
**Maestría en Redes de Datos**

Nombre y Apellido del alumno: Daniel Alberto Priano

Director: Ing. Luis Marrone

Asesor científico: Dra. María Claudia Abeledo

Título del Tema de Tesis:

# Análisis de protocolos de enrutamiento en Redes definidas por software (Software Defined Networks)

# Contenido

## Agradecimientos

A mi gran maestro, director y amigo, Luis Marrone

A mi amiga y colega, María Claudia Abeledo

A Javier Guevara y Fabio Bruschetti, que tanto me alentaron

A mi hijo, Guido Priano.

A Amelia

# Capítulo 1: Introducción

## Objetivo específico:

Las redes definidas por software (SDN: Software Defined Networks) han generado un nuevo paradigma cambiando el enfoque sobre anteriores conceptos. En esta tesis se pretende mostrar la relación e interacción de las denominadas SDN con las políticas y protocolos de ruteo tanto sobre conceptos preexistentes como los cambios, si hubiere, que se producen a partir del surgimiento de las SDN mencionadas. Se pretende evaluar la problemática de enrutamiento de este nuevo paradigma comparándolo con la performance obtenida en escenarios tradicionales, es decir, incluir en la comparación, los procesos de ruteo de los protocolos instalados en los enrutadores tradicionales. Además, se evaluará la variación de dicha performance cuando se utilizan mecanismos basados íntegramente en implementaciones de software ad hoc, es decir, sobre políticas que incluyen la generación del código, como por ejemplo, OpenVSwitch [1], entre otros. Estos alcances forman parte del estudio realizado a través del trabajo de tesis.

Dentro del desarrollo de las SDN se han descrito áreas de oportunidad que en conjunto pueden proporcionar una solución integral para ofrecer calidad de servicio. Algunas de las áreas identificadas son: monitoreo de la red, algoritmos de enrutamiento, ingeniería de tráfico, calendarización y priorización de flujos.

El objetivo del enrutamiento con calidad de servicio (QoS) no sólo consiste en encontrar el camino de costo mínimo con respecto a una determinada métrica, sino brindar un camino que cumpla los requisitos para el desempeño estable de la red y las aplicaciones. Además de la tasa de transmisión, los enlaces disponen de una serie de atributos que sirven como métricas del desempeño. [2]

## Motivación

Como se mencionara anteriormente, la frase "red definida por software" se corresponde con el acrónimo SDN (en inglés Software Defined Network). Este se acuñó cuando fue necesario distinguir las redes definidas por software antes mencionadas de la definición del concepto de las redes basadas en hardware, desde ahora en más, “tradicionales”. Desde ese momento, "SDN" se ha convertido en el tipo de configuración dinámica que tiene lugar siempre que los servicios basados en software en una red ubicada en un centro de datos, sean accesibles a través de una dirección de Protocolo de Internet (IP).[3].

Los protocolos de enrutamiento heredados como OSPF[59] y BGP[4] se han desarrollado de forma muy completa, pero su sistema complejo y rígido ha sido difícil de adaptar a los requerimientos actuales en los servicios de los usuarios a través de una red como Internet. Dentro de los mencionados protocolos de enrutamiento convencionales también se incluyen RIP[5] e EIGRP[6]. Todos ellos, en su conjunto, debido al sistema rígido e intrincado mencionado, reducen su capacidad de adaptación a las redes cada vez más complejas en cuanto al volumen y tipos de datos a transmitir. La aparición de Software Defined Network (SDN) ha traído nuevos conceptos para la solución de este problema.

También, debido al aumento de las fallas imprevistas que tienen lugar, la capacidad de predecir / conocer el tiempo máximo aproximado que tardan estas redes en converger para evitar y / o minimizar la pérdida de paquetes / datos durante estas fallas se ha vuelto crucial en el mundo de hoy.

Los enrutadores de la red tardan en converger a través del protocolo de enrutamiento implementado y reanudan la comunicación o transferencia de información,

lo que se denomina tiempo de convergencia de enrutamiento. Este tiempo de convergencia está obviamente también presente en SDN por lo que es importante a la hora de evaluar el rendimiento abordar la comparación del enrutamiento en SDN con el enrutamiento convencional.

# Capítulo 2: Estado del Arte

## Introducción al Estado del Arte en SDN

Para establecer las categorías y subcategorías desarrolladas en la presente revisión del estado del arte, se seleccionó el método de índice [7]. Este método consiste en desarrollar, en primer lugar, un índice SDN global o general, y refinarlo hasta que sea lo suficientemente específico en las diferentes categorías o subcategorías del estudio. En este sentido, se encuentra que en la última década, el desarrollo de la investigación SDN ha aumentado en las categorías de estudio tales como: Arquitectura, Controladores, NFV (Network Function Virtualization) [8], Aplicaciones y Reglamentos.

Las redes definidas por software (SDN) ayudan a las organizaciones a acelerar la implementación y entrega de aplicaciones, reduciendo drásticamente los costos de TI a través de la automatización del flujo de trabajo habilitada por las políticas. La tecnología SDN permite arquitecturas en la nube al proporcionar entrega automatizada de aplicaciones bajo demanda y movilidad a escala. SDN mejora los beneficios de la virtualización del centro de datos, lo que aumenta la flexibilidad y utilización de los recursos y reduce los costos de infraestructura y los gastos generales.

SDN logra estos objetivos empresariales al converger la administración de servicios de red y aplicaciones en plataformas de implementación extensibles y centralizadas que pueden automatizar el aprovisionamiento y la configuración de toda la infraestructura. Las políticas de TI comunes y centralizadas reúnen grupos de TI y flujos de trabajo dispares. El resultado es una infraestructura moderna que puede ofrecer nuevas aplicaciones y servicios en minutos, en lugar de los días o semanas requeridos en el pasado.

Este nuevo enfoque lleva a la definición de red definida por software (SDN), caracterizada a través de cinco aspectos fundamentales:

a) Separación de planos (control, datos y gerenciamiento),

b) Dispositivos programables y de bajo costo,

c) Control centralizado (a través de un controlador que pueda gestionar dichos dispositivos),

d) Virtualización y automatización de la red

e) Caracterización de la apertura del código y estructuras a utilizar (Open Source).

La definición enunciada marca algunos de los aspectos sobresalientes en las mejoras realizadas sobre las políticas de TI, enunciadas anteriormente.

En otro orden, y atendiendo a características generales, los resultados experimentales muestran que la recuperación del enrutamiento en la red SDN tiene una ventaja en la topología de red a gran escala. En comparación con el enrutamiento SDN, la convergencia de enrutamiento en la red heredada está mucho más influenciada por el retraso del enlace. Cuando la demora de enlace de la red es alta y la red es grande, el tiempo de convergencia de enrutamiento en la red SDN es menor que la red heredada.

El beneficio que ofrece la ventaja del control centralizado, permite a SDN lograr eficiencia en el cálculo de enrutamiento y control de grano fino para paquetes. Existe bibliografía que estudia el rendimiento de convergencia de los mecanismos de enrutamiento heredados y el enrutamiento SDN midiendo el rendimiento en términos de retardo de reenvío de paquetes y tiempo de convergencia después del fallo de enlace / nodo.

El paradigma de redes definidas por software (SDN) intenta mejorar el rendimiento de la red, facilitando la gestión de la misma y su escalabilidad basándose en software de código abierto y agregando una nueva entidad llamada controlador que gestiona toda la red. Uno de los objetivos del controlador es tomar decisiones con respecto al enrutamiento en lugar de distribuirlo entre los nodos de la red, como de costumbre. Sin embargo, este campo dentro de SDN necesita más trabajo porque no está claramente definido cómo funcionarán los protocolos de enrutamiento tradicionales a través de una SDN.

Desde la perspectiva expuesta, SDN Architecture (Arquitectura SDN) ha consolidado las subcategorías objeto de estudio, tales como: plano de control, plano de datos y plano de aplicación. En Controladores, subcategorías en dispositivos de código abierto, como ONOS (Open Network Operating System)[9] y ODL(Open Day Light) [10], y fuente patentada, como VMware NSX [11] y Huawei Agile[12]. Con respecto a NFV, los componentes asociados se destacan como subcategorías, así como la relación de NFV con SDN. En términos de aplicaciones relacionadas con SDN, las subcategorías se han centrado en desarrollos como: IoT (Internet of things), Datacenter, 5G y Big data.

Y en términos de regulaciones, se han establecido subcategorías regulatorias sectoriales: ONF [13], IRTF [14] e ITU [15].

Como base de documentación, se han utilizado fuentes, entre otras, como IEEE Xplore [16], las bases de datos de Google Scholar, así como la documentación de ONF e ITU; con búsqueda de palabras clave como arquitectura SDN, controladores SDN, NFV para SDN, aplicaciones SDN y regulaciones para SDN.

Para el análisis de los documentos de esta investigación, de carácter documental y exploratorio, se asumieron fuentes en cada una de las categorías y subcategorías de temas de acuerdo a los subtítulos, circunscribiendo el continente europeo y países como EE.UU., y temporalmente limitadas a la última década, ya que está en el espacio

y tiempo en el que se han desarrollado las mayores investigaciones y aplicaciones basadas en SDN. Como consecuencia, se proporcionará un contexto para este paradigma de red. La categorización se observa, con las tendencias tecnológicas actuales en las Figuras 2-1 y 2-2.

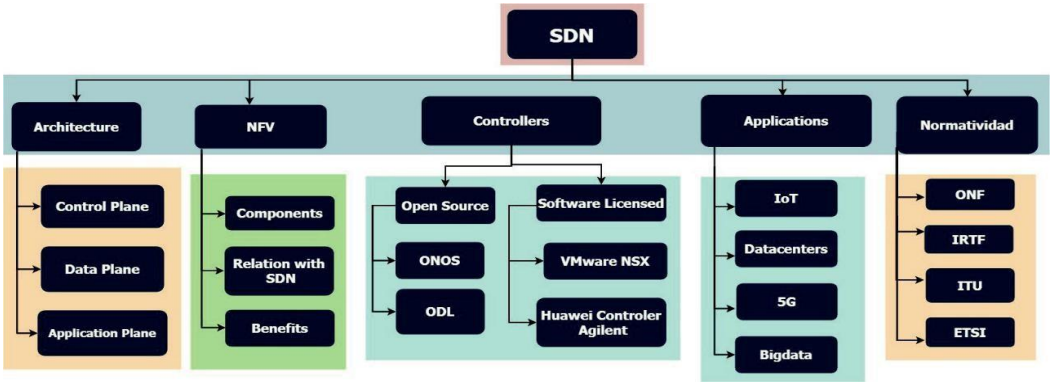



**Figura 2-1:** Modelo de investigación para el estado del arte desarrollado .[17]

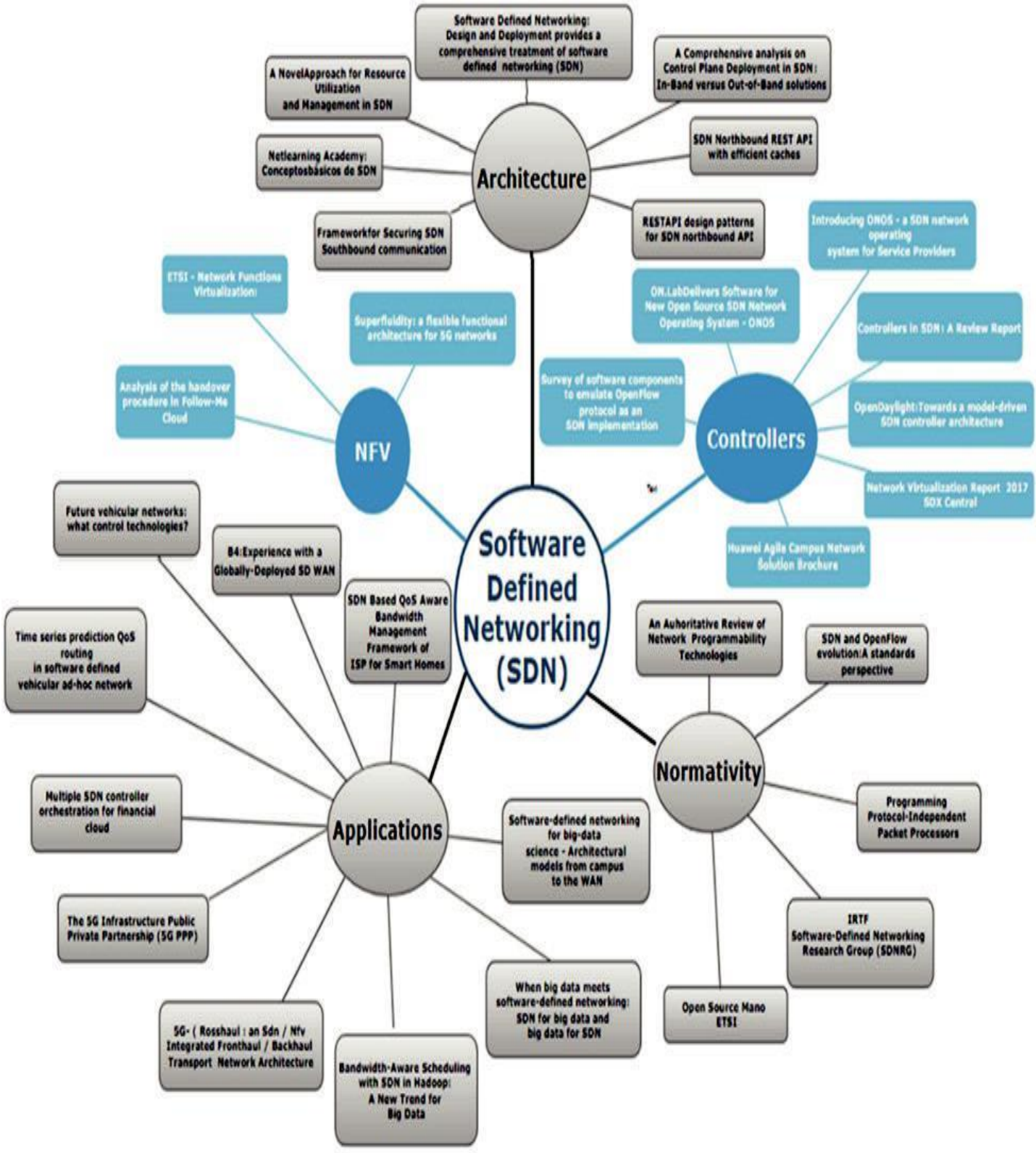



**Figura 2-2.** Subcategorización del modelo de investigación para el desarrollo del estado del arte [13]

## Ingeniería de Tráfico

La Ingeniería de tráfico, (TE o Traffic Engineering), es un mecanismo importante para optimizar el rendimiento de la red de datos, analizando dinámicamente, prediciendo y regulando el comportamiento de los datos transmitidos por la red, como se manifiesta en [18].

Se puede decir, entonces, que la mencionada TE estudia la medición y gestión del tráfico de red y diseña mecanismos de enrutamiento razonables para guiar el tráfico de red a fin de mejorar la utilización de los recursos de red y cumplir los requisitos de la calidad de servicio (QoS) de la misma.

En comparación con las redes tradicionales, SDN tiene muchas ventajas para ser compatible con TE debido a sus características distintivas, como el aislamiento de los planos de control y datos, el control centralizado global y la programabilidad del comportamiento de la red.

Un ejemplo clásico de distribución de tráfico en la red, lo que origina la necesidad de la implementación de ingeniería de tráfico es la solución al problema del pez, mostrado en la figura 2-3, donde los routers r6 y r7 formarían la cola y el router r8 la cabeza del pez.

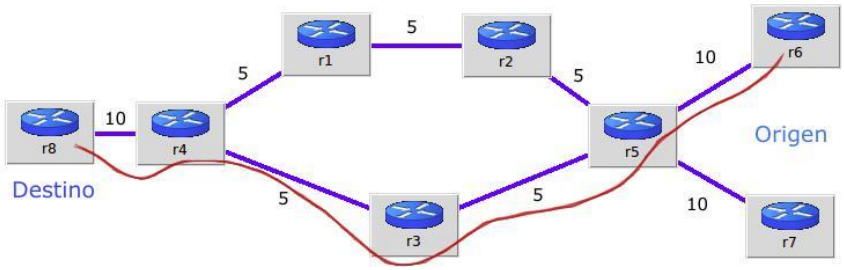



**Figure 2-3: Problema del Pez. Fuente: Cisco System**

En la figura 2-3, los enlaces interiores de la red tienen un ancho de banda de 20 Mbps, su costo resulta de 5, de igual forma ocurre con los enlaces extremos, cuyo ancho de

banda es de 10 Mbps y su costo de 10, ya que el costo del enlace en OSPF es $C_i = 10^8$/ancho de banda (bps).

OSPF utiliza como costo de la ruta la sumatoria de todos los costos de los enlaces individuales que conforman el camino. Siendo el camino óptimo aquel que minimiza el resultado.

La ruta elegida resulta r6–r5–r3–r4–r8, la cual tiene un costo de 30, mientras que la r6-r5-r2-r1-r4-r8 tiene un costo de 35.

El problema que surgía en las redes tradicionales en esta topología, es que los protocolos de enrutamiento establecen la mejor ruta basándose en parámetros como el costo y envían todo el tráfico por esa ruta. Es decir que, en este caso, todo el tráfico sería enviado por la ruta de menor coste, r6–r5–r3–r4–r8, marcada con la línea roja de la figura.

A la vista salta el pésimo aprovechamiento de la red, que es solucionado en las redes tradicionales de diferentes formas. Una de ellas es la utilización de MPLS-TE[19], que establecería dos túneles, uno por cada ruta y conseguiría aprovechar de manera óptima la red.

Si bien es cierto que SDN presenta ventajas, no significa que hasta ahora las técnicas de TE no fueran eficientes. Existen gran variedad de técnicas TE, la mayoría de ellas implementadas sobre redes MPLS (Multiprotocol Label Switching).

Las redes SDNs posibilitan la Ingeniería de tráfico de manera natural, permitiendo la coordinación de políticas entre distintos ISPs y combinan las facilidades de los equipos con la visión en un plano general. [20]

En este punto podemos destacar la complicada implementación de MPLS TE que por tratarse de un multiprotocolo utiliza muchos recursos de la red. A su vez, la aplicación de TE determina la modificación de los protocolos de enrutamiento para que puedan transportar los atributos típicos de la ingeniería de tráfico. Uno de ellos, como ejemplo, es el ancho de banda disponible, calculado dinámicamente en los nodos para conocer el volumen de tráfico cursado a través de los enlaces. Sus registros conforman la tabla TE, utilizada en la selección de los túneles, que a su vez maneja, por ejemplo, prioridades y reservas de recursos. No se profundizará sobre este tema, ya que no es el objetivo del presente trabajo.

## Calidad de Servicio (QoS)

Con la aparición de las aplicaciones multimedia, las características del tráfico en las redes se transformaron en críticas para el soporte de aquellas aplicaciones que requerían, fundamentalmente, un bajo retardo, bajo Jitter o reducida pérdida de paquetes.

Para lograr que las redes puedan transmitir los datos que se envían en estas aplicaciones críticas se utilizan los modelos de calidad de servicio, que le permiten a los usuarios disponer de aplicaciones multimedia de calidad razonable acorde con un contrato de servicio entre el cliente y el ISP (internet Service Provider) que especifica el servicio de "forwarding" que el cliente recibirá.

El contrato que especifica los parámetros de QoS acordados entre el proveedor y el usuario se denomina SLA (Service Level Agreement [21]) o Acuerdo de Nivel de Servicio.

