. Mayo, 2014.
- [48] W. Xia, Y. Wen, et al. A Survey on Software-Defined Networking. IEEE Communication Surveys & Tutorials, Vol 17, No. 1, First Quarte 2015. 2015.
- [49] J. Skorupa, M. Fabbi, et al. Ending the Confusion About Software-Defined Networking: A Taxonomy. Gartner Research, G00248592. Marzo, 2013.
- [50] M. Mahalingam et al. VXLAN: A framework for overlaying virtualized layer 2 networks over layer 3 networks. Internet Engineering Task Force, Internet Draft. Noviembre, 2013.
- [51] M. Sridharan et al. NVGRE: Network virtualization using generic routing encapsulation. Internet Engineering Task Force, Internet Draft. Agosto, 2013.

- [52] B. Davie, J. Gross. A stateless transport tunneling protocol for network virtualization (STT). Internet Engineering Task Force. Abril, 2014.
- [53] K. Vikramajeet. Analysis of Openflow Protocol in Local Area Networks. Master of Science Thesis. Tampere University of Technology. Febrero, 2013.
- [54] Open Networking Foundation. OpenFlow Switch Specification, Version 1.0.0 Implemented (Wire Protocol 0x01). Diciembre, 2009.
- [55] Open Networking Foundation. OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0\*02). Febrero, 2011.
- [56] Jason Davis. Introduction to Software-Defined Networking (SDN) and Network Programability. CiscoLive BRKSDN-1014 [Power Point-Slides]. 2015.
- [57] T. Dierks, E. Rescorla. Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. Agosto, 2008.
- [58] A. Lara, A. Kolasani, B. Ramamurthy. Network Innovation Using OpenFlow: A Survey. IEEE Communications Survey & Tutorials. Agosto, 2013.
- [59] W. Braun, M. Menth. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. University of Tuebingen. Enero, 2014.
- [60] Open Networking Foundation. OpenFlow Switch Specification, Version 1.2.0 Implemented (Wire Protocol 0\*02). Diciembre, 2011.
- [61] Open Networking Foundation. OpenFlow Switch Specification, Version 1.3.0 Implemented (Wire Protocol 0\*02). Junio, 2012.
- [62] Open Networking Foundation. OpenFlow Switch Specification, Version 1.4.0 Implemented (Wire Protocol 0\*02). Octubre, 2013.
- [63] Open Networking Foundation. OpenFlow Switch Specification, Version 1.5.0 Implemented (Wire Protocol 0\*02). Diciembre, 2014.
- [64] Open Networking Foundation. OpenFlow Switch Specification, Version 0.8.9 Implemented (Wire Protocol 0\*97). Diciembre, 2008.
- [65] B. Lantz, B. Heller, N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks. Octubre, 2010.
- [66] J. Yan, D. Jin. VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation. ACM SIGCOMM Symposium on SDN Research 2015 (SOSR15), Santa Clara, CA. Junio, 2015.
- [67] S. Wang, C. Chou, C. Yang. EstiNet OpenFlow Network Simulator and Emulator, IEEE Communications Magazine (Volume: 51, Issue: 9). Septiembre, 2013.
- [68] Y. Wang, H. Kung. A New Methodology for Easily Constructing Extensible and High-Fidelity TCP/IP Network Simulators. Octubre, 2002.
- [69] J. Postel. Internet Control Message Protocol. RFC 792. Internet Engineering Task Force. Septiembre 1981.

- [70] B. Linkletter. OFNet SDN network emulator.  
<http://www.brianlinkletter.com/ofnet-a-new-sdn-network-emulator>.  
Noviembre, 2016.
- [71] D. Klein, M. Jarschel. An OpenFlow Extension for the OMNeT++ INET Framework. OMNeT++. Marzo, 2013.
- [72] A. Varga. INET Framework for the OMNeT++ Discrete Event Simulator.  
<http://github.com/inet-framework/inet>. 2012.
- [73] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In International Conference on Simulation Tools and Techniques for Communications, Networks and Systems. Marzo, 2008.