En líneas generales, la implementación de QoS en los routers de una red IP consta de las siguientes fases:

1. Marcado de los paquetes conforme a un código obtenido de las cabeceras de los paquetes en curso, pudiéndolos agrupar según algún criterio o regla.
2. Una vez marcados los paquetes se verán afectados por un tratamiento diferenciado que permitirá la monitorización de tráfico, ya que a través de la me-

dición y el remarcado obtendrán los recursos necesarios, según el tratamiento en las colas de los buffer y ancho de banda, permitiendo el cumplimiento del acuerdo de servicio.

Dos de las arquitecturas que se emplean son los servicios integrados (Integrated Services, IntServ) y los servicios diferenciados (Differentiated Services, DiffServ).

El objetivo de los Servicios Integrados (IntServ) es preservar el modelo de datagramas de las redes basadas en IP y al mismo tiempo soportar reservas de recursos para aplicaciones en tiempo real.

Los routers tienen que guardar una cierta información de estado de cada flujo, efectuándose una reserva de recursos (RSVP o Resource Reservation Protocol)[21] en todos los nodos que participan del camino, es decir, que los paquetes que pertenecen a un mismo flujo utilizarán el mismo camino, emulando la conmutación de circuitos.

Para que un flujo pueda recibir QoS debe lograr su ingreso por el control de admisión, quien verificará la existencia de recursos en cada router. La existencia de los recursos de dichos routers que conforman el camino son los que permiten la admisión del flujo en la red para su arribo al destino.

Si hay recursos disponibles, el flujo se admite y cada router guarda la información de su estado (soft state). Para mantener las reservas y las aplicaciones deben renovar los pedidos de recursos periódicamente durante la transmisión generando overhead en la red.

Los flujos comparten los recursos disponibles en la red. Esto recibe el nombre de enlace compartido (link sharing). Por ejemplo, el ancho de banda de la red es compartido por varios flujos los cuáles pueden contener distintos tipos de tráfico.

Ahora bien: La idea básica de los Servicios diferenciados (DiffServ) es clasificar los diferentes flujos de datos en clases o agregados y asignarles QoS.

El conjunto de nodos que operan dentro de un dominio DiffServ tienen una misma política de aprovisionamiento de servicios comunes y un conjunto de grupos PHB [22](tratamiento diferenciado) implementados en cada nodo.

Las operaciones de clasificación, marca, política, y adaptación de tráfico se realizan normalmente en los nodos frontera, es decir, al ingresar al dominio DiffServ acorde a un Contrato de Condiciones de Tráfico (Traffic Conditioning Agreement o TCA[23]). El mencionado TCA, especifica las reglas de clasificación y el perfil de tráfico (medición, marcado, descarte y shaping).

Los nodos frontera interconectan distintos dominios con los nodos internos e implementan funciones para traducir los distintos PHB (Per Hop Behavior) entre dominios. Además, se implementan funciones para hacer cumplir el mencionado TCA.

Los diferentes tipos de servicios son brindados a lo largo de la frontera del dominio identificando el tráfico y asignándole el comportamiento por salto que tendrá dentro del dominio.

Para lograr el acondicionamiento debe realizarse la medición, el cumplimiento de las políticas y modelado o ajuste del retardo y/o remarcado para asegurar que el tráfico dentro del dominio cumpla con lo estipulado por el TCA.

El reenvío en cada nodo (forwarding) utiliza un comportamiento por salto (PHB: Per Hop Behavior). Es el tratamiento diferenciado que recibe cada paquete individual acorde con las disciplinas de servicio de colas específicas, cuyos mecanismos no entran en la estandarización

Los PHB se realizan en cada nodo de la red, con independencia de cómo se construyan los servicios extremo a extremo o interdominio, brindando un tratamiento diferenciado a diferentes clases de tráfico.

Por otra parte, la admisión de los flujos se realiza en los nodos frontera y se puede configurar manual o dinámicamente. En el caso de la configuración dinámica se utilizará RSVP [24], lo cual reduce las posibilidades de escalabilidad. Aun así, la escalabilidad es mayor en comparación con la metodología que utilizan los Servicios Integrados.

Quizás la diferencia más grande entre los dos sistemas se plantee en términos de la escalabilidad en grandes redes, debido a que en el modelo de IntServ la reserva de recursos y el manejo se realiza por flujos, que si bien permite una mayor granularidad, utiliza mayores recursos de ancho de banda de los enlaces de la red que en el modelo de DiffServ.

Las redes Diffserv clasifican los paquetes según un pequeño número de grupos de flujos o agregados identificados con el punto de código de servicios diferenciados (DSCP) [25], información suministrada en el campo TOS (Type of Service) de IPv4 o del campo Traffic Class de IPv6 de cada encabezado IP del paquete. Por lo tanto, además de eliminar la dependencia del estado de cada flujo, DiffServ puede implementar el aprovisionamiento de calidad de servicio sin la necesidad de señalización de extremo a extremo.

En cuanto a las características formales de la calidad de servicio en redes o QoS (Quality of Service), se puede explicitar como un conjunto de estándares y mecanismos para garantizar un rendimiento de calidad para aplicaciones críticas. Mediante el uso de mecanismos QoS, los administradores de red pueden usar los recursos existentes de manera eficiente y garantizar el nivel de servicio requerido sin expandir de forma reactiva ni aprovisionar en exceso o sobredimensionar sus redes.

El concepto de calidad de servicio es aquel en el que los requisitos de algunas aplicaciones y usuarios son más críticos que otros, lo que significa que parte del tráfico necesita un tratamiento preferencial. Ese es el objetivo de QoS, proporcionar un servicio de entrega preferencial para las aplicaciones que lo necesitan asegurando un ancho de banda suficiente, controlando la latencia y reduciendo la pérdida de datos, como se explica en [26].

En referencia a QoS, con SDN se puede tener el control de QoS centralizado, dejando a los routers que se encarguen de sus cometidos y mediante el controlador SDN implementar QoS directamente en los switches. Esto soluciona los problemas ocasionados en una red de mayor tamaño y facilitan la gestión de una forma eficaz de la QoS de una red en su totalidad. Con esta mejora se puede en cualquier momento cambiar las normas de QoS dinámicamente, haciendo así que usar SDN sea realmente útil.

## Balanceo de carga

En cuanto al Balanceo de carga, hoy en día, nuestras redes tienen que manejar una gran cantidad de tráfico, atender a miles de clientes y cumplir los requisitos impuestos. Es muy difícil para un solo servidor soportar una carga tan grande. La solución es usar múltiples servidores con un balanceador de carga que actúa como interfaz.

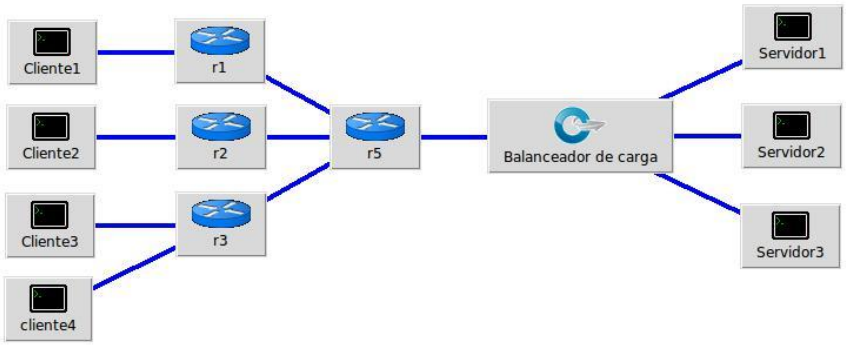



Figura 2-4: Balanceo de carga - Fuente: Cisco System

De esta forma, y como se muestra en la figura 2-4, cuando los clientes envíen sus peticiones el balanceador de carga las recibirá y reenviará a los diferentes servidores según la estrategia de balanceo de carga.

De las estrategias más comunes es la llamada Round Robin, que consiste en distribuir las peticiones secuencialmente entre el grupo de servidores. Otra de ellas es conocida como Least Connections, donde se envía la nueva petición al servidor con la menor cantidad de conexiones con clientes en ese momento. También se usa la estrategia de IP hash en la que la dirección IP del cliente es utilizada para determinar que servidor le atenderá.

El balanceador de carga usa un hardware dedicado que es costoso e inflexible. Los balanceadores de carga actualmente contienen pocos algoritmos que se pueden utilizar. Los administradores de red no pueden usar sus propios algoritmos de balanceo puesto que los balanceadores de carga no son programables, están limitados por el vendedor (Vendor locked).

La creciente demanda en el uso de los servicios ofrecidos en internet genera sobrecargas, es decir, aumentos repentinos de tráfico en las redes ocasionando la congestión de los enlaces.

En 2009, el 15% de la congestión en una red de centro de datos conformada por 150 switches y 1500 servidores duraba más de 100 segundos, aun cuando muchos de los enlaces se encontraban subutilizados; esto implica una necesidad considerable de optimizar la asignación de recursos de red. [27]

La mayoría de los centros de datos opta por la implementación de un balanceador de carga con el fin de satisfacer el gran número de usuarios y los requerimientos que estos generan.

Las estrategias de balanceo implementadas a los servicios web (HTTP) suelen basarse en las métricas de enlace proporcionadas al momento de la instalación y no consideran las condiciones de red dinámicas relativas a la utilización del ancho de banda, la pérdida de paquetes y los retrasos. Por lo tanto, las mismas carecen de flexibilidad para ajustarse a los distintos requerimientos de la red, o del tráfico o de las aplicaciones que operan sobre ésta. [28]

Las redes deben gestionar grandes volúmenes de tráfico sin introducir tiempos de espera innecesarios y garantizar la disponibilidad de servicios. En el artículo [29] se propone un algoritmo de balanceo de carga basado en un sistema de criterios combinados (ancho de banda disponible y retardo) que utiliza tecnologías de redes definidas por software para obtener en tiempo de ejecución, evalúa parámetros obtenidos desde distintos puntos de la red. Este conjunto de parámetros permite seleccionar aquel servidor que esté en las mejores condiciones para responder entre un conjunto de servidores que almacenan y distribuyen la misma aplicación. De esta manera, los resultados mejoran hasta un 50% el tiempo de respuesta del servidor en comparación con el método tradicional Round Robin.

El equilibrio de carga del servidor basado en SDN en comparación con el método tradicional de balanceo de carga, puede mejorar efectivamente el rendimiento del equilibrio de carga del servidor y reducir la complejidad de la implementación. [26]

Utilizando OpenFlow como protocolo de interfaz con el conmutador, la esencia del problema del equilibrio de la carga en un servidor basado en SDN es la creación dinámica de tablas de flujo.

A través de la utilización de un algoritmo que permite el diseño de tablas de flujo dinámicas, pueden combinarse "tablas de flujo individuales" con una "tabla de flujo grupal".

La "tabla de flujo individual" puede monitorear el tráfico de cada cliente mientras que una "tabla de flujo grupal" permite clasificar eficazmente los hosts de los mismos.

Este algoritmo mencionado anteriormente, evita un número excesivo de tablas de flujo y también resuelve el defecto que se genera cuando el número de coincidencias

en la tabla de flujo es demasiado amplio, demostrando buena factibilidad y un mayor rendimiento de programación de tráfico de la red. [26]

Hasta aquí se destaca que, mediante SDN, es posible solucionar algunos de los problemas a los que se enfrentan las redes tradicionales.

Por último, otra ventaja de SDN se basa en el balanceador de carga, el cual presenta ciertas ventajas en comparación con el método tradicional. Puede mejorar efectivamente el rendimiento del balanceador y reducir la complejidad de la implementación. Los balanceadores de carga SDN son programables y permiten diseñar e implementar una estrategia de equilibrio de carga propia. Otras virtudes del balanceador de carga SDN es que no se necesita hardware dedicado, ahorrando así coste en la red. Un mismo switch se puede convertir en un potente balanceador mediante el uso de controladores SDN.

## Encaminamiento o ruteo multicamino en SDN

Uno de los primeros conceptos a analizar en esta sección, son las aplicaciones que relacionan el balanceador de carga con routing multicamino, sobre lo que mucho se ha escrito.

Este encaminamiento es de los más comunes e implementados en SDN y no se profundiza en todos los aspectos, sino que se da una visión general de su implementación y funcionamiento.

El Routing multicamino es una técnica que explota los recursos de la red mediante la propagación del tráfico desde un nodo de origen a un nodo de destino por medio de múltiples rutas a lo largo de la red.

Esta técnica es empleada en un gran número de propósitos, incluyendo la agregación de ancho de banda, la minimización del retardo de extremo a extremo, el aumento de la tolerancia a fallos, la mejora de la fiabilidad y el balanceo de carga, entre otras.

Existen tres elementos fundamentales en el routing multicamino: El descubrimiento de rutas (Path discovery), la distribución del tráfico y el mantenimiento de rutas, como se ve en [30].

Como resultado del estado del arte se pone de relevancia que la aparición de SDN contribuye a resolver la entropía que se genera al combinar protocolos de enrutamiento interno, externos, ingeniería de tráfico, balanceo de carga y ruteo multicamino, como ocurre en las redes tradicionales. Así, se abandona la idea del ruteo por destino buscando la mejor ruta y se reemplaza por la búsqueda de una distribución de flujos balanceada, permitiendo una mejor utilización de los recursos la red y de todos los enlaces disponibles.

Los esfuerzos por lograr lo expresado en el párrafo anterior se pone de manifiesto a través de las investigaciones sobre ingeniería de tráfico en MPLS, que como ya comentamos resulta excesivamente compleja su implementación y mantenimiento, máxime si lo que se busca es el comportamiento dinámico de la red. Se puede inferir entonces, que cualquier solución para un nuevo paradigma inicialmente tiene que ser simple, es decir, se tendría que emular el comportamiento habitual de las redes tradicionales en uso de una forma sencilla, para luego, a partir de ello, poder evolucionar hacia redes inteligentes.

En este trabajo, uno de los fundamentos es la búsqueda de la solución más sencilla sobre redes SDN.

**El tema de enrutamiento SDN se ampliará en el Capítulo N°4**

# Capítulo 3: Redes definidas por software.

## Sobre SDN

Las empresas y los proveedores de servicios están rodeados por una serie de fuerzas en competencia. El crecimiento monumental en el contenido multimedia, la explosión de la computación en la nube, el impacto del aumento del uso de dispositivos móviles y las continuas presiones empresariales para reducir los costos, están convergiendo para causar estragos en los modelos de negocios tradicionales ya que mientras tanto, los ingresos se mantienen planos,

Para mantener el ritmo, muchos empresarios están recurriendo a la tecnología SDN para revolucionar el diseño y las operaciones de la red, ya que SDN permite una administración consistente de la red, que puede estar formada por partes de tecnología compleja, como veremos más adelante.

Hay cuatro áreas críticas en las que la tecnología SDN puede marcar la diferencia para una organización:

1) Programabilidad de la red: SDN permite que el comportamiento de la red sea controlado por el software que reside más allá de los dispositivos de red que proporcionan conectividad física. Como resultado, los operadores de red pueden adaptar el comportamiento de sus redes para soportar nuevos servicios, e incluso clientes individuales. Al desacoplar el hardware del software, los operadores pueden introducir nuevos servicios innovadores y diferenciados rápidamente, sin las limitaciones de las plataformas cerradas y patentadas.

2) Centralización y control: SDN se basa en topologías de red centralizadas lógicamente, que permiten el control inteligente y la gestión de los recursos de la red. Los dispositivos funcionan de forma autónoma con un conocimiento limitado del estado de la red. Con el tipo de control centralizado que proporciona una red basada en SDN,

la administración del ancho de banda, la restauración, la calidad de servicio y las políticas pueden ser altamente inteligentes y optimizadas, y una organización obtiene una visión integral de la red.

3) Abstracción de la red: los servicios y aplicaciones que se ejecutan en tecnología SDN se extraen de las tecnologías y el hardware subyacentes que proporcionan conectividad física desde el control de la red. Las aplicaciones interactuarán con la red a través de API's (Aplication Programming Interface), en lugar de interfaces de administración estrechamente acopladas al hardware.

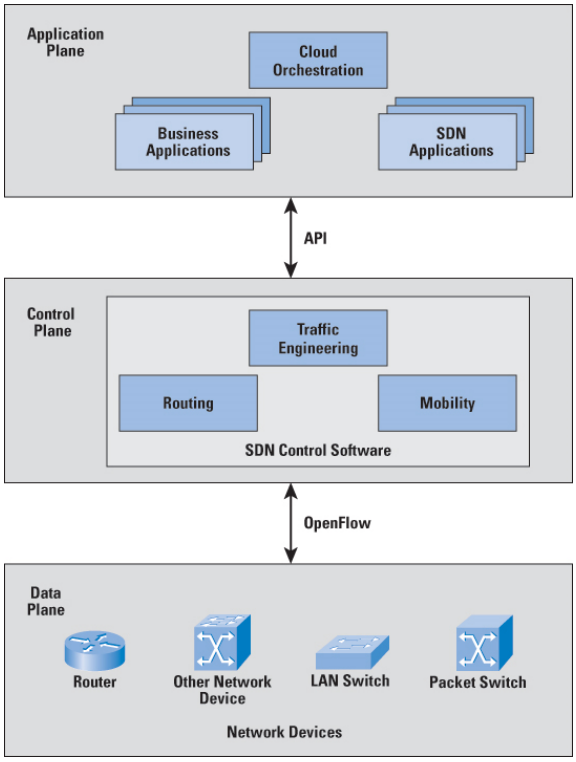
4) Apertura: las arquitecturas de SDN marcan el comienzo de una nueva era de apertura: habilitan la interoperabilidad de múltiples proveedores y fomentan un ecosistema neutral. La apertura viene del enfoque SDN en sí. Las API abiertas admiten una amplia gama de aplicaciones, incluidas la organización de la nube, OSS / BSS, SaaS, entre otros. Definamos estos conceptos:

a) Los Sistemas de Soporte de Operaciones (OSS) se refieren principalmente a los sistemas de red que están directamente vinculados con la operación de la misma, por ejemplo, configuración de los elementos, detección temprana de fallas, mantenimiento, etcétera. Básicamente, es lo que permite a los operadores de telecomunicaciones mantener el servicio móvil en funcionamiento.

El Sistema de Soporte de Negocios (BSS), por su parte, es el elemento complementario y se encarga de la administración de los elementos del negocio para los operadores de telecomunicaciones. Estos incluyen herramientas para atención al cliente, cobro, facturación, entre otros.[31]

b) El software como servicio (SaaS) permite a los usuarios conectarse a aplicaciones basadas en la nube a través de Internet y usarlas [32] SaaS ofrece una solución de software integral que se adquiere de un proveedor de servicios en la nube mediante un modelo de pago por uso. Toda la infraestructura subyacente, el middleware, el software y los datos de las aplicaciones se encuentran en el centro de datos del proveedor. El proveedor de servicios administra el hardware y el software y, con el contrato de servicio adecuado, garantizará también la disponibilidad y la seguridad de la aplicación y de los datos.

Figura 3-1 – Estructura lógica de una red SDN [33]




A esto debemos agregarle las aplicaciones de red críticas para cada empresa u organización que, por supuesto, dependerán de cada entorno.

Además, el software inteligente puede controlar el hardware de múltiples proveedores con interfaces programáticas abiertas como OpenFlow.

Para convertir el concepto de SDN en implementación práctica, se deben cumplir dos requisitos. Primero, debe haber una arquitectura lógica común en todos los conmutadores, enrutadores y otros dispositivos de red que deben ser gestionados por un controlador SDN. Esta arquitectura lógica puede implementarse de diferentes maneras, siempre que el controlador SDN vea una función de conmutador lógico uniforme, en diferentes equipos de proveedores y en diferentes tipos de dispositivos de red. En segundo lugar, se necesita un protocolo seguro y estándar entre el controlador SDN y el dispositivo de red.

## OpenFlow

OpenFlow aborda estos dos requisitos, ya que es un protocolo entre los controladores SDN y los dispositivos de red, así como una especificación de la estructura lógica de las funciones del conmutador de red. OpenFlow se define en OpenFlow Switch Specification, publicado por Open Networking Foundation (ONF). ONF es un consorcio de proveedores de software, redes de distribución de contenido y proveedores de equipos de redes cuyo objetivo es promover las redes definidas por software.

Este trabajo se basa en una especificación de OpenFlow, versión 1.3., del 25 de junio de 2012. La especificación original, 1.0, se desarrolló en la Universidad de Stanford y se implementó ampliamente. OpenFlow 1.2 fue el primer lanzamiento de ONF después de heredar el proyecto de Stanford. OpenFlow 1.3 expande significativamente las funciones de la especificación. Es probable que la versión 1.3 se convierta en la base estable sobre la cual se construirán futuras implementaciones comerciales para OpenFlow.[34]. Sin embargo, empresas como Noviflow han integrado en sus productos como NoviWare 400.0, OpenFlow 1.5 en su línea NoviSwitch de conmutadores y enrutadores SDN de alto rendimiento. NoviWare 400 incluye una gran cantidad de mejoras, incluida la negociación de versiones OpenFlow 1.3, 1.4 y 1.5 y la acción OpenFlow 1.5 Copy-Fields. NoviFlow también ha implementado Copy-Fields como una acción de Experimentador para usar con cualquier controlador OpenFlow 1.3, 1.4 y 1.5 compatible con Experimentador estándar y / o aplicación SDN.[35]

De acuerdo a su documentación, Openflow es un protocolo que ha ido evolucionando con el tiempo y que ha sufrido algunos cambios drásticos entre versiones. Hasta fines del 2018 OpenFlow se encontraba en la versión 1.5, pero la versión 1.3 es la que se tomará como referencia para explicar las características principales del protocolo. Aunque, como se verá más adelante, el hecho de que no sea la versión más reciente no significa que los dispositivos utilizados no trabajen con ella.. Sin embargo, es presumible que en el futuro las características de las versiones más modernas de OpenFlow acaben por ser implementadas en la mayoría de los switches, por lo que conviene conocerlas.

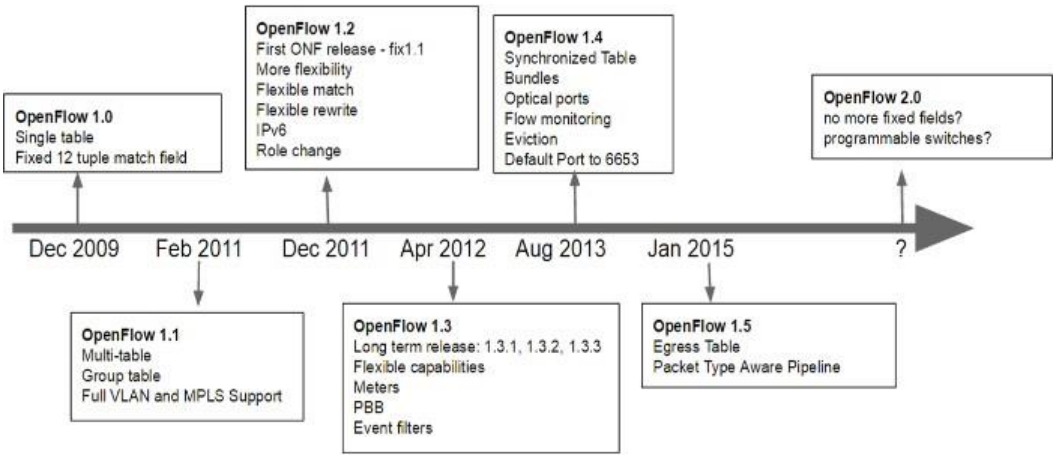



Figura 3-2 - Historial de versiones OpenFlow [36]

## Estandarizaciones en SDN

En este apartado se mencionarán las necesidades de estandarización sobre redes SDN. Inicialmente y debido al gran éxito de las tecnologías de la comunicación, se han realizado diversas aplicaciones en redes, y los requisitos que traen son divergentes. Para apoyar estos divergentes requisitos, es necesario hacer que las redes sean aún más controlables y manejables. La necesidad de tratar el tráfico diferente de diferentes maneras, conlleva un aumento a la orientación al servicio. Numerosas tecnologías que permiten métodos de control de datos más directos y de menor nivel (por ejemplo, nivel de flujo) están surgiendo para el reenvío de paquetes. Estas tec-

nologías pueden simplificar la reacción de los recursos de la red (por ejemplo, conmutadores o enrutadores) y aumentar significativamente la capacidad de control de las mencionadas redes para los operadores a cargo. Con el modelado y la programación de los recursos de red, las redes se pueden centralizar de manera automatizada, lo que permite un funcionamiento más ágil de las mismas.

Este cambio de método de control también puede brindar la oportunidad de rediseñar las funciones de control e introducir un control lógicamente centralizado y programable de los recursos de la red a través de interfaces y protocolos estandarizados.

Por lo tanto, podemos decir que este enfoque permite:

- control de red lógicamente centralizado, que disminuye el número de puntos a controlar y gestionar;

- apoyar la virtualización de la red como una característica importante de la arquitectura de la red;

- definir, controlar y gestionar los recursos de la red utilizando software; permitiendo así servicios de red que se proporcionarán de manera determinista de acuerdo con la solicitud de comportamiento; y la personalización de la red, que es necesaria para un despliegue de red eficiente y eficaz en sus operaciones.

Para realizar las características antes mencionadas, la recomendación UIT-T Y.3300 proporciona el marco de redes definidas por software (SDN) especificando los fundamentos de SDN con sus definiciones, objetivos, capacidades de alto nivel, requisitos y arquitectura de alto nivel.

Si bien se han realizado varios esfuerzos para desarrollar tecnologías y estándares relacionados con SDN como [UIT-T Y.3001], [UIT-T Y.3011], [b-UIT-T Y.2622], [b-ETSI NFV], [b-IETF I2RS], [b-IETF RFC 3746], [b-ONF] y [b-OpenDayLight]) con diferentes enfoques, todos comparten el mismo objetivo de proporcionar la programabilidad de los recursos de red como se ha descrito anteriormente, que es una tecnología central para las redes del futuro.

## Tablas de flujos

Como ya se ha comentado previamente, los flujos se agrupan en tablas similares a las de rutas o las de conmutación. Cuando se recibe un paquete, se analiza la tabla elemento a elemento para comprobar si se cumplen los requisitos de matching en alguna de ellas. En la versión 1.0 de Openflow, únicamente existía una tabla de flujos. Sin embargo, a partir de OpenFlow 1.1 se soportan tablas anidadas. OpenFlow sigue un proceso general perfectamente definido a la hora de recorrer las tablas de flujos en el switch. Este proceso, o pipeline, es el siguiente:

Las tablas de flujos en el switch se ordenan por números, empezando en 0. La tabla 0 debe existir siempre, ya que debe haber al menos una tabla de flujos en el switch. El proceso empieza siempre en esta primera tabla. Cuando entra un paquete se intenta asociar con alguna entrada de la tabla 0. En caso de encontrar una coincidencia, se añaden las instrucciones asociadas a ese flujo al denominado "action set" del paquete. El action set es el conjunto de acciones que se aplican al paquete entrante una vez que han acabado de recorrerse las tablas de flujos. Si entre las instrucciones de la entrada se encuentra la de pasar a otra tabla (Instrucción go to), se avanza a esa tabla y se repite el mismo proceso. Nótese que siempre debe avanzarse a tablas de numeración más alta, nunca más baja. Dicho de otra forma, siempre debe avanzarse hacia delante, y no puede volverse nunca a tablas anteriores. Cuando no quedan más tablas que recorrer, es decir, cuando el "match" de una tabla no incluye la instrucción "go to" indicando que debe avanzarse a otra tabla, se ejecuta el action set asociado al paquete. El action set se ejecutará en un orden preestablecido.

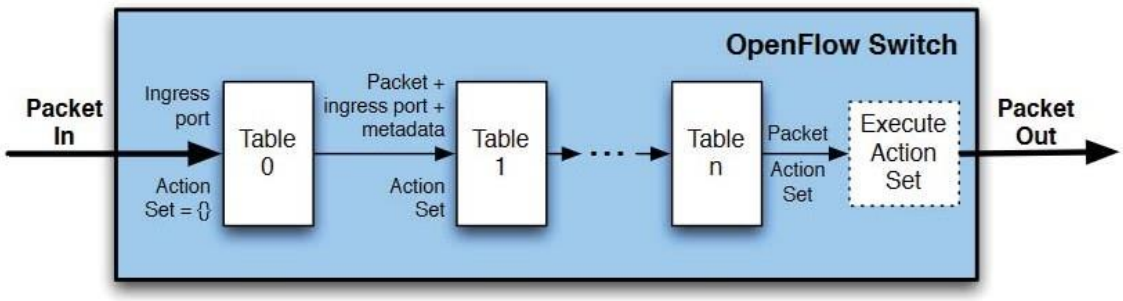



Figura 3-3 - Pipeline de OpenFlow 1.3.

En caso de que no exista una entrada asociada al paquete entrante en una tabla, se produce un fallo de tabla ("table miss"). En este caso, el comportamiento dependerá de cómo esté configurada la tabla. Se pueden añadir entradas en las tablas específicamente para indicar cómo procesar paquetes no equiparados (unmatched packets).

Las opciones son:

1. Enviar el paquete al controlador para que este se encargue de encaminarlo.
2. Pasar el paquete a otra tabla del switch.
3. Desechar el paquete.

Nótese que el orden de las entradas de los flujos influye en su prioridad, de forma similar a las reglas de un firewall o las listas de acceso de un router. Esto implica que podría darse el caso de que existan dos entradas en la tabla flujos que pudieran estar asociadas a un paquete entrante, sin embargo, al recorrerse en orden descendente la tabla, solo se ejecutarán las instrucciones asociadas a la primera entrada que tenga coincidencia.

En OpenFlow 1.5 se modifica el pipeline. La parte vista hasta ahora pasa a denominarse Ingress Processing (procesamiento de entrada) y se incluye el Egress Processing (procesamiento de salida). Fundamentalmente, el Egress Processing sigue el mismo funcionamiento que el Ingress Processing, se recorren las tablas buscando equiparaciones válidas para el paquete y se ejecutan las instrucciones de la tabla asociadas a la entrada. Su función es permitir mayor granularidad y organización en las entradas de las tablas de flujos. Se trata de un procesamiento opcional y por tanto no es necesario que el switch lo implemente para poder encaminar tráfico correctamente.

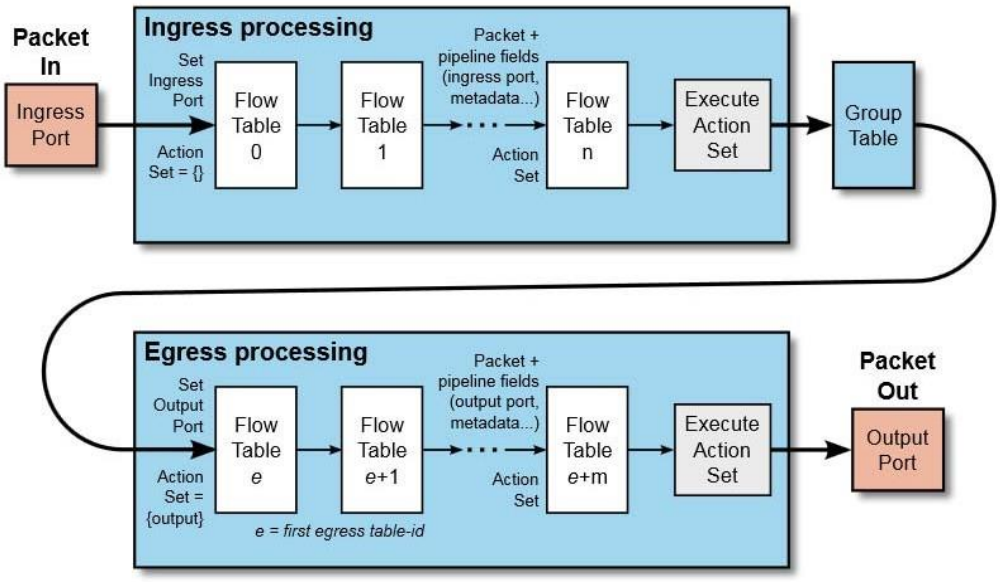



Figura 3-4: Pipeline de OpenFlow 1.5.

## Instrucciones y action sets

Como ya se ha establecido, cada paquete entrante tiene asociado un action set y cada entrada en las tablas de flujos contiene un conjunto de instrucciones. Es importante entender esta distinción. El action set asociado al paquete entrante contiene las acciones que se ejecutarán sobre el paquete al acabar de recorrer las tablas. Las instrucciones asociadas a la entrada de un flujo en la tabla de flujos se ejecutan cuando un paquete entrante equipara en esa entrada.

Primero se hablará sobre las instrucciones. Existen tres tipos de ellas:

1. Instrucciones que hacen avanzar el pipeline (esto es, que avanzan a otra tabla).
2. Instrucciones que modifican el action set del paquete.
3. Instrucciones que modifican el paquete entrante inmediatamente sin esperar a que se ejecute el action set al final del pipeline (de carácter opcional).

En total, existen seis instrucciones. Dos son obligatorias y cuatro son opcionales. Esto significa que el switch debe necesariamente tener soporte para las obligatorias. Respecto a las opcionales, el controlador puede consultar a los switches cuáles de las

instrucciones opcionales están soportadas por cada switch. Se destacan a continuación, las instrucciones obligatorias, que son las siguientes:

- Write-actions <acción/es>: Inserta una acción o conjunto de acciones en el action set del paquete entrante. Si alguna de las acciones que se pretenden insertar ya está en el action set, se sobrescribe.

- Goto-table <ID de la tabla>: Indica cuál es la siguiente tabla que debe consultarse.

Las instrucciones opcionales, se indican a continuación:

- Meter <id métrica>: Aplica una limitación a la métrica especificada.
- Apply-Actions <acciones>: Se aplica la acción especificada sobre el paquete inmediatamente, sin esperar a que se ejecute el action set.
- Clear-Actions <acciones>: Elimina todas las acciones contenidas en el action set.
- Write-Metadata <metadatos/mascara>: Actualiza los metadatos.

Cada entrada en la tabla de flujos sólo puede tener una instrucción de cada tipo asociada y se ejecutan en el orden en el que están definidas en dicha entrada. Esto, en la práctica, y aplicado a las instrucciones obligatorias, implica que Goto-table siempre va después de Write- actions, ya que de lo contrario se avanzaría de tabla antes de escribir las acciones en el action set del paquete entrante.

Los action sets, por otro lado, son el conjunto de acciones que se ejecutan sobre el paquete al acabar el pipeline. Aunque las acciones se añaden al action set a medida que se van recorriendo las sucesivas tablas de flujos, estas no se ejecutan en orden cronológico de llegada. A continuación, se definen las acciones en orden de prioridad de ejecución:

- Copy TTL inwards: Copiar el campo TTL de la cabecera IP más externa del paquete entrante a la segunda más externa con campo TTL. Esta acción se ejecuta normalmente cuando existe una cabecera IP anidada dentro de una cabecera MPLS (que tiene TTL).
- Pop: elimina todas las etiquetas VLAN, PBB y MPLS del paquete.
- Push-MPLS: añade una etiqueta MPLS al paquete.
- Push-PBB: añade una etiqueta PBB al paquete.
- Push-VLAN: añade una etiqueta VLAN al paquete.
- Copy TTL outwards: Copiar el campo TTL de la segunda cabecera IP más externa del paquete entrante con campo TTL a la cabecera IP más externa.
- Decrementar TTL de la cabecera IP.
- Set: se usa junto a las instrucciones opcionales.
- Qos: aplica acciones de QoS
- Group: se usa junto a las instrucciones opcionales. En el presente trabajo no se profundizará en su funcionamiento.
- Output: envía el paquete por el Puerto especificado

Solo con el objeto de aumentar la comprensión de su funcionamiento, se plantea ,a continuación, un ejemplo simple de aplicación indicando como se establecen las rutas en un conmutador y posteriormente como se realiza la interacción con el controlador.

Considérese una topología consistente en una red que posee dos conmutadores (SW) identificados como S1 y S2, a los cuales se conectan 2 host identificados como h1 y h2 conectados a cada SW respectivamente, como se indica en la figura 3- 5

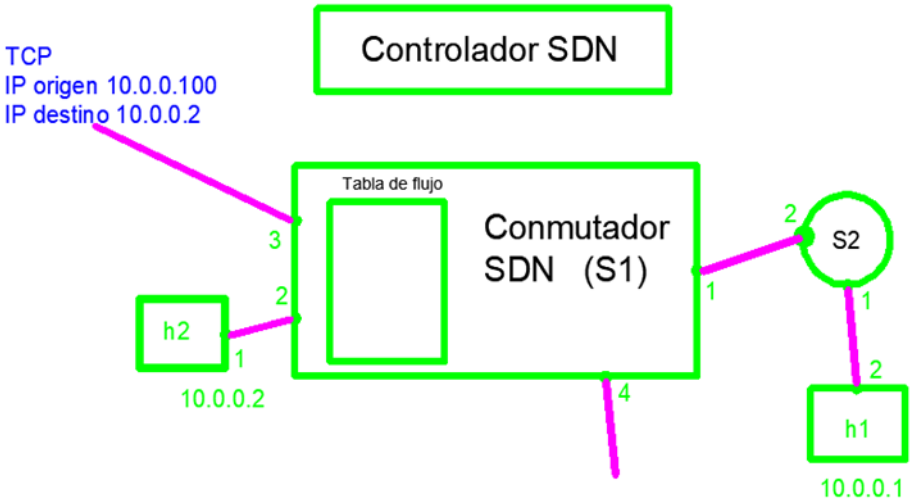



Figura 3-5- Funcionamiento del conmutador – Fuente propia

Se analizará el comportamiento del conmutador S1, el cual se conecta con otros dispositivos de la red mediante los puertos 1, 3 y 4, y también con el host mediante el puerto 2.

Dentro del conmutador S1 se encuentra una tabla de flujo que se utiliza para definir las acciones que se aplicarán a los paquetes que ingresan al SW, acorde con los criterios de coincidencia para cada entrada.

Los criterios se pueden definir tomando como base campos del encabezado Ethernet, IP, TCP, UDP o cualquier otro protocolo que contenga el encabezado del paquete.

En el presente caso se utilizará como criterio de coincidencia las direcciones IP de origen y destino. Aquellos paquetes que coincidan con los criterios deberán cumplir con las acciones indicadas en la tabla, que podrán ser enviarlo a un puerto del S1, enviarlo al controlador para que modifique los campos sin completar en el paquete u otras acciones.

Se agregarán criterios y acciones que permitan que los paquetes que lleguen al puerto 3 o al 2 se dirijan al host h1 cuya dirección IP es 10.0.0.1. Para ello deberán salir a través del puerto 1, atravesar S2 y posteriormente llegar a h1. También interesa que los paquetes que lleguen con dirección de destino 10.0.0.2 sean despachados por el puerto 2 con destino al host h2.

En estos casos no se establece ninguna condición que refleje la dirección de origen, ni el protocolo que transporta.