- [74] R. Khondoker, A. Zaalouk, et al. Feature-Based Comparison and Selection of Software Defined Networking (SDN) Controllers. Fraunhofer Institute for Secure Information Technology. 2013.
- [75] A. Centeno, C. Rodriguez, et al. Controladores SDN, Elementos para su Selección y Evaluación. Revista Telem@tica. Vol. 13. No. 3, septiembre-diciembre, 2014, p. 10-20. Diciembre, 2014.
- [76] G. Landi, G. Bernini, et al. COSIGN. Combining Optics and SDN In next Generation data centre Networks. Deliverable D3.1. SDN Framework Functional Architecture. Diciembre, 2014.
- [77] P. Morreale, J. Anderson. Software Defined Networking Design and Deployment. CRC Press. 2015.
- [78] G. Landi, G. Bernini, et al. COSIGN. Combining Optics and SDN In Next Generation Data Centre Networks. Deliverable D1.3. Comparative Analysis of Control Plane Alternatives. Enero, 2015.
- [79] A. Shalimov, D. Zuikov, et al. Advanced study of SDN/OpenFlow controllers. In Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia. Octubre, 2013.
- [80] D. Drutskey, E. Keller, et al. Scalable network virtualization in software defined networks. Internet Computing, IEEE.
- [81] M. Nascimento, C. Rothenberg, et al. Virtual routers as a service: the routeflow approach leveraging software defined networks. In Proceedings of the 6th International Conference on Future Internet Technologies, CFI '11. New York, NY, USA. ACM. pp. 34–37, 2011.
- [82] Cisco Systems. Introduction to Cisco IOS NetFlow - A Technical Overview. Mayo, 2012.
- [83] A. Myers. JFlow: Practical Mostly-Static Information Flow Control. Proceedings of the 26th ACM Symposium on Principles of Programming Languages (POPL '99). Enero, 1999.
- [84] S. Chowdhury, D. Cheriton, et al. PayLess: A low cost network monitoring framework for Software Defined Networks. 2014 IEEE Network Operations and Management Symposium (NOMS). Mayo, 2014.

- [85] A. Tootoonchian, M. Ghobadi, et al. OpenTM: Traffic Matrix Estimator for OpenFlow Networks. Proceedings of the 11th international conference on Passive and active measurement. Abril, 2010.
- [86] C. Yu, C. Lumezanu, et al. Flowsense: monitoring network utilization with zero measurement cost. Proceedings of the 14th International Conference on Passive and Active Measurement, PAM'13. Marzo, 2013.
- [87] Canini, D. Venzano, et al. A Nice Way to Test OpenFlow Applications. NSDI. Abril, 2012.
- [88] H. Mai, A. Khurshid, et al. Debugging the data plane with ant eater. In Proceedings of the ACM SIGCOMM 2011 conference, SIGCOMM '11, New York, NY, USA. pp. 290–301. 2011.
- [89] A. Khurshid, W. Zhou, et al. Veriflow: verifying network-wide invariants in real time. In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, New York, NY, USA. pp. 49–54. 2012.
- [90] A. Wundsam, D. Levin, et al. Ofrewind: enabling record and replay troubleshooting for networks. In Proceedings of the 2011 USENIX conference on USENIX annual technical conference, USENIXATC'11, Berkeley, CA, USA. pp. 29–29. 2011.
- [91] N. Handigol, B. Heller, et al. Where is the debugger for my softwaredefined network? In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, New York, NY, USA. pp. 55–60. 2012.
- [92] Open Networking Foundation. SDN in the Campus Environment. ONF Solution Brief. Septiembre, 2013.
- [93] Y. Jarraya. A Survey and a Layered Taxonomy of Software-Defined Networking. IEEE Communication Surveys & Tutorials. 2014.
- [94] S. Mehdi, J. Khalid, et al. Revisiting traffic anomaly detection using software defined networking. In Recent Advances in Intrusion Detection. Springer. 2011.
- [95] A. Nayak, A. Reimers, et al. Resonance: Dynamic access control for enterprise networks. In Proceedings of the 1st ACM workshop on Research on enterprise networking. ACM, 2009.