Se puede agregar una tercer entrada de forma que los paquetes que ingresen a S1 y cuyo protocolo de transporte sea TCP, sean despachados por el puerto 4, sin poner interés en sus direcciones IP de origen o destino. Se obtiene una tabla como la que se muestra a continuación:

| TABLA DE FLUJO | | | | | | |
|---|---|---|---|---|---|---|
| Criterio de coincidencia | | | Acciones | Prioridad | Tiempo de duración de la regla | Tiempo de espera inactivo |
| IP origen | IP destino | Protocolo | | | | |
| * | 10.0.0.1 | * | out: 1 | 2 | 20 seg | 5 seg |
| * | 10.0.0.2 | * | out:2 | 2 | 20 seg | 5 seg |
| * | * | TCP | out: 4 | 1 | 0 | 0 seg |

Si se concentra la atención en los primeros campos de la tabla de flujo se observará que cualquier paquete que no transporte información del protocolo de TCP y se dirija hacia los host 1 o 2 será enrutado convenientemente a través de los puertos 1 y 2 respectivamente.

Cuando en este escenario ingresa un paquete con dirección IP de origen 10.0.0.100 y de destino 10.0.0.2 que transporta un mensaje TCP, se observa que hay dos entradas de coincidencia y por lo tanto el S1 no estará en condiciones de decidir si entregarlo al puerto 2 o al 4. Para resolver este problema se incorpora el campo de prioridad indicado en el esquema, donde se adopta aquella entrada con mayor número, entregando entonces el paquete en el puerto 2.

Por último, se observa la presencia de dos campos de coincidencia relacionados con tiempos. El primero indica el tiempo de duración de la regla desde el momento que el controlador la instaló, es decir, la permanencia en la tabla. El segundo indica que la regla será removida si en ese periodo no se efectúa ninguna coincidencia, es decir no llega ningún paquete que cumpla con la misma.

Colocar 0 en el campo “Tiempo de duración de la regla” establece la condición que la regla solo podrá ser removida por el controlador. De la misma forma, colocar 0 en el campo “Tiempo de espera inactivo” también significará que la regla solo podrá ser removida por el controlador.

El controlador tendrá el control del SW, mediante una conexión TCP, que se establece al momento de instalarlo.

El controlador tendrá instaladas aplicaciones que reconocerán ciertos eventos y reaccionarán apropiadamente. Se exponen ejemplos a continuación.

Si se supone que en la topología anterior se agrega un nuevo conmutador, se establecerá una conexión TCP entre controlador y conmutador. Luego, intercambiando desde cada lado de la conexión un mensaje de OFPT_HELLO [37][38] que son mensajes simétricos enviados por ambos, switch y controlador, se establece la conexión OpenFlow, modificando la topología conocida por el controlador como se verá más adelante.

El evento producido, claramente generará cambios en el conmutador S1. El controlador debe modificar la tabla de flujos de S1, mediante el uso de los mensajes OFPFlowMod, instalando la entrada en particular, indicando un criterio de coincidencia, una prioridad de acción y sus tiempos asociados.

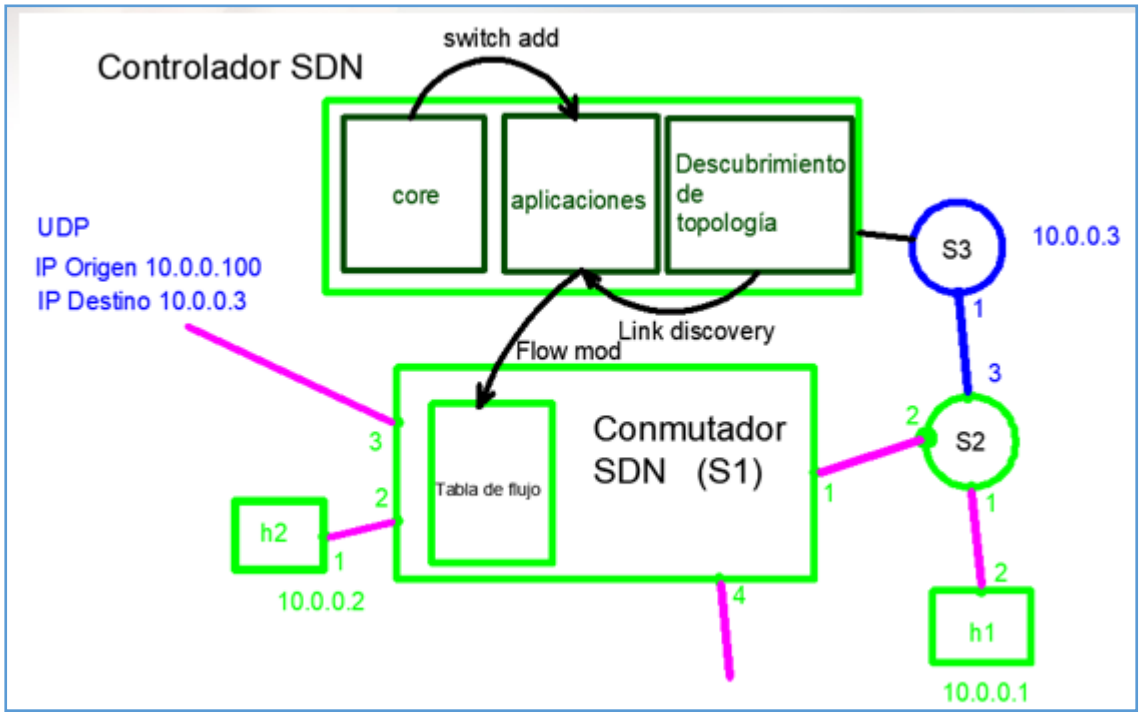



Figura 3-6- Evento- incorporación de un Nodo : Fuente propia

La figura 3-6 muestra un detalle del evento producido al incorporar el nodo S3.

En el caso, que el paquete no coincida con ninguna entrada de la tabla, o cuando específicamente se establezca como acción, el paquete o su encabezado será enviado

al controlador, mediante mensaje OFPT_PACKET_IN. El módulo central del controlador dispondrá de alguna aplicación que esté dedicada a resolver el envío de estos paquetes. En ese caso la aplicación determinará las reglas de flujo que serán instaladas en el S1 usando los mensajes OFPFlowMod. (figura 3-7). Si no existiese dicha aplicación, se eliminará el paquete:

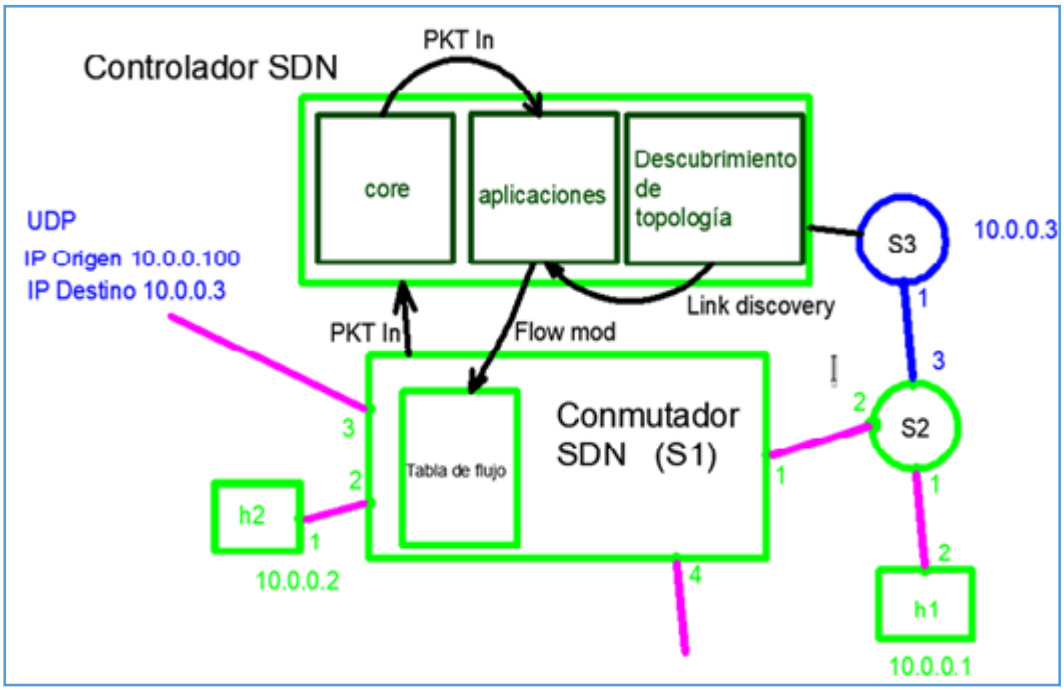



Figura 3-7- Evento- Paquete no coincide con ninguna entrada de la tabla: Fuente propia

Por último, en el caso que se produzca el descubrimiento de un enlace, el controlador dispone de un módulo de descubrimiento topológico que utiliza los paquetes de control intercambiados cuando la red SDN cambia su topología.

Cuando el controlador recibe uno de estos paquetes de descubrimiento de un SW, indicando que modificará algún enlace, genera un evento de descubrimiento y lo envía a alguna aplicación dedicada a ese evento en particular. Este evento podría ser la incorporación del S3, en el ejemplo anterior. La aplicación debe ser capaz de responder generando los mensajes OFPFlowMod para agregar o quitar entradas como se indicó anteriormente.

## Para finalizar

Finalmente, desde dentro de la SDN, los servicios y aplicaciones de red inteligentes pueden ejecutarse dentro de un entorno de software común.

Una ventaja clave de la tecnología SDN es la capacidad de los operadores de red para escribir programas que utilizan las API SDN y dan a las aplicaciones control sobre el comportamiento de la red. SDN permite a los usuarios desarrollar aplicaciones compatibles con la red, monitorear de manera inteligente las condiciones de la misma y adaptar automáticamente la configuración de la red según sea necesario. [39]

En definitiva, la red definida por software (SDN) es un enfoque de la virtualización de red que busca optimizar los recursos de la red y adaptar rápidamente las redes a las cambiantes necesidades comerciales, las aplicaciones y el tráfico. Funciona ágilmente al separar el plano de control de la red y el plano de datos, creando una infraestructura programable por software que es distinta de los dispositivos físicos. Con SDN, las funciones de orquestación de red, administración, análisis y automatización se convierten en el trabajo de los controladores SDN. Dado que estos controladores no son dispositivos de red, pueden aprovechar la escala, el rendimiento y la disponibilidad de los modernos recursos de almacenamiento y computación en la nube. Cada vez más, los controladores SDN se construyen en plataformas abiertas, utilizando estándares abiertos y API abiertas, lo que les permite organizar, administrar y controlar equipos de red de diferentes proveedores. SDN ofrece una amplia gama de beneficios comerciales. La separación de las capas de control y transporte aumenta la flexibilidad y acelera el tiempo de comercialización para nuevas aplicaciones. La capacidad de responder más rápidamente a los problemas e interrupciones mejora la disponibilidad de la red. Y la capacidad de programación facilita a las organizaciones de Tecnologías Informáticas la automatización de las funciones de red, reduciendo los costos operativos.

SDN encaja con otra tecnología, la virtualización de funciones de red (NFV). NFV ofrece la posibilidad de virtualizar funciones de red basadas en dispositivos, como firewalls, balanceadores de carga y aceleradores de WAN. El control centralizado que proporciona SDN puede administrar y orquestar de manera eficiente las funciones de red virtual habilitadas por NFV.[40]

# Capítulo 4: Enrutamiento y Calidad de Servicio

La función principal de una red consiste en transmitir información de un punto a otro. Para el logro de este objetivo, se emplean diversas rutas en la red. Su desempeño se define por las diversas condiciones en las trayectorias como la capacidad de tráfico y los retardos entre nodos, entre otros. Estos factores son considerados indispensables en el proceso de enrutamiento orientado a la calidad de servicio (QoS) y tiene como objetivo garantizar el rendimiento de las aplicaciones que requieren de un trato preferencial para su desempeño. Este objetivo suele cumplirse en las redes convencionales, pero con alta complejidad y asignaciones de recursos en el proceso de enrutamiento, convirtiéndose en uno de los principales desafíos para el buen desempeño de la red en su conjunto. Una solución a estos problemas es la implementación de redes definidas por software (SDN). Las mismas, tienen la capacidad de monitorear los requerimientos de las aplicaciones y los recursos disponibles, antes de la realización del proceso de asignación de rutas en la red por cada flujo de datos. Es posible, entonces, abstraer los parámetros de la red SDN y emplearlos en conjunto para proporcionar rutas favorables con QoS empleando procesos en el tratamiento de los flujos que impliquen sencillez y centralización en la organización de los mismos. También es posible el posterior monitoreo de los parámetros asociados al comportamiento esperado de la red.

## 4.1.Sobre los protocolos de enrutamiento en redes tradicionales

La arquitectura actual de redes distribuidas posee numerosos detalles de importancia. Por una parte, deben existir mecanismos que permitan a los enrutadores de tráfico o routers, mantener información actualizada y confiable sobre los caminos óptimos por los que deben enviar los paquetes. Por otra parte, deben ser capaces de responder ante cambios o fallos en la topología de la red. Para conseguir este objetivo dentro de una red en la que cada nodo toma sus decisiones de forma independiente, se hace necesaria la existencia de protocolos de enrutamiento (routing) dinámico [41].

En las redes tradicionales existen diferentes protocolos de enrutamiento dinámico, tanto para enrutar dentro de un mismo sistema autónomo (Interior Gateway Protocol, IGP) como para enrutar entre diferentes sistemas autónomos (Exterior Gateway

Protocol, EGP). Cada protocolo puede usar un algoritmo diferente para determinar el camino óptimo, pero todos comparten dos propiedades comunes. Primero, necesitan comunicarse con otros dispositivos para completar sus tablas de rutas. Segundo, necesitan informarse los unos a los otros regularmente de que siguen operativos y asegurarse de que mantienen información de enrutamiento actualizada. Existen diferencias de funcionamiento entre los diferentes protocolos. En el caso de RIP[42] (Routing Information Protocol), utiliza paquetes de datos UDP de difusión para intercambiar información de enrutamiento. Se envían actualizaciones de información de enrutamiento cada 30 segundos, lo que se denomina publicación. Si un dispositivo no recibe una actualización de otro dispositivo durante 180 segundos o más, el dispositivo receptor marca las rutas servidas por el dispositivo no actualizado como inutilizables. Si aún no hay actualización después de 240 segundos, el dispositivo elimina todas las entradas de la tabla de enrutamiento para el dispositivo que no se actualiza. En el caso de OSPF (Open Shortest Path First), todos los routers intercambian información entre sí a través de mensajes de diferente tipo y todos ellos son conscientes de la topología completa de la red.

Otro esfuerzo se orientó a generar modelos de redes donde el ruteo de realiza mediante etiquetas, que posibilitan la creación de verdaderos circuitos virtuales reduciendo así el procesamiento en los nodos internos de la red pero incrementando el trabajo de los nodos de borde o de acceso a dicha red, quizás una idea tomada de ATM, este dispositivo de acceso es el encargado de colocar las etiquetas y establecer la ruta que conducirá a la trama hasta el nodo de egreso de la red.

Este notable cambio en el enfoque se observa con el protocolo MPLS (Multiprotocol Label Switching). Realmente es un conjunto de protocolos que se formó sobre la base de combinar las mejores características del reenvío de la capa 2 (o conmutación) con las mejores partes de la capa 3 de enrutamiento IP para formar una tecnología que combina el envío de paquetes extremadamente rápido [43] con las técnicas de señalización de trayectoria muy flexibles y complejas adoptadas desde el mundo IP.

IP y MPLS son ejemplos de un modelo de reenvío con control distribuido. En estos paradigmas las rutas y la información de accesibilidad se intercambian en trayectos de plano de datos que están programados para establecer las rutas de origen a destino.

El protocolo EVPN [44] (La red privada virtual Ethernet) es otro intento para resolver los problemas de escalabilidad de las redes de la capa 2 que se describe mediante la interconexión de puentes de capa 2 distantes sobre una infraestructura MPLS (o GRE), luego de ello se intercambia la información de direccionamiento y accesibilidad de capa 2 a través de estos túneles, no alterando la escalabilidad de las redes de la capa 3.

Hasta ahora, los protocolos del plano de control han sido fundamentales para el correcto funcionamiento de las redes en todos los ámbitos.

Pero en los últimos tiempos se plantean nuevos retos al paradigma tradicional. Por un lado, la entrada en escena de la virtualización, cada vez más relevante, ha trastocado la forma de concebir las redes. Por otro lado, el crecimiento de las infraestructuras de red lleva a plantearse la optimización del modelo actual de control en términos de rendimiento y consumo de ancho de banda. A mayor número de routers conectados, mayor consumo de ancho de banda en la red dedicado a protocolos de enrutamiento y mayor consumo de recursos dentro de cada router para mantener actualizada la topología, en caso de redes tradicionales.

Además, para los casos de uso de IoT (Internet of Things – Internet de las cosas), existen literalmente cientos de diferentes tipos de dispositivos y sensores para el armado de redes propias.  Cada uno tiene sus requisitos únicos, que incluyen la cantidad de conexiones, el costo por conexión, la disponibilidad de energía y la cantidad de transferencia de datos requerida, tanto en sentido ascendente como descendente.

Dependiendo de la aplicación, las redes de dispositivos IoT requerirán conectividad escalable, fiable y segura para dispositivos y sensores remotos. Tal vez el mayor desafío sea proporcionar conexiones de bajo costo a dispositivos remotos, algunos de los cuales usarán baterías y no tendrán suministro de alimentación.

Así, para estos casos, se utilizan protocolos de ruteo o enrutamiento totalmente diferentes y muchos de ellos adaptables a usos específicos de este tipo de redes, que dependerán de la calidad de servicio necesaria[45].

## 4.2.Sobre el proceso de reenvío y la separación de los planos de control y datos en redes tradicionales

El proceso de reenvío fue variando desde finales de los años 80 ó principios de los 90, donde no había routers enormes con gran cantidad de recursos, ni operaciones aceleradas por hardware. Todo era un asunto de usar el poder de procesamiento apropiadamente con el objeto de ahorrar ciclos de CPU para momentos difíciles.

Las tablas de enrutamiento de los routers eran el componente esencial de proceso de forwarding, esa época las métricas utilizadas estaban íntimamente relacionadas con los recursos puestos en juego a la hora de enviar los datos, recursos que mostraban las condiciones iniciales de los caminos recorridos.

Los recursos que se intentaban preservar eran la capacidad de procesamiento en los nodos, eligiendo como métrica el número de saltos, o el uso de los mejores enlaces, a través de métricas que involucren el ancho de banda del mismo, como en el caso de OSPF donde el costo de las rutas resulta al sumar el costo de cada enlace

(Ci = 10^8/ancho de banda (bps)), seleccionando aquel que minimice el resultado. Las mejores rutas a cada destino conformabanla tabla de enrutamiento.

Por una cuestión de simplicidad consideremos solamente el ruteo interno con ello evitaremos complicar la explicación.

Cuando un paquete llegaba al router se seguían los pasos indicados en la figura para lograr el reenvío. Obsérvese la figura 4-1 y los pasos que se detallan posteriormente.


Figura 4-1 - Proceso reenvío de un router antiguo [45]

1. El router recibe una trama, y para asegurarse de que puede trabajar con ella, revisa el *frame check sequence* de la trama que recibió. Si algún error es hallado, la trama es descartada.

2. Si el chequeo de FCS es positivo, el *router* entonces revisa el campo *Ethernet type* para saber qué tipo de paquete es y extrae el paquete.

3. Una vez que el paquete ha sido extraído, la siguiente acción depende de la versión del paquete: si es IPv4, el *header checksum* es verificado; un paquete con un *checksum* erróneo es descartado. Si el paquete es IPv6, la revisión no se realiza debido a que el *header* de IPv6 no contiene un *checksum*.

4. El router revisa si la dirección de destino del paquete coincide con alguna de las direcciones IP configuradas en sus *interfaces* que se encuentren en el estado *"up, line protocol up"*. Si hay alguna coincidencia, entonces el *router* es el destino del paquete y este es procesado de manera local.

5. Si la dirección de destino no coincide con ninguna de las interfaces del *router* en modo up, entonces el proceso de enrutamiento comienza. Pero para que eso su-

ceda, el TTL del paquete debe ser mayor que 1. Si lo es, entonces puede ser enrutado; si campo TTL no contiene un valor mayor que 1, entonces es descartado y su remitente notificado con un mensaje *ICMP Time Exceeded*.

6. El *router* revisa su tabla de enrutamiento y busca una red -o subred- que coincida de la manera más cercana posible con la dirección de destino del paquete. Esto es llamado "*Longest prefix match*" en inglés, refiriéndose a una coincidencia con la mayor cantidad de bits posible entre la dirección de destino del paquete y la que se encuentre en la tabla de enrutamiento.

7. Una vez que el siguiente salto y la interfaz de salida han sido encontrados en la tabla de enrutamiento, el router necesita la información *data link* del siguiente salto para poder construir una trama nueva y así entregar el paquete. El mismo se apoya en mecanismos como ARP, para relacionar información de capa 3 con información de capa 2.

8. Como el paquete será enrutado, el TTL es reducido por una unidad, y como consecuencia, el contenido del *header* IPv4 ha cambiado (TTL). por lo tanto, el *checksum* IPv4 es recalculado. En el caso de IPv6, se resta una unidad al valor "hop count" del *header* sin necesidad de recalcular *checksum*, porque el *header* no lo contiene.

9. Finalmente, el router encapsula el paquete dentro del *header* y *trailer* de *data link* recién creados y enviar la trama [46].

El proceso de reenvío era ejecutado paquete por paquete por el CPU del router, desde el momento en el que el paquete era recibido por una interfaz hasta que era enviado. Es decir que se debían ejecutar el proceso de 9 pasos descrito anteriormente por cada paquete recibido, siendo implementado como un proceso individual corriendo en el sistema operativo del router.

Para encontrar el mejor camino a una red destino de un paquete, el router tenía que recorrer su tabla de enrutamiento, también llamada Routing Information Base (RIB),

y encontrar la entrada en la RIB que coincida con la dirección del paquete en una línea de bits más larga.

.

La tabla de enrutamiento no contenía la información para ejecutar el frame rewrite, reescritura de la trama de ese paquete. Solo luego de construir el nuevo header de la trama usando la tabla de ARP, el paquete podía ser enviado. Este proceso debía ser repetido por todos y cada uno de los paquetes, para luego construir un nuevo header para la trama.

Buscar la información necesaria para realizar las operaciones descriptas eran las tareas más demandantes de recursos de todo el proceso de reenvío, y tener que realizarla cada vez que un paquete era recibido era simplemente, ineficiente.
Cuando el tamaño y la velocidad de las redes se incrementaron fueron creciendo, los routers se volvieron cuellos de botella incapaces de adaptarse al crecimiento, resultaba imprescindible disminuir la latencia del enrutamiento y switching de las tramas.

El primer paso para solucionar el problema de la alta latencia dio origen al Fast Switching que permitía que los paquetes haciendo uso de un caché on-demand, donde el primer paquete de una solicitud era procesado y los resultados de sus búsquedas eran almacenados en caché, que luego era utilizados por los paquetes posteriores, reduciendo así el tiempo de cálculo y apuntando a optimizar el consumo de recursos.

La información almacenada en el caché incluiría la dirección IP de destino, información del siguiente salto e información del header (encabezado)de capa 2, la cual sería escrita en la trama antes de enviarla a través de la correspondiente interfaz de salida evitando el procesamiento de los próximos paquetes dirigidos hacia el mismo destino.

Si una gran cantidad de paquetes fuesen dirigidos a destinos que aún no estaban en el caché, el router debería procesarlos y su comportamiento seria el mismo que en el caso anterior, sin observarse mejoras. La situación se vería agravada cuando las entradas que estuviesen en el caché fuesen removidas por estar desactualizadas.

En 1996, un nuevo mecanismo fue creado para superar las limitaciones de Fast Switching y brindar mejoras a las redes, obteniendo más rendimiento de routers basados en software, e incluso hacer este proceso más simple de implementar en hardware, Para ello era necesario pensar íntegramente el proceso para enrutar paquetes.

Ya se ha mencionado sobre la búsqueda de las rutas en la RIB, La tabla de enrutamiento es un componente hecho para construir y almacenar información de redes, no está realmente optimizado para la búsqueda de una ruta. Para optimizar el procedimiento de búsqueda se debe cambiar su estructura lineal por algo más eficiente, más rápido, y que permita reducir la recursión. La estructura que satisface estos requerimientos es la base de información de reenvío (Forwarding Information Base (FIB)).

La FIB es una base de datos creada dinámicamente usando la información contenida en la tabla de enrutamiento donde los prefijos o direcciones de destino son copiados y sus siguientes saltos resueltos de forma recursiva. La meta es encontrar toda la información para reenviar el paquete en una sola búsqueda.

A diferencia de la RIB, cuyo propósito es construir y almacenar información sobre redes, la FIB está construida y ordenada en una manera que le permite optimizar la búsqueda rápida de información y el longest prefix match. Esta estructura es llamada trie, una estructura de datos similar a un árbol, lista para ser recorrida, compuesta de nodos y hojas, y se concentra en la obtención rápida de datos.
De forma que el sistema de enrutamiento dispone ahora de tablas que se especializan en almacenar la rutas e interactuar con los protocolos de enrutamiento y tablas que organicen los datos para optimizar la búsqueda e interactuar con los datos que circulan por la red. Esto resulta de gran ayuda e impulsó la separación de los planos dentro del mismo dispositivo.

Independientemente de la cantidad de direcciones de destino que completen la RIB o la FIB solo una pequeña cantidad corresponde a sus vecinos físicos, con lo cual sin importar el destino del que se trate, las tramas que se envíen solo pasarán por una cantidad limitada de routers (sus vecinos), que forman el próximo salto. Por ellos el router tiene una tabla de adyacencias con la información de reenvío preparada de antemano para reescribir la trama acelerando el proceso.

Para lograr el conocimiento de la topología y construir la RIB, la entidad o programa de control tiene que desarrollar una vista de la red. Esta visión de la red puede ser programada manualmente, aprendida a través de la observación, o construida a partir de información recopilada a través del diálogo con otras instancias de planos de control, que puede ser mediante el uso de uno o varios protocolos de enrutamiento, programación manual o una combinación de ambos.

La mecánica de los planos de control y de datos se demuestra en la Figura 4-2, que representa una red de conmutadores interconectados.

En la parte superior de la figura, se muestra una red de conmutadores, con una ampliación de los detalles de los planos de control y de datos de dos de estos

indicados como (A) y (B)

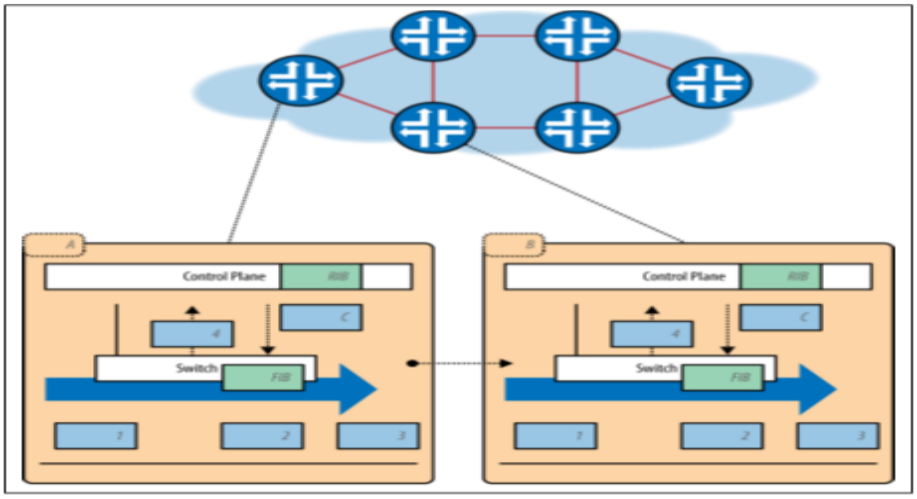



Figura 4-2: Planos de control y datos de una red típica [25]

Los paquetes son recibidos por el conmutador A en el plano de control más a la izquierda y finalmente remitidos al interruptor B en el lado derecho de la figura.

Dentro de cada dispositivo, los planos de control y de datos están separados, ejecutándose el plano de control en su propio procesador o tarjeta y el plano de datos en un plano separado dentro del mismo chasis.

Cualquier enrutador o conmutador multislot construido en los últimos 10 años tiene su plano de control (es decir, su cerebro) ejecutándose en un procesador, tarjeta dedicada y las funciones de conmutación del plano de datos ejecutándose independientemente en una o más tarjetas de línea, cada una de las cuales tiene un procesador dedicado y/o un procesador de paquetes.

En la Figura 4-3 se muestra como los diversos protocolos de enrutamiento, el control RIB y FIB son el corazón del plano de control. Observe que no se especifica dónde reside el plano de control y donde el de datos, pero este último se encuentra en la tarjeta de red (cuadro LC) y el plano de control está situado en el procesador de ruta (RP). [47]

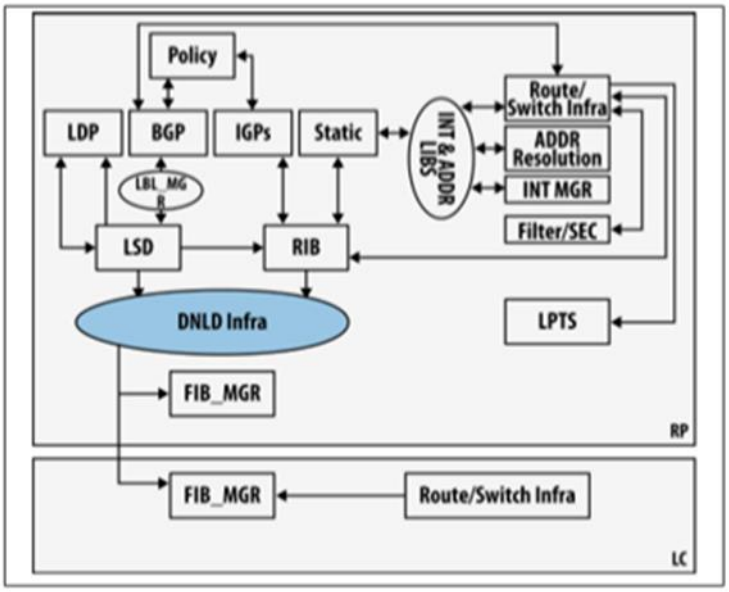



Figura 4-3: Planos de control y datos de un dispositivo de red típico

Un datagrama correcto se procesa en el plano de datos realizando búsquedas en la tabla FIB que son programadas anteriormente por el plano de control. Esto a veces se denomina la ruta rápida para el procesamiento de paquetes porque no necesita más cuestiones que las de identificar el destino del paquete usando el FIB preestablecido.

La única excepción a este procesamiento es cuando los paquetes no pueden coincidir con esas reglas, como cuando se detecta un destino desconocido, y estos paquetes se envían al procesador de ruta, donde el plano de control puede procesarlos mediante el RIB.

Históricamente, las búsquedas en tablas de hardware han demostrado un rendimiento de reenvío de paquetes mucho mayor y por lo tanto han dominado diseños de elementos de red, particularmente para anchos de banda más alto. Sin embargo, los recientes avances en el procesamiento de entrada y salida de los procesadores genéricos, impulsados por el crecimiento y la innovación de la computación en nube, están dando muy buenos diseños, particularmente en los rangos de desempeño medios a bajos.

No es raro ver que las plataformas basadas en hardware tengan un diseño con una tabla que al desbordar vuelvan a intentar las búsquedas que requieran más información adicional que las necesarias para lograr el "camino rápido" (por hardware) ya que este si bien actúa con mayor velocidad tiene limitaciones de recursos en el número de entradas. La segunda búsqueda utiliza una tabla mantenida por software - una búsqueda de ruta "lenta" que posibilita la utilización de mayor cantidad de recursos. Tampoco es raro combinar productos básicos y ASICs (Application Specific Integrated Circuit) [48] para realizar funciones basadas en la capa 2 frente a las funciones basadas en la capa 3, sin consolidarlas en un solo chip.

En algunos casos, la decisión de avanzar devuelve un puerto local, indicando que el tráfico está destinado a un proceso que se ejecuta localmente, como OSPF o BGP versión 6.

Los datagramas toman lo que se conoce como la trayectoria de desplazamiento por el cual abandonan la ruta de reenvío por hardware y son redirigidos hacia el procesador de rutas utilizando un canal de comunicación interno. Esta ruta es generalmente un camino de bajo rendimiento, ya que no está diseñado normalmente para el envío de paquetes de tráfico de alto rendimiento; Sin embargo, algunos diseños simplemente añaden una ruta adicional al tejido interno de conmutación con este propósito, lo que puede dar como resultado un reenvío a velocidad dentro del sistema.

## 4.3. Sobre la escalabilidad en redes tradicionales

Un problema de las redes incluye la administración el crecimiento de la base de información, es decir, el crecimiento del tamaño de la tabla de enrutamiento, no hacerlo podría dar como resultado una gran inestabilidad física de la red. Esto, a su vez, puede conducir a altas tasas de cambios de rutas dentro de la red o incluso la pérdida del servicio.

Otro desafío a superar a medida que crece el tamaño de la información de enrutamiento, es la responsabilidad de publicar los permisos de acceso (políticas) que posibilitan los envíos de datos de un destino a otro, atravesando sistemas autónomos y sus límites administrativos.

El progreso de Internet ocurrió porque estos protocolos evolucionaron tanto en términos de su funcionalidad que los proveedores de hardware aprendieron a implementarlos de forma altamente escalable.

En una red de capa 2, los comportamientos alrededor del aprendizaje de las direcciones MAC y los mecanismos utilizados para garantizar la escalabilidad permiten obtener un gráfico sin bucles y evitar la inundación con tráfico broadcast, unicast desconocido y multicast.

Sin embargo, las preocupaciones por escalar a partir de la capa 2 y capa 3 y los diseños de planos de control resultantes se fusionan o hibridan porque las redes de capa 2, en última instancia no escalan bien debido al gran número de hosts finales.

El centro de la cuestión son los hosts finales que, hoy en día, se mueven con la virtualización entre las redes, lo que resulta en una mezcla masiva de las tablas de reenvío que deben ser actualizadas lo suficientemente rápido como para no interrumpir el flujo de tráfico.

La cantidad de direcciones MAC de los hosts de grandes empresas pueden ser enormes, la gestión de estas direcciones es difícil, imagine si tiene que gestionar todas las direcciones MAC en varias empresas o en Internet.

En una red de capa 3, el reenvío se centra en la accesibilidad de las direcciones de red. La información sobre esta accesibilidad se basa en el prefijo IP de destino

En todos estos casos, el enrutador encamina el tráfico entre redes de la capa 3 y sólo enviará paquetes a la capa 2 cuando sepa que el paquete ha llegado a la red de destino final para ser entregada a un host específico.

## 4.4.Introducción al enrutamiento en SDN

En el caso de SDN y a un nivel muy alto, el plano de control establece el conjunto de datos local utilizado para crear las entradas de la tabla de reenvío. Las mismas son utilizadas por el plano de datos para reenviar tráfico entre puertos de entrada y salida de un dispositivo.

Al centralizar el control se puede obtener toda la información disponible en cada conmutador de la red en una única entidad, que acorde con los requerimientos de calidad de servicio, pueda manejar los flujos adaptándose dinámicamente a los cambios para lograr manipularlos a través de aplicaciones que utilizando la mencionada información interactúen modificando las reglas de flujo.

## 4.5.Enrutamiento SDN: Arquitecturas para el análisis

Dado, entonces, que el nuevo concepto introducido por SDN es el desacoplamiento de estos dos planos en la red, los dispositivos realizan solo el reenvío de datos, mientras que reenvían las decisiones que se toman en base al conjunto de reglas determinadas por un controlador externo La arquitectura SDN de referencia es ilustrada en la Fig. 4-4[49].

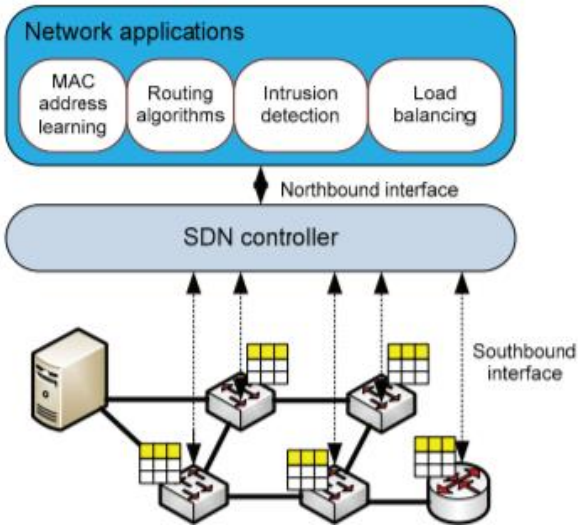



Figura 4-4: Arquitectura SDN

Para este análisis, se ha considerado el protocolo Open Flow. Dicho protocolo se utiliza para la comunicación entre el controlador SDN y los dispositivos del plano de datos (es decir, OpenFlow switches). OpenFlow permite el enrutamiento basado en flujo de decisiones Los conmutadores OpenFlow diferencian y procesan el tráfico, el cual fluye de acuerdo con las instrucciones recibidas del controlador. En sentido amplio, el flujo podría definirse como una secuencia de paquetes con características similares. Para definir un flujo, el controlador puede usar cualquier subconjunto de 9 campos de encabezado de paquete L2-L4 que se muestran en la Fig. 4-5, junto con el identificador de la interfaz a la que el paquete ha llegado. Esta opción de control altamente granular permite la implementación de enrutamiento dinámico de múltiples rutas, que podría aumentar significativamente la capacidad de la red [50].


Figura 4-5: Campos L2-L4 utilizados para una definición de flujo en OpenFlow

Las instrucciones del controlador se almacenan dentro de los dispositivos OpenFlow en forma de reglas de la tabla de flujo (Fig. 4-4). Cuando un paquete llega, el proceso de búsqueda comienza a buscar la regla correspondiente en la tabla. Si el paquete no coincide con alguna regla, se descarta. Sin embargo, el caso común es que se instale

una regla de baja prioridad que indique al conmutador que envíe el paquete al controlador . El controlador SDN luego define los siguientes pasos de procesamiento, de acuerdo con la red en ejecución

En el presente caso, se implementó funcionalidad de enrutamiento dinámico dentro de un Controlador POX OpenFlow [51]. Como se ilustra en la Fig. 4-6, el controlador gestiona varios módulos clave: Cálculo de costos de enlace, Cálculo de rutas, Módulo de enrutamiento, Recolección de estadísticas, Descubrimiento de topología, Interfaz de flujo abierto, Datos topológicos, entre otros. Se describe a continuación, algunos de ellos.

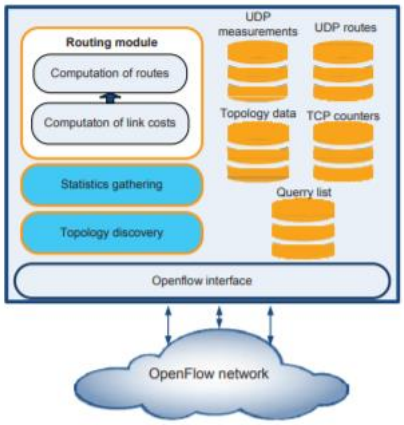



Figura 4-6: Esquema del diseño de un controlador propuesto.