- [96] M. Reitblatt, N. Foster, et al. “Abstractions for network update” en ACM SIGCOMM. 2012.
- [97] H. Nilsson, A. Courtney, et al. Functional Reactive Programming, Continued\*. Yale University. Proceedings of the 2002 ACM SIGPLAN workshop on Haskell. 2002.
- [98] K. Yap, R. Sherwood, et al. Blueprint for introducing innovation into wireless mobile networks. In Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures. ACM. pp. 25–32. 2010.
- [99] K. Yap, M. Kobayashi, et al. Openroads: Empowering research in mobile networks. ACM SIGCOMM Computer Communication Review. Vol. 40 no. 1. pp.125–126. 2010.

- [100]P. Dely, J. Vestin, et al. CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud. In Globecom Workshops. IEEE. Diciembre, 2012.
- [101]L. Suresh, J. Zander, et al. Towards Programmable Enterprise WLANS with Odin. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks. Agosto, 2012.
- [102]D. Awduche, A. Chiu, et al. Overview and Principles of Internet Traffic Engineering. RFC 3272. IETF. Mayo, 2002.
- [103]I. Akyldiz, A. Lee, et al. A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks. Enero, 2014.
- [104]W. Kim, P. Sharma, J. Lee, et al. Automated and Scalable QoS Control for Network Convergence. Usenix, INM/WREN'10 Proceedings of the 2010 internet network management conference on Research on enterprise networking. Abril, 2010.
- [105]A. Mirchev. Survey of Concepts for QoS improvements via SDN. Seminar Future Internet SS2015. Facultad de Informática de la Universidad Técnica de Múnich. Septiembre, 2015.
- [106]H. Egilmez, S. Dane, et al. "OpenQoS: OpenFlow controller design and test network for multimedia delivery with quality of service". Proc. NEM Summit, Implementing Future Media Internet Towards New Horizons. Diciembre, 2012.
- [107]K. Noghani, M. Sunay. Streaming Multicast Video over Software-Defined Networks. 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems. Octubre, 2014.
- [108]ETSI. Network Functions Virtualization Introductory White Paper. Octubre, 2012.
- [109]S. Rao. SDN and its USE-CASES - NV and NFV. A State-of-the-Art Survey. NEC Technologies India Limited.
- [110]A Tong, K. Wade. NFV and SDN Guide for Carriers and Service Providers.
- [111]E. Ander-Egg. Técnicas de Investigación Social. Editorial Lumen. 24<sup>a</sup> Edición. 1995.
- [112]M. Kobayashi, S. Seetharamam, G. Parulkar, et al. Maturing of OpenFlow and Software-defined Networking through deployments. Computer Networks. 2013.
- [113]David Erickson et al., A demonstration of virtual machine mobility in an OpenFlow network, in: Proc. of ACM SIGCOMM (Demo). 2008.
- [114]N. Handigol, S. Seetharaman, M. Flajslik, et al. Plugn-Serve: Load-balancing web traffic using OpenFlow, in: Proc. of ACM SIGCOMM (Demo), 2009.
- [115]M. Diogo, F. Natalia, T. da Costa, et al. OMNI: OpenFlow MaNagement Infrastructure. 2011 International Conference on the Network of the Future Noviembre, 2011.

- [116]P. Godfrey, I. Ganichev, S. Shenker, et al. Pathlet Routing. ACM SIGCOMM 2009. Agosto, 2009.
- [117]A. Anand, V. Sekar, A. Akella. SmartRE: An Architecture for Coordinated Network-wide Redundancy Elimination. ACM SIGCOMM. 2009.
- [118]B. Heller, R. Sherwood, N. McKeown. The controller placement problem. Proceedings of Hot Topics in Software Defined Networking (HotSDN). 2012.
- [119]K. Yap, M. Kobayashi, D. Underhill, et al. The stanford OpenRoads deployment. Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization (WINTech). 2009.
- [120]Open Networking Foundation. Migration Use Cases and Methods. Migration Working Group. 2014.
- [121]S. Rao. SDN Series Part Eight: Comparison of Open Source SDN Controllers. The New Stack. Marzo, 2015.