1) Descubrimiento de topología: este módulo es una parte integral del controlador POX. Es responsable de descubrir y mantener la topología de la red. La información sobre la conectividad de red está constantemente disponible para los otros módulos de servicio que residen en el controlador. En caso de que el enlace sube o baja, este módulo eleva los eventos de cambio para notificar a los oyentes registrados.

2) Recopilación de estadísticas: para realizar un enrutamiento dinámico, el controlador necesita una vista actualizada del estado de la red. Una opción es medir la utilización del enlace periódicamente. Para ello se deberá enviar las solicitudes de estadísticas de puerto a los conmutadores OpenFlow. Cuando un conmutador OpenFlow recibe este mensaje, informa una cantidad de bytes enviados a través de la red correspondiente a la interfaz. Al comparar los valores del informe anterior con el último, el controlador SDN puede calcular la carga del enlace en el último ciclo de medición. Sin embargo, a medida que los flujos TCP tienden a usar todo el ancho de banda disponi-

ble en la ruta, este método podría llegar a la conclusión de que, por ejemplo, los enlaces cargados con solo una conexión TCP de larga duración (o algunas de ellas) también debe evitarse para el enrutamiento. En otras palabras, si es lo mismo el número de bytes que se envía a través de dos enlaces diferentes, durante el ciclo de medición, serán tratados igualmente incluso si la diferencia en el número de flujos de tráfico en ellos es considerable.

Como consecuencia, esto tendrá un efecto negativo en la equidad de enrutamiento y el rendimiento general de la red. Por esta razón, se realiza la estimación del ancho de banda disponible en los enlaces, para luego poner esta información a disposición del otro módulo.

Como los enlaces son bidireccionales, las estadísticas separadas son mantenidas para cada dirección. La razón se basa en el hecho de que TCP tiende a distribuir el ancho de banda disponible entre los flujos TCP individuales. Si se asume que el enlace es el cuello de botella para cada conexión TCP, se puede agregar una fórmula para la cantidad de ancho de banda que se podría ofrecer a un nuevo flujo TCP .

Ciertamente, esta es una suposición muy aproximada, pero para determinar el ancho de banda real que podría obtener una nueva conexión TCP, el controlador necesita estadísticas detalladas para cada flujo de tráfico en la red.

Las estimaciones del ancho de banda de los enlaces disponibles solo se utilizan como argumentos de entrada del módulo de enrutamiento. Desde todo el enrutamiento, se pueden proponer algoritmos que protejan enlaces con poco ancho de banda disponible.

Por otro lado, cuando se instala una ruta para el flujo UDP, el controlador agrega el interruptor de entrada del flujo a la lista de consultas (Fig. 4-6).

Los switches de entrada también son adecuados en términos de precisión, ya que las estadísticas recopiladas no tomarán la pérdida de paquetes dentro de la red. Para minimizar el control general, el controlador envía solo una solicitud para todos los flujos UDP instalados en el conmutador. La respuesta también consiste en un mensaje con los contadores para todas las entradas de flujo UDP

De las estadísticas obtenidas, el controlador deriva la información sobre el rendimiento de flujos UDP en el último ciclo de medida. Como se ilustra en la Fig. 4-6, además, se mantiene la lista de rutas UDP actuales, de modo que pueda calcularse el rendimiento UDP en cada enlace de red.

3) Módulo de enrutamiento: este módulo calcula los costos de enlace y determina rutas para flujos de tráfico. Utiliza datos de la topología y los resultados del módulo de recopilación de estadísticas como entradas

## 4.6. Northbound y Southbound

- Southbound

En SDN, las interfaces Southbound constituyen la especificación del protocolo OpenFlow que permite la comunicación entre controladores y conmutadores y otros nodos de red, considerados como los componentes de nivel inferior. Esto permite que el enrutador identifique la topología de la red, determine los flujos de la red e implemente la solicitud que se le envía a través de las interfaces hacia el norte.

Las API Southbound permiten que el usuario final obtenga un mejor control sobre la red y promueve que el nivel de eficiencia del controlador SDN evolucione en función de las demandas y necesidades en tiempo real. Además, la interfaz es un estándar de la industria que justifica el enfoque ideal que el controlador SDN debe comunicar con el plano de reenvío para modificar las redes que le permitirían avanzar progresivamente junto con las necesidades empresariales avanzadas. Para componer una capa de red más receptiva a las demandas de tráfico en tiempo real, los administradores pueden agregar o eliminar entradas a la tabla de flujo interna de conmutadores y enrutadores de red.

Algunas de las API populares hacia el sur son OpenFlow (analizada en el apartado anterior), Cisco y OpFlex. Otros proveedores de conmutadores y enrutadores que admiten OpenFlow son IBM, Dell, Juniper, Arista, entre otros

- Northbound

En oposición con la API Southbound, las interfaces Northbound permiten la comunicación entre los componentes de nivel superior. Mientras que las redes tradicionales usan firewall o balanceador de carga para controlar el comportamiento del plano de datos, SDN instala aplicaciones que usan el controlador y estas aplicaciones se comunican con el controlador a través de su interfaz Northbound.

Los expertos dicen que sería bastante difícil mejorar la infraestructura de red, ya que sin una interfaz Northbound, las aplicaciones de red provendrán directamente de los proveedores de equipos, lo que puede dificultar la evolución. Además, las API Northbound hacen que sea más fácil para los operadores de red innovar o personalizar los controles de red y procesar esta tarea no requiere la ayuda de expertos, ya que la API puede ser limpiada por un programador que sobresale en lenguajes de programación como Java, Python o Ruby

## 4.7. Otro enfoque: Segment Routing

Segment Routing (SR) es una forma flexible y escalable de enrutamiento de origen. La fuente elige un camino y codifica en el encabezado del paquete como una lista ordenada de segmentos. Los segmentos son identificados para cualquier tipo de instrucción. Cada segmento se identifica mediante el ID de segmento (SID) que consiste en un entero plano de 32 bits sin signo. Un ejemplo de instrucción puede ser el siguiente:

• Vaya al nodo N usando la ruta más corta

• Vaya al nodo N sobre la ruta más corta al nodo M y luego siga los enlaces Capa 1, Capa 2 y Capa 3

• Aplicar servicio S

Con Segment Routing, la red ya no necesita mantener un estado por aplicación y por flujo. En su lugar, obedece las instrucciones de reenvío proporcionadas en el paquete. El enrutamiento de segmentos se basa en una pequeña cantidad de extensiones a Cisco IntermediateSystem-to-IntermediateSystem (IS-IS) y los protocolos Open Shortest Path First (OSPF). Puede funcionar con MPLS o un plano de datos IPv6, y se integra con las capacidades de servicios múltiples de MPLS, que incluyen VPN de capa 3 (L3VPN), servicio de cable privado virtual (VPWS), servicio de LAN privada virtual (VPLS) y Ethernet VPN (EVPN). El enrutamiento de segmentos se puede aplicar directamente a la arquitectura de conmutación de

etiquetas multiprotocolo (MPLS) sin cambio en el plano de reenvío. El enrutamiento de segmentos utiliza el ancho de banda de la red de manera más efectiva que Redes MPLS tradicionales y ofrece una latencia más baja. Un segmento está codificado como una etiqueta MPLS.

La lista de segmentos está codificada como una pila de etiquetas. El segmento a procesar está en la parte superior de la pila. El relacionado a la etiqueta se saca de la pila, después de completar un segmento.

El enrutamiento de segmentos se puede aplicar a la arquitectura IPv6 con un nuevo tipo de encabezado de extensión de enrutamiento.

El segmento se codifica como una dirección IPv6. Una lista ordenada de segmentos se codifica como una lista ordenada de direcciones IPv6 en el encabezado de la extensión de enrutamiento. El segmento a procesar se indica mediante un puntero en la ruta del encabezado de extensión. El puntero se incrementa después de completar un segmento.

El enrutamiento de segmentos proporciona protección automática del tráfico sin restricciones topológicas. La red protege el tráfico contra fallas de enlaces y nodos sin requerir señalización adicional en la red existente. La Tecnología de enrutamiento rápido de IP (FRR), en combinación con las capacidades de enrutamiento explícito en el enrutamiento de segmentos, garantiza una cobertura de protección total con rutas de respaldo óptimas. La protección del tráfico no impone requisitos de señalización adicionales.[52]

Segment Routing se encuentra preparado para su aplicación en SDN: el enrutamiento de segmentos es una arquitectura convincente concebida para ser adoptada por las redes definidas por software (SDN) y es la base del enrutamiento de ingeniería de aplicaciones (AER). Logra un equilibrio entre la inteligencia distribuida basada en la red, como la protección automática de enlaces y nodos, e inteligencia centralizada basada en el controlador, como la optimización del tráfico. Puede proporcionar una red con garantías de rendimiento, uso eficiente de los recursos de red y una escalabilidad muy alta para aplicaciones.

La red utiliza información de estado mínima para cumplir con estos requisitos. Segment Routing se puede integrar fácilmente con una arquitectura SDN basada en controlador. La siguiente figura(Figura 4-7) ilustra una muestra de un escenario SDN en el que el controlador realiza una optimización centralizada, incluido el control de admisión de ancho de banda. En este escenario, el controlador tiene una imagen completa de la topología y los flujos de

la red. Un enrutador puede solicitar una ruta a un destino con ciertas características, por ejemplo, demora, ancho de banda, diversidad, entre otras.

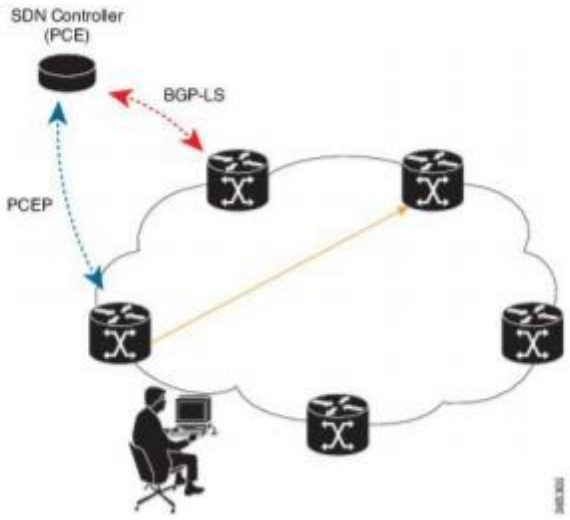



Figura 4-7

Además, las listas de segmentos permiten la virtualización completa de la red sin agregar ningún estado de aplicación a la misma. El estado está codificado en el paquete como una lista de segmentos. La red solo mantiene el estado del segmento, pudiendo admitir una gran cantidad, y una frecuencia más alta, de aplicaciones basadas en transacciones y solicitudes sin crear ninguna carga en la red.

## 4.8. Sobre la necesidad de una interface que posibilite la programación de aplicaciones- I2RS

Uno de los problemas de centralizar los planos de control es poder acceder en tiempo real y sincronizada a los dispositivos que conforman la red con el objeto de configurarlos y posteriormente obtener estadísticas que permitan el análisis postprocesamiento, hoy en día este proceso es lento pero se están realizando esfuerzos a través del proyecto I2RS (RFC 7921) [50] para acelerar este bucle de retroalimentación con los elementos de la red.[48][49]

El Grupo de trabajo I2RS se creó en noviembre de 2012 con el objetivo de desarrollar un conjunto de casos de uso y una arquitectura básica para admitir una interfaz con

el sistema de enrutamiento. El término "sistema de enrutamiento" describe un dispositivo de hardware, un enrutador virtual o cualquier software que proporcione funciones de enrutamiento.[53]

La interfaz del sistema de enrutamiento permitirá las siguientes funcionalidades a los paquetes de software:

- Inyectar o recuperar información, políticas y parámetros operativos hacia o desde el sistema de enrutamiento.
- La interfaz permitirá proporcionar acceso de lectura y escritura a la base de información de enrutamiento (RIB), pero no a la base de información de reenvío o FIB.
- Controlar y analizar las operaciones de BGP, y establecer y activar políticas relacionadas con el protocolo.
- Optimizar y elegir puntos de salida de red, basando la decisión en factores distintos a los proporcionados por los protocolos de enrutamiento.
- Apoyar una reacción rápida y distribuida a los ataques basados en la red.
- Desviar el tráfico hacia el destino bajo ataque mientras mantiene la operación normal para otras rutas.
- Modificar el enrutamiento de la capa de servicio para mejorar el tráfico de concentrador y radio existente.
- Extraer información de topología de la red.

Los enrutadores que forman la infraestructura de enrutamiento de Internet mantienen los detalles y las funciones del estado de varias capas. Por ejemplo, un típico enrutador mantiene una base de información de enrutamiento (RIB) e implementa los protocolos de enrutamiento como OSPF, IS-IS y BGP para intercambiar información

de accesibilidad, información de topología, estado del protocolo y otra información sobre el estado de la red con otros enrutadores.

Los enrutadores convierten toda esta información en entradas de reenvío, que luego se utilizan para reenviar paquetes y flujos entre los elementos de la red.

En el capítulo 3 detallamos como SDN trata los paquetes, sin embargo es posible observar que las reglas que conforman la tablas de flujo requerían la información topológica de la red que permita establecer la base de información de enrutamiento RIB y la configuración de los dispositivos.

El plano de reenvío y las entradas de reenvío especificadas contienen información del estado activo que describe el comportamiento operativo esperado y observado del enrutador y eso también es necesario para las aplicaciones de red. Las aplicaciones orientadas a la red requieren fácil acceso a esta información para aprender la topología de la red, para verificar que el estado programado esté instalado en el plano de reenvío, para medir el comportamiento de varios flujos, rutas o entradas de reenvío, así como para comprender los estados configurados y activos del enrutador. Las aplicaciones orientadas a la red también requieren un fácil acceso a una interfaz, que les permitirá programar y controlar el estado relacionado al reenvío.

La interfaz al sistema de enrutamiento (I2RS) [54] facilita el control y la observación del estado de las rutas, también permite que las aplicaciones orientadas a la red se construyan interactuando con las redes existentes.

Estas aplicaciones orientadas a la red pueden aprovechar I2RS como una interfaz programática para crear nuevas formas de combinar la recuperación de los datos del enrutamiento de Internet, análisis de ellos y la configuración del estado dentro de los enrutadores.

I2RS proporciona un marco para que las aplicaciones (incluidas las aplicaciones de controlador) registren y soliciten la información adecuada para cada aplicación en particular.

Hay cuatro claves respecto a los controladores que dan forma a la arquitectura I2RS.

1- la necesidad de una interfaz que sea programática y asíncrona y eso ofrece acceso rápido e interactivo para operaciones atomizadas.

2- el acceso a la información estructurada y al estado que frecuentemente no es directamente configurable o modelado en las implementaciones existentes o en los protocolos de configuración.

3- la capacidad de suscribirse a notificaciones de eventos estructurados del enrutador

4- la operación de I2RS debe proporcionar modelos de datos estándar para ser utilizados por aplicaciones de red.

El estado efímero de un enrutador es el estado que desaparece cuando se reinicia el dispositivo de enrutamiento o al reiniciarse el software que maneja el I2RS en el dispositivo. El protocolo de enrutamiento o la aplicación podrían inyectar el estado en un elemento de enrutamiento a través de la funcionalidad de inserción de estado de I2RS y ese estado luego podría distribuirse mediante algún protocolo de enrutamiento o señalización para ser utilizado localmente (p. ej., para programar el plano de reenvío compartido).

Un cliente local opera en el mismo contenedor físico que el sistema de enrutamiento mientras que, un cliente remoto lo hace en toda la red. Los detalles de cómo las aplicaciones se comunican con un cliente remoto está fuera del alcance de I2RS.

Los agentes y los clientes de I2RS se comunican entre sí mediante un protocolo asincrónico, un solo cliente puede publicar múltiples solicitudes simultáneas, ya sea a un solo agente o a múltiples agentes. Además, un agente puede procesar múltiples solicitudes, ya sea de un solo cliente o de múltiples clientes, simultáneamente.

El agente I2RS proporciona acceso de lectura y escritura a los datos seleccionados en el elemento de enrutamiento que se dispone para los servicios I2RS.

Los agentes I2RS pueden escribir el estado efímero estático (por ejemplo las entradas a la RIB) y leer tanto las rutas estáticas como las dinámicas (p. ej., Identificador de ruta conmutada de etiqueta MPLS (LSP-ID) o el número de vecinos activos de BGP). También, el agente I2RS permite a los clientes suscribirse a diferentes tipos de notificaciones de eventos que afectan diferentes instancias de los objetos, por ejemplo la notificación de un evento que resuelve el siguiente salto en la RIB de una manera que le permita ser utilizado por un administrador de RIB para su instalación en el plano de reenvío como parte de una ruta particular.

Los tipos de modelos de datos, asociados con el sistema de enrutamiento que el agente I2RS puede acceder y modificar, permite la configuración dinámica, configuración estática, configuración local y configuración de enrutamiento y señalización.

Cuando un elemento de enrutamiento implementa algún subconjunto de sistema de enrutamiento. No necesita tener un plano de reenvío asociado para ello. Los ejemplos de elementos de enrutamiento pueden incluir:

-Un enrutador con un plano de reenvío y RIB Manager que se ejecuta IS-IS, OSPF, BGP, PIM, etc.,

-Un elemento de enrutamiento se puede administrar localmente, ya sea a través de la línea de comandos interfaz (CLI), SNMP o el protocolo de configuración de red (NETCONF).

El módulo de enrutamiento y señalización es la parte del elemento de enrutamiento que se implementa para interactuar con el sistema de enrutamiento de Internet. Incluye no solo protocolos estandarizados (es decir, IS-IS, OSPF, BGP, PIM, RSVP-TE, LDP, etc.), sino también la capa de administración de RIB.

Para lograr el enrutamiento dinámico un agente I2RS necesita acceso al estado del elemento de enrutamiento más allá del subsistema de enrutamiento, ya que puede necesitar información de varios contadores, estadísticas, datos de flujo y eventos locales. Este estado operativo necesario para las aplicaciones de red basadas en I2RS no está contenido en la información estandarizada de enrutamiento y señalización emergente de I2RS, aun cuando dicha información es proporcionada al agente, es decir que está fuera del alcance de I2RS, sin embargo, los modelos de información y datos son parte de él.

Un ejemplo del estado de un sistema estático consiste en especificar el comportamiento de colas para una interfaz o el tráfico, como el agente I2RS modifica u obtiene esta información está fuera del alcance de esta norma, pero los modelos para el tratamiento de los datos son parte de ella.

En los últimos tiempos se han realizado muchos esfuerzos para mejorar el acceso a la información disponible para un sistema de enrutamiento y reenvío. Los mayores beneficios se lograron haciendo visible y utilizable la información de la red para su gestión a través de aplicaciones. Hay dos desafíos relacionados al hacerlo. Primero, la cantidad y diversidad de la información potencialmente disponible es muy grande. En segundo lugar, la variación tanto en la estructura de los datos como en los tipos de operaciones requeridas tiende a introducir complejidad en el protocolo.

Los tipos de operaciones contemplados aquí son complejos por naturaleza, pero es fundamental que I2RS sea fácilmente implementable y robusto.

Los modelos de datos demasiado complejos tienden a complicar los sistemas de información al intentar describir y contemplar todas las opciones posibles, complicando la extensibilidad, por eso I2RS no intenta agregar complejidad más allá de lo necesario para satisfacer los requisitos.

Un administrador de topología incluye un cliente I2RS que usa los modelos de da-tos I2RS y un protocolo para recopilar información sobre el estado de la red comunicándose directamente con uno o más agentes I2RS. Desde estos agentes, el Administrador de topología recopila la configuración de enrutamiento y datos operativos, como la interfaz y la Información de la etiqueta de ruta conmutada (LSP). Además, el administrador de topología puede recopilar datos de estado de enlace de varias maneras, a través de modelos I2RS, con BGP-LS (RFC 7752)[55], o escuchando el IGP.

El conjunto de funcionalidades y la información recopilada por el Administrador de topología puede integrarse como un componente de una aplicación, como una aplicación de cálculo de ruta. Como soporte de la aplicación, el Administrador de topología podría ser útil para que otras aplicaciones de red al proporcionar una imagen coherente del estado de accesibilidad de la red a través de otra interfaz. Esa interfaz podría usar el mismo protocolo I2RS o podría proporcionar un servicio de topología utilizando extensiones de los modelos de datos I2RS.

Para realizar esta tarea, la entidad o programa de control tiene que desarrollar una vista de la red, junto con la topología que satisface ciertas condiciones. Las mismas se corresponden con una visión de la red que puede ser programada manualmente, aprendida a través de la observación, o construida a partir de información recopilada a través del intercambio con otros registros del plano de control.

En el caso de SDN, toda esta información que se debe procesar y transmitir pertenece al denominado plano de control. En este contexto, se puede definir el plano de control como la inteligencia que determina los caminos óptimos para enviar la información y que responde a los incidentes y a las nuevas demandas de la red [56].

En síntesis, la idea básica de I2RS es crear un protocolo y componentes que actúen como un medio de programar la base de información de enrutamiento (RIB) de un dispositivo de red utilizando un protocolo de ruta rápida que permite un corte rápido de las operaciones de aprovisionamiento con el fin de permitir real -time interacción con el RIB y el gerente RIB que lo controla. Anteriormente, el único acceso que tenía el RIB era a través del sistema de configuración del dispositivo (en el caso de Juniper [57], Netconf [58] o SNMP [59]).

I2RS proporciona diversos niveles de abstracción en términos de programación de rutas de red, políticas y configuración de puertos. Un ejemplo es proporcionar a los sistemas de soporte operativo (OSS) [60] un acceso rápido y óptimo a la RIB.

I2RS también se presta bien a un creciente deseo de centralizar lógicamente el enrutamiento y las decisiones de ruta y programabilidad. El protocolo tiene requisitos para ejecutarse en un dispositivo o fuera de un dispositivo. De esta manera, la funcionalidad de controlador distribuido se abraza en los casos en que se desea.

Finalmente, otro subcomponente clave de I2RS es la topología normalizada y abstracta. La definición de un modelo de objeto común y extensible representará esta topología. El servicio también permite que se expongan múltiples abstracciones de representación topológica. Un aspecto clave de este modelo es que los dispositivos que no son routers (o enrutadores de protocolos de enrutamiento) pueden manipular y cambiar más fácilmente el estado de la RIB en el futuro. Hoy en día, los usuarios no tienen una gran dificultad para obtener esta información en el mejor de los casos. En el futuro, los componentes de una gestión de red / OSS, analítica u otras aplicaciones

que todavía no podemos prever serán capaces de interactuar rápida y eficientemente con el estado de enrutamiento y la topología de red.

## 4.9.Sobre la interacción con las redes en uso en SDN

Una afirmación relacionada con SDN viene de la proposición de pizarra limpia, es decir, el planteo de una tecnología que borre las utilizadas anteriormente, pensando los mecanismos necesarios para su funcionamiento, sin tener que interconectarla al modelo distribuido, evitando los costos tecnológicos en cuanto a la complejidad agregada para su adaptación.

Esta proposición postula que al observar el desarrollo gradual de una tecnología como MPLS que siguió un camino donde las actualizaciones y modificaciones de sus características hicieron abultadas las bases de código de las implementaciones existentes, transformándolas en demasiado complejas y frágiles.

En algunas implementaciones [61], utilizando la distribución centralizada de etiquetas para emular la funcionalidad distribuida del LDP [62] o RSVP [63], el conocimiento centralizado de la topología de red utilizando una base de código, arrojó resultados con por lo menos un orden de magnitud menores que las bases comerciales actualmente disponibles.

La afirmación natural es que en un sistema de control altamente prescriptivo y centralizado, el comportamiento de la red puede aproximarse al de un reenvío completamente estático, que es discutiblemente estable.

Cuando la utilización de la red se incrementa, deben desplegarse nuevos dispositivos para satisfacer la demanda. Resultaría importante poder satisfacer la demanda de transferencia de reenvío sin modificar el número de dispositivos gestionados y sus entidades de protocolo de control resultantes en la red, o por lo menos lograr determinar las condiciones en las cuales debe integrarse ese dispositivo, ya que, por cada incorporación de uno de ellos, se incrementa la información de control.

Si pensamos que nuestro sistema centralizado debe interactuar con dispositivos distribuidos típicos en los sistemas de enrutamiento y conmutación tradicionales, es importante comprender que los planos de control deben sincronizarse para lograr coherencia en la información, tanto en el controlador como en router tradicional. El plano de control adicional afecta la escalabilidad del plano de control general de la red durante la convergencia de red, es decir, el tiempo que tarda la totalidad de los planos de control para alcanzar un circuito libre de bucles que representa el estado de la red. Esto se manifiesta en la resiliencia y el rendimiento del sistema en general, y cuanto mayor sea el número de planos de control habrá potencialmente una mayor fragilidad en el sistema

Contrariamente, si se sintoniza adecuadamente también aumenta la resiliencia del sistema, ya que crea un sistema que eventualmente se vuelve consistente independientemente de las condiciones.

En pocas palabras, el número de dispositivos con planos donde se aplican los protocolos en modelos de control de consistencia distribuidos puede crear complejidad de gestión.

Por lo tanto, parece inevitable descartar la idea inicial de la pizarra limpia ya que las redes actuales, con sus protocolos de enrutamiento seguirán existiendo y deberán interactuar con las redes SDN, generando una topología mixta como se observa en los trabajos comentados de I2RS.

## 4.10. Estrategias para el presente trabajo

Los cambios en la configuración de enrutadores y conmutadores modernos afectan los resultados obtenidos en los planos de control y datos y el modo de administrar centralmente, afectará y modificará los obtenidos por los nodos distribuidos.

Para muchos operadores de red, el control tiene que ver con la flexibilidad para afectar los resultados de las decisiones de reenvío y la capacidad de hacerlo simple y programático. Las soluciones simples y la sencillez en su implementación se privilegian por sobre otras, más complejas. Esta será la base del desarrollo que se expone en el Capítulo Nº5.

Esto incluye hacer la red más elástica y eficiente sobre la base de conocimientos adicionales o demandas que tenemos por encima y más allá de la determinación algorítmica considerada óptima.

Tanto las rutas estáticas como las políticas de ruta también tienen escalabilidad limitada en la mayoría de las implementaciones.

En principio, una mirada centralizada permitirá establecer estrategias para modificar los escenarios del enrutamiento distribuyendo los flujos conforme se modifiquen parámetros de calidad de servicio, contemplando no solo el camino óptimo como medida del rendimiento de la red, sino el uso posible de todos los enlaces. También se contemplan como parámetros posibles para la redistribución, el uso de un valor umbral de la perdida de paquetes o el ancho de banda disponible.

# Capítulo 5: Desarrollo

## Introducción

En este capítulo se pretende mostrar, dado un escenario propuesto cuya elección se justifica más adelante, la sencillez en la implementación de soluciones sobre redes SDN en comparación con redes tradicionales, sobre el enrutamiento y direccionamiento de flujos específicos. Sobre el escenario elegido, se buscó la solución priorizando, fundamentalmente, su simplicidad, con el objetivo de comparar la respuesta de redes tradicionales y la de redes definidas por software (SDN) frente al envío de dos tipos de tráfico: uno de ellos sobre protocolo UDP que representa al tráfico priorizado respecto del segundo de ellos sobre protocolo TCP.

## Herramientas utilizadas para el desarrollo

Para el desarrollo de la experimentación, se han elegido herramientas de medición y entornos de simulación específicos que se detallan a continuación.

### IPERF versión 3

iPerf [64]es la **herramienta más popular para medir el ancho de banda de una red y la perdida de paquetes**, esto es debido a que es multiplataforma y la podemos encontrar tanto en sistemas operativos Windows, Linux, Mac OS X como también en distribuciones basadas en Unix como FreeBDS. Hasta hace poco tiempo, la versión 2 de iPerf era la más usada, y con ella había variantes de ella con entorno gráfico, ya que iPerf trabaja a través de línea de órdenes habitualmente. La versión iPerf3 incorporó una gran cantidad de cambios y mejoras

**iPerf3 es un rediseño de iPerf**. Esta nueva versión se ha programado desde cero con el objetivo de tener un código más simple y mucho más pequeño, asimismo también se ha hecho una biblioteca para que otros programas puedan utilizar esta base. iPerf3 ha incorporado varias características que tienen otras herramientas como nuttcp [65]

y netperf [66]pero que no estaban en la versión iPerf original como por ejemplo un modo de copia cero y también sacar la salida en formato JSON [67].

Actualmente iPerf3 es compatible con CentOS Linux, FreeBSD y Mac OS X ya que se ha desarrollado bajo estos sistemas operativos, no obstante, también debería funcionar sin problemas con OpenBSD y otras distribuciones Linux, aunque de momento no tienen soporte oficial por parte de los desarrolladores.

Además, IPERF versión 3.0.7 es considerada más estable para el sistema operativo utilizado que es el UBUNTU 14.04.1 LTS (Long Term Support) [68]

### GNS3

GNS3 [69] (Graphic Network Simulation o Simulación Gráfica de Redes) es un simulador gráfico de red que permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellos. Con GNS3 los usuarios tendrán la posibilidad de poder escoger cada uno de los elementos que llegarán a formar parte de una red informática. GNS3 está estrechamente vinculada con:

- Dynamips [70], un emulador de IOS que permite a los usuarios ejecutar binarios imágenes IOS de Cisco Systems.

- Dynagen [71], un front-end basado en texto para Dynamips

- Qemu [72], un emulador de PIX.GNS3 es una excelente herramienta complementaria a los verdaderos laboratorios para los administradores de redes de Cisco o las personas que quieren pasar sus CCNA, CCNP, CCIE DAC o certificaciones.

La versión utilizada para el presente trabajo fue la GNS3 2.1.21. Esta es una versión menor que corrige errores encontrados desde la versión 2.1.20

### Mininet

El emulador Mininet es una plataforma de pruebas de red de código abierto y rápidamente configurable. Mininet consiste en una herramienta de prueba y desarrollo para SDN, aprovechando los requisitos de los ordenadores de una manera óptima para lograr emular diferentes proyectos en un ambiente virtual. Utiliza una virtualización

liviana para hacer que un solo sistema se vea como una red completa, ejecutando el mismo kernel, sistema y código de usuario logrando que un host Mininet se comporte como una máquina real. Hasta ahora, es la herramienta de apoyo a la investigación de las SDN OpenFlow más conocida. [73][75]. Mininet utiliza hosts virtuales, switches y enlaces para crear una red en un solo núcleo del sistema operativo, y utiliza la pila de red real para procesar paquetes y conectarse a las redes reales. Además, las aplicaciones de red basadas en Unix/Linux, también se pueden ejecutar en los hosts virtuales. En una red OpenFlow emulada por Mininet, una aplicación de controlador real OpenFlow se puede ejecutar en una máquina externa o en el mismo equipo en el que se emulan los hosts virtuales [74][75].

En cuanto al controlador utilizado se definió RYU [76]. Ryu posee la ventaja que está basado en Python, como los scripts de Mininet y permite manejarse siempre en el mismo lenguaje. También es uno de los pocos controladores que soporta hasta la versión 1.5 de OpenFlow.

## Escenario propuesto

El procedimiento y pruebas se realizaron bajo distintas condiciones y tipos de flujo. El escenario elegido se basa en el planteo de un problema conocido de los enrutamientos sobre redes tradicionales, utilizado para presentar la necesidad de Ingeniería de Tráfico y Calidad de Servicio para obtener la solución deseada. De acuerdo a diversas fuentes y entre ellas Cisco System, se asegura la elección de una topología adecuada y confiable. Dicha solución planteada sobre este escenario requiere la aplicación, no sólo de protocolos de enrutamiento, sino también de mecanismos de calidad de servicio y de balanceo de carga. Se consideró la siguiente topología que se describirá a continuación:

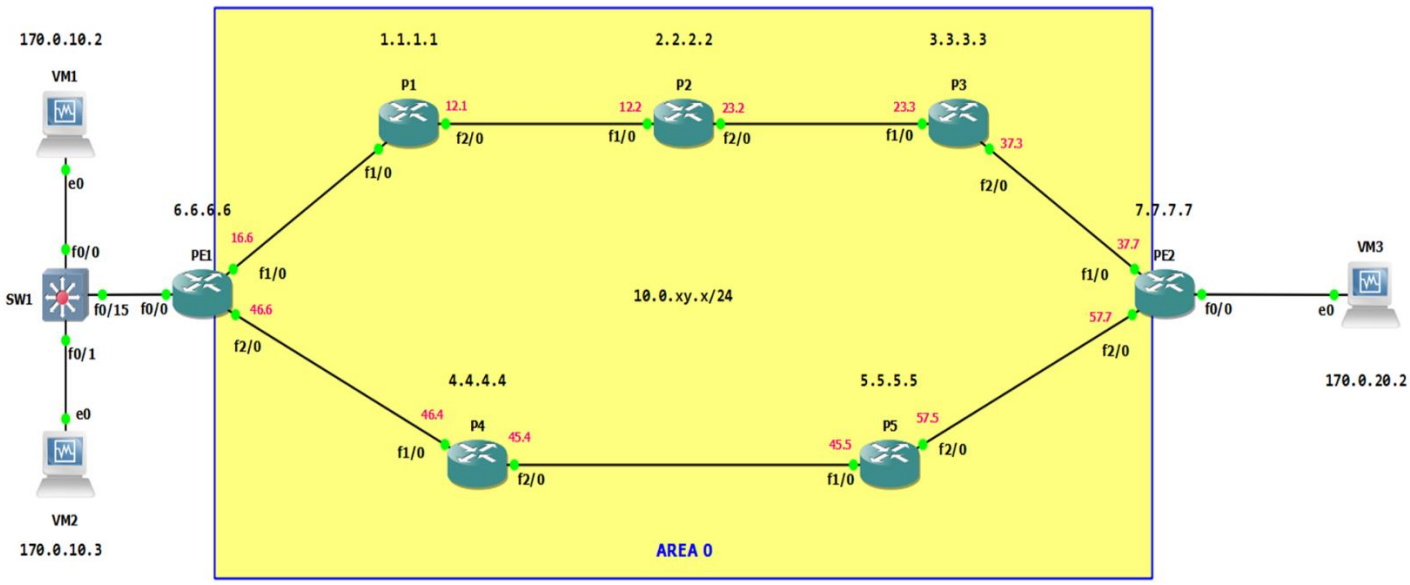



Figura 5.1. Escenario Propuesto. Fuente propia

Los flujos de información provenientes de la VM1 y la VM2 (Virtual Machine 1 y 2) fueron considerados, para su ingreso al "área 0" y en particular, al router de borde indicado en la Figura 5.1. como PE-1, según un sistema de encolamiento. Se detalla, a continuación, una breve descripción sobre los tipos de colas y, particularmente, el elegido para el desarrollo del presente trabajo.

## Tipos de Colas

Un flujo es un conjunto de paquetes que tienen la misma dirección IP origen y destino y los mismos números de puerto tanto origen como destino.

Para definir sistemas de colas, hay que mencionar que primero existe un flujo o varios, que puede o no estar en una congestión en un sistema de comunicación de datos.

A ello, se desarrollan sistemas de encolamiento que tienen un impacto en la medida que afecta: ancho de banda, retardo, jitter y pérdida de paquetes. Ciertamente, los sistemas de encolamiento son muy a menudo los mecanismos desarrollados más importantes. Para QoS se usan los siguientes métodos:

- Encolamiento FIFO (First In First Out): no hay concepto de prioridad ni clasificación. El primer paquete que entra a un sistema de cola, es el primero en salir, como se muestra en la Figura

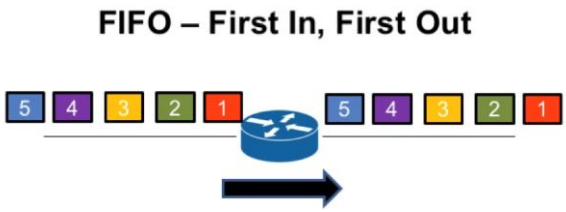



Figura 5.2. Encolamiento FIFO. Fuente: Cisco Systems [77]

- Encolamiento por espera equitativa ponderada (WFQ - Weighted Fair Queuing): Este método de encolamiento clasifica paquetes en flujos por prioridades. A medida que cada nivel de prioridad de los grupos clasificados se vacían, el sistema actúa sobre la siguiente prioridad o grupo, según se aprecia en la Figura 5.3


Figura 5.3. Encolamiento WFQ

- Encolamiento por espera equitativa ponderada basado en clases (CBWFQ – Class Based Weighted Fair Queuing): a comparación de WFQ la cual tiene algunas limitantes de escalamiento, ya que la implementación del algoritmo se ve afectada a medida que el tráfico por enlace aumenta, colapsa debido a la cantidad numerosa de flujos que analizar. CBWFQ usa un algoritmo ponderado de programación de Round Robin, donde establece tiempos para el sistema de colas proporcionado un ancho de banda para cada flujo, respectivamente (Figura 5.4).

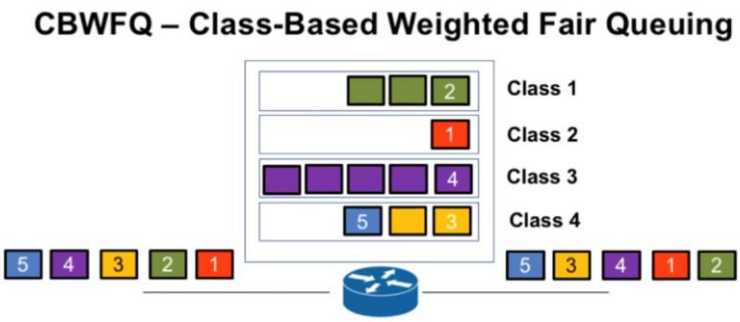



Figura 5.3. Encolamiento WFQ

- Encolamiento de baja latencia (LLQ – Low Latency Queueing): consta de colas de prioridad personalizadas, basadas en clases de tráficos, en conjunto con una cola de prioridad, la cual tiene preferencia absoluta sobre las otras colas.

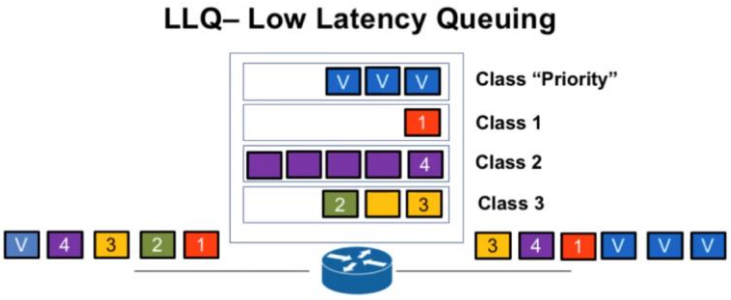



Figura 5.4. Encolamiento LLQ

- Encolamiento CBWFQ + LLQ: es actualmente el método de encolamiento recomendado para Voz sobre IP (VoIP) y telefonía IP, y con videoconferencias. Con las funcionalidades de CBWFQ, y tráficos que necesiten prioridad absoluta (LLQ), combinan el mejor método de encolamiento para traficar datos, transacciones y flujos UDP (VoIP, Videoconferencias)

El tipo de encolamiento LLQ (Figura 5.4) y el tipo CBWFQ (Figura 5.3) son los utilizados en el presente desarrollo.