- [122]A. Metzler & Associates. Ten Things to Look for in an SDN Controller.
- [123]O. Salman, A. Kayssi, A. Chehab. SDN Controllers: A Comparative Study. Abril, 2016.
- [124]G. Romero. Evaluation of OpenFlow Controllers. Octubre, 2012.
- [125]S. Shah, J. Faiz, M. Farooq, et al. An Architectural Evaluation of SDN Controllers. IEEE International Conference on Communications (ICC). Junio, 2013.
- [126]J. Xie, D. Guo, Z. Hu, et al. Control Plane of Software Defined Networks: A Survey. Elsevier, Computer Communications. Mayo, 2015.
- [127]A. Tavakoli, M. Casado, S. Shenker. Applying NOX to the Datacenter. Enero, 2009.
- [128]A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood, On controller performance in software-defined networks, in: Proc. USENIX Hot-ICE, San Jose, CA. 2012.
- [129]S. Kandula, S. Sengupta, A. Greenberg, et al. The nature of data center traffic: Measurements & analysis, in: Proc. ACM SIGCOMM IMC, Chicago, Illinois, USA. 2009.
- [130]D. Erickson. The Beacon OpenFlow Controller. In: Proc. ACM HotSDN, Hong Kong, China. 2013.
- [131]A. Shalimov, R. Smeliansky, V. Pashkov. Advanced study of SDN/OpenFlow controllers. Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia. Octubre, 2013.
- [132]S. Yeganeh, Y. Ganjali. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. Proceedings of the first workshop on Hot topics in software defined networks, ACM SIGCOMM. Agosto, 2012.
- [133]B. Heller, R. Sherwood, N. McKeown. The Controller Placement Problem. HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks. Agosto, 2012.

- [134]Y. Jimenez, C. Cervello-Pastor, A. Garcia. On the controller placement for designing a distributed sdn control layer. 2014 IFIP Networking Conference. Junio, 2014.
- [135]Y. Hu, W.Wang, X. Gong, et al On reliability-optimized controller placement for software-defined networks. China Communications. Febrero, 2014.
- [136]A. Linguori, M. Winandy. The Diamond Approach for SDN Security. IEEE Softwarization. Marzo, 2018.
- [137]Y. Hori, S. Mizoguchi, R. Myyazaki, et al. A Comprehensive Security Analysis Checksheet for OpenFlow Networks. Conference: International Conference on Broadband and Wireless Computing, Communication and Applications. Noviembre, 2017.
- [138]D. Kreutz, F. Ramos, P. Verissimo. Towards Secure and Dependable Software-Defined Networks. ACM. Agosto, 2013.
- [139]S. Hayward, S. Natarajan, S. Sezer. A Survey of Security in Software Defined Networks. IEEE Communication Surveys & Tutorials. Julio, 2015.
- [140]ONF. Principles and Practices for Securing Software-Defined Networks. ONF TR-511. Enero, 2015.
- [141]Z. Yan, R. MacLavery. Autonomic Trust Management in a Component Based Software System. In Proceedings of the 3rd International Conference on Autonomic and Trusted Computing (ATC2006). Septiembre, 2006.
- [142]R. Holz, T. Riedmaier, N. Kammenhuber, et al. X.509 Forensics: Detecting and Localising the SSL/TLS Man-in-the-Middle. Conference: European Symposium on Research in Computer Security. Septiembre, 2012.
- [143]Y. Desmedt, Y. Franke. Threshold Cryptosystems. In Proceeding CRYPTO '89 Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology. Agosto, 1989.
- [144]J. Perkins, S. Kim, S. Larsen, et al. Automatically Patching Errors in Deployed Software. In Proceedings of ACM SIGOPS 22nd symposium on Operating systems principles. Octubre, 2009.
- [145]ONF. Migration Tools and Metrics. Migration Working Group. ONF TR-507. 2014.
- [146]J. Roper. Software Defined Networking: Should Do Now? Should Do Never? Simply Don't Know!. Entuity Network Analytics.
- [147]sFlow. Traffic Monitoring using sFlow. 2003.