En nuestro experimento se pretende priorizar el flujo que transporta mensajes UDP, tomando como premisa obtener la mínima pérdida de paquetes correspondiente a dicho flujo, por ello se buscó un modelo de colas que brinde una prioridad absoluta para ellos.

Si bien, no se desconoce que UDP se utiliza en aplicaciones que pueden ser tratadas como best effort, también se utilizan como transporte para aplicaciones real time, se consideró que a los fines del experimento no resulta de interés una granularidad tan fina.

## Diseño del experimento

Como se explica en la introducción a este capítulo, se mostrará que el protocolo de enrutamiento por si solo es incapaz de dar respuesta a este escenario con el objetivo planteado de sencillez y simplicidad, ya que los proveedores de servicios evitan la utilización de Ingeniería de Tráfico, aún aquellos que utilizan redes MPLS, debido a su alta complejidad.

Dado el propósito de la minimización de la pérdida de paquetes, se muestra a continuación los resultados obtenidos, los cuales responden a diferentes casos. Estos casos fueron diseñados de manera tal que cada uno de ellos agrega una mejora sobre el anterior, para analizar la respuesta de la red en su conjunto según los escenarios propuestos.

**Para las redes tradicionales** se consideró un envío sencillo para probar el ancho de banda disponible. Luego, la siguiente prueba, consideró ambos flujos de tráfico, los que se enviaron por la ruta definida por el protocolo de ruteo. Por último, se enviaron nuevamente ambos flujos de tráfico agregando políticas de calidad de servicio.

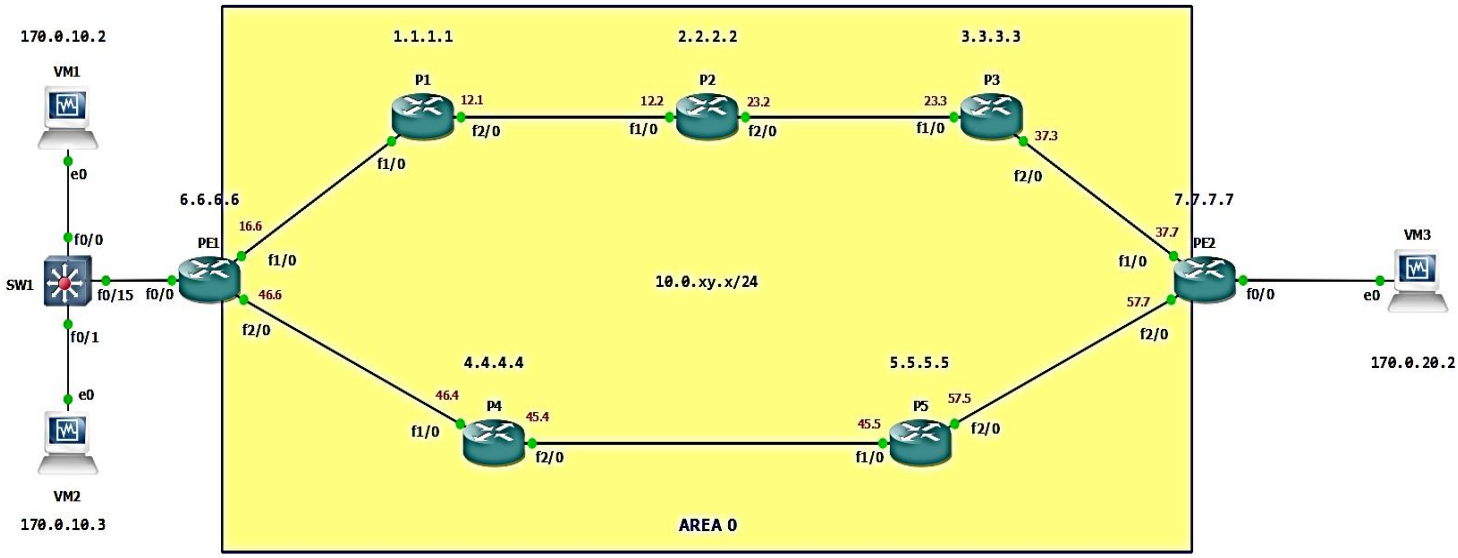
**Para el caso de SDN**, se realizó un primer envío sobre protocolo TCP a través de la ruta más larga, con el mismo objetivo que en el caso de las redes tradicionales: alcance del ancho de banda disponible. Luego se consideraron nuevamente, sendos tráficos sobre UDP y TCP. En un primer caso, se enviaron ambos por la ruta más larga del escenario planteado. Por último, con el objeto de mejorar la disminución de la pérdida de paquetes, se envió el tráfico UDP por la ruta más corta y el tráfico TCP por la ruta más larga.

Las reglas sobre SDN, a los efectos de este experimento, se realizaron directamente sobre la configuración de los dispositivos ya que se busca plantear las experiencias que resuelven el problema, como se plantea en el último párrafo de la página 62. Es

importante destacar que también estas reglas se podrían introducir en el controlador, de manera tal que las líneas de código integradas permitan modificar las tablas de flujo cuando las condiciones del tráfico lo determinen.

## Casos analizados

Volviendo al escenario de la Figura 5.1.




## Pruebas sobre Redes tradicionales

### Pruebas con Iperf3 Flujo Simple (sin QoS) sobre GNS3 - (Redes Tradicionales – Protocolo OSPF)[78]

Se realizó el envío de un primer flujo bajo protocolo TCP [79] por el puerto 5201 desde VM1 a VM3. Dicho test, ha determinado la capacidad del sistema a 17Mbits/sec aproximadamente. De acuerdo a este parámetro disponible, el límite del ancho de banda será determinante para realizar las siguientes pruebas de rendimiento. De acuerdo al protocolo OSPF, la ruta elegida es, dentro del "Área 0": PE1 – P4 – P5 – PE2

Se explicita, a continuación, el envío de flujo TCP generado en Iperf desde VM1 a VM3, el cual lo recibe a través del puerto 5201:

```
VM1: iperf3 -c 170.0.20.2 -p 5201 -t 60

-----------------------------------------------------------

Server listening on 5201

-----------------------------------------------------------

Accepted connection from 170.0.10.2, port 48222

[  5] local 170.0.20.2 port 5201 connected to 170.0.10.2 port 48222

[ ID] Interval           Transfer     Bandwidth

[  5]   0.00-1.00   sec  1.96 MBytes  16.5 Mbits/sec
[  5]   1.00-2.00   sec  2.05 MBytes  17.2 Mbits/sec
[  5]   2.00-3.00   sec  2.05 MBytes  17.2 Mbits/sec
[  5]   3.00-4.00   sec  2.06 MBytes  17.2 Mbits/sec
[  5]   4.00-5.00   sec  2.08 MBytes  17.4 Mbits/sec
[  5]   5.00-6.00   sec  2.05 MBytes  17.2 Mbits/sec
[  5]   6.00-7.00   sec  2.05 MBytes  17.2 Mbits/sec
[  5]   7.00-8.00   sec  2.06 MBytes  17.3 Mbits/sec
[  5]   8.00-9.00   sec  2.07 MBytes  17.3 Mbits/sec
[  5]   9.00-10.00  sec  2.05 MBytes  17.2 Mbits/sec
[  5]  10.00-11.00  sec  2.05 MBytes  17.2 Mbits/sec
[  5]  11.00-12.00  sec  2.05 MBytes  17.2 Mbits/sec
[  5]  12.00-13.00  sec  2.07 MBytes  17.4 Mbits/sec
..................

[  5]  54.00-55.00  sec  2.05 MBytes  17.3 Mbits/sec
[  5]  55.00-56.00  sec  2.08 MBytes  17.4 Mbits/sec
[  5]  56.00-57.00  sec  2.05 MBytes  17.2 Mbits/sec
[  5]  57.00-58.00  sec  2.05 MBytes  17.2 Mbits/sec
[  5]  58.00-59.00  sec  2.05 MBytes  17.2 Mbits/sec
[  5]  59.00-60.00  sec  2.08 MBytes  17.4 Mbits/sec
[  5]  60.00-60.09  sec   168 KBytes  16.2 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -

[ ID] Interval           Transfer     Bandwidth       Retr

[  5]   0.00-60.09  sec   125 MBytes  17.4 Mbits/sec   81             sender

[  5]   0.00-60.09  sec   124 MBytes  17.3 Mbits/sec                  receiver
```

**Resultado:** Flujos lanzados vía TCP. Se determina la capacidad máxima (Ancho de banda) del sistema según se muestra en la Figura 5.5

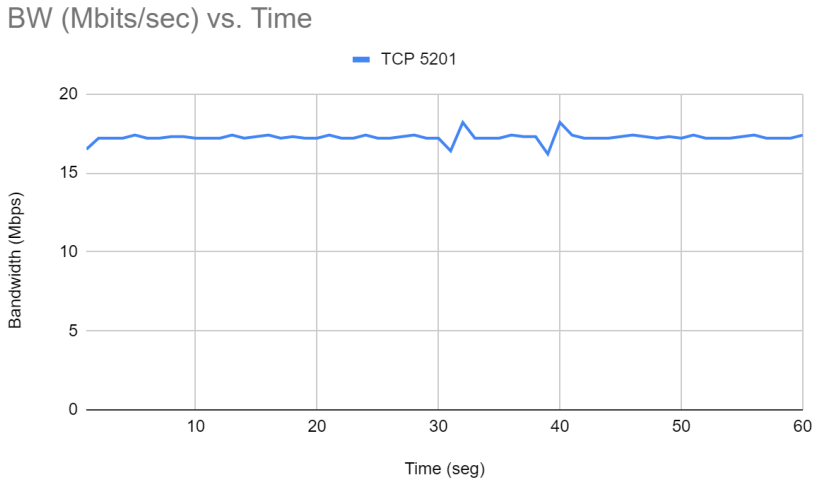



Figura 5.5. Prueba de rendimiento de ancho de banda. Fuente propia

## Pruebas con Iperf3 Flujos dobles sin Calidad de Servicio (QoS) sobre GNS3 - (Redes Tradicionales – Protocolo OSPF)

Se realizó el envío de dos flujos desde VM1 a VM3 y desde VM2 a VM3: el primero representado por tráfico UDP [80] por el puerto 5060 y el segundo como tráfico TCP a través del puerto 5201, respectivamente. De acuerdo al protocolo OSPF, la ruta elegida es, dentro del "Área 0", la misma que en el caso anterior: PE1 – P4 – P5 – PE2

Se explicita, a continuación, el envío de paquetes UDP generado en Iperf desde VM1 a VM3, el cual lo recibe a través del puerto 5060:

```
VM1: iperf3 -c 170.0.20.2 -p 5060 -t 60 -u -b

-----------------------------------------------------------

Server listening on 5060

-----------------------------------------------------------

Accepted connection from 170.0.10.2, port 48294

[  5] local 170.0.20.2 port 5060 connected to 170.0.10.2 port 48294

[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams

[  5]   0.00-1.00   sec   840 KBytes  6.88 Mbits/sec  7.074 ms  50/155 (32%)
[  5]   1.00-2.00   sec  1.02 MBytes  8.52 Mbits/sec  7.376 ms  53/183 (29%)
[  5]   2.00-3.00   sec  1.01 MBytes  8.46 Mbits/sec  6.915 ms  55/184 (30%)
[  5]   3.00-4.00   sec   992 KBytes  8.13 Mbits/sec  8.013 ms  58/182 (32%)
[  5]   4.00-5.00   sec  1.04 MBytes  8.72 Mbits/sec  6.878 ms  51/184 (28%)
[  5]   5.00-6.00   sec  1008 KBytes  8.26 Mbits/sec  7.896 ms  56/182 (31%)
[  5]   6.00-7.00   sec  1.02 MBytes  8.52 Mbits/sec  7.163 ms  56/186 (30%)
[  5]   7.00-8.00   sec  1008 KBytes  8.26 Mbits/sec  6.983 ms  57/183 (31%)
[  5]   8.00-9.00   sec  1008 KBytes  8.25 Mbits/sec  6.737 ms  56/182 (31%)
[  5]   9.00-10.00  sec   992 KBytes  8.14 Mbits/sec  6.915 ms  60/184 (33%)
[  5]  10.00-11.00  sec  1008 KBytes  8.26 Mbits/sec  7.394 ms  56/182 (31%)
[  5]  11.00-12.00  sec  1000 KBytes  8.19 Mbits/sec  6.748 ms  57/182 (31%)
[  5]  12.00-13.00  sec  1016 KBytes  8.32 Mbits/sec  7.790 ms  55/182 (30%)
[  5]  13.00-14.00  sec  1.22 MBytes  10.2 Mbits/sec  7.156 ms  29/185 (16%)
[  5]  14.00-15.00  sec  1.16 MBytes  9.70 Mbits/sec  7.826 ms  35/183 (19%)
[  5]  15.00-16.00  sec  1.11 MBytes  9.31 Mbits/sec  7.750 ms  40/182 (22%)
[  5]  16.00-17.00  sec  1.09 MBytes  9.11 Mbits/sec  7.473 ms  45/184 (24%)
……….. .
[  5]  48.00-49.00  sec  1008 KBytes  8.26 Mbits/sec  7.001 ms  57/183 (31%)
[  5]  49.00-50.00  sec  1000 KBytes  8.19 Mbits/sec  7.774 ms  56/181 (31%)
[  5]  50.00-51.00  sec   992 KBytes  8.11 Mbits/sec  7.226 ms  60/184 (33%)
[  5]  51.00-52.00  sec  1.01 MBytes  8.46 Mbits/sec  7.841 ms  56/185 (30%)
[  5]  52.00-53.00  sec   960 KBytes  7.87 Mbits/sec  7.511 ms  63/183 (34%)
[  5]  53.00-54.00  sec   536 KBytes  4.39 Mbits/sec  9.731 ms  116/183 (63%)
[  5]  54.00-55.00  sec  1.02 MBytes  8.59 Mbits/sec  6.751 ms  51/182 (28%)
[  5]  55.00-56.38  sec   880 KBytes  5.21 Mbits/sec  40.043 ms  51/161 (32%)
[  5]  56.38-57.00  sec   736 KBytes  9.78 Mbits/sec  7.083 ms  113/205 (55%)
[  5]  57.00-58.00  sec  1.03 MBytes  8.65 Mbits/sec  6.894 ms  51/183 (28%)
[  5]  58.00-59.00  sec  1008 KBytes  8.26 Mbits/sec  7.960 ms  57/183 (31%)
[  5]  59.00-60.00  sec  1.07 MBytes  8.98 Mbits/sec  6.780 ms  48/185 (26%)
[  5]  60.00-60.06  sec  16.0 KBytes  2.36 Mbits/sec  6.590 ms  0/2 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -

[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams

[  5]   0.00-60.06  sec  85.7 MBytes  12.0 Mbits/sec  6.590 ms  3282/10962 (30%)
```

Se explicita, a continuación, el envío de flujo TCP generado en Iperf desde VM2 a VM3, el cual lo recibe a través del puerto 5201:

```
VM2: iperf3 -c 170.0.20.2 -p 5201 -t 60


-----------------------------------------------------------

Server listening on 5201

-----------------------------------------------------------

Accepted connection from 170.0.10.3, port 45102

[  5] local 170.0.20.2 port 5201 connected to 170.0.10.3 port 45102

[ ID] Interval           Transfer     Bandwidth

[  5]   0.00-1.00   sec   895 KBytes  7.33 Mbits/sec
[  5]   1.00-2.00   sec   713 KBytes  5.84 Mbits/sec
[  5]   2.00-3.00   sec   795 KBytes  6.50 Mbits/sec
[  5]   3.00-4.00   sec   803 KBytes  6.59 Mbits/sec
[  5]   4.00-5.00   sec   820 KBytes  6.71 Mbits/sec
[  5]   5.00-6.00   sec   817 KBytes  6.70 Mbits/sec
……
[  5]  39.00-40.00  sec   785 KBytes  6.43 Mbits/sec
[  5]  40.00-41.00  sec   781 KBytes  6.39 Mbits/sec
[  5]  41.00-42.00  sec   863 KBytes  7.07 Mbits/sec
[  5]  42.00-43.00  sec   853 KBytes  6.99 Mbits/sec
[  5]  43.00-44.00  sec   758 KBytes  6.21 Mbits/sec
[  5]  44.00-45.00  sec   522 KBytes  4.27 Mbits/sec
[  5]  45.00-46.00  sec   148 KBytes  1.22 Mbits/sec
[  5]  46.00-47.00  sec   696 KBytes  5.70 Mbits/sec
[  5]  47.00-48.00  sec   748 KBytes  6.13 Mbits/sec
[  5]  48.00-49.00  sec   740 KBytes  6.04 Mbits/sec
[  5]  49.00-50.00  sec   782 KBytes  6.42 Mbits/sec
[  5]  50.00-51.00  sec   830 KBytes  6.80 Mbits/sec
[  5]  51.00-52.00  sec   802 KBytes  6.58 Mbits/sec
[  5]  52.00-53.00  sec   738 KBytes  6.04 Mbits/sec
[  5]  53.00-54.00  sec   407 KBytes  3.34 Mbits/sec
[  5]  54.00-55.00  sec   836 KBytes  6.84 Mbits/sec
[  5]  55.00-56.01  sec   769 KBytes  6.25 Mbits/sec
[  5]  56.01-57.00  sec   469 KBytes  3.88 Mbits/sec
[  5]  57.00-58.00  sec   731 KBytes  5.99 Mbits/sec
[  5]  58.00-59.00  sec   775 KBytes  6.34 Mbits/sec
[  5]  59.00-60.00  sec   798 KBytes  6.54 Mbits/sec
[  5]  60.00-60.04  sec  35.4 KBytes  6.51 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -

[ ID] Interval           Transfer     Bandwidth       Retr

[  5]   0.00-60.04  sec  42.9 MBytes  5.99 Mbits/sec  148             sender

[  5]   0.00-60.04  sec  42.7 MBytes  5.96 Mbits/sec                  receiver
```

**Resultado:** Ambos flujos fueron enviados simultáneamente. Ya que uno de los flujos enviados fue a través de protocolo UDP y el otro por TCP, el manejo de paquetes se realizó bajo modalidad best-effort (mejor esfuerzo) en uno de los casos, mientras que en el otro se trata de un protocolo orientado a conexión (TCP). Dado que UDP se uti-

liza para los envíos en tiempo real se eligió como premisa, para el próximo caso, priorizar estos flujos (UDP) frente a los enviados a través de TCP, aún cuando se reconoce que algunas aplicaciones sobre TCP no tendrán un tratamiento best effort. Esto se verá en el siguiente caso con Iperf3 sobre redes tradicionales con Calidad de Servicio.

Así, se aprecia una diferencia en el ancho de banda utilizado por cada uno de los flujos dadas las características propias de cada uno de ellos. En el caso del flujo UDP, intenta alcanzar el máximo ancho de banda permitido por el sistema, aunque claramente se ve afectado por el protocolo TCP (orientado a conexión) que limita el alcance de dicho máximo para UDP.(Figura 5.6)

Por otra parte, el porcentaje de paquetes perdidos para el flujo bajo protocolo UDP, alcanza picos de 60%, lo que indica la ausencia de la aplicación de políticas de calidad de servicio (QoS) (Figura 5.7)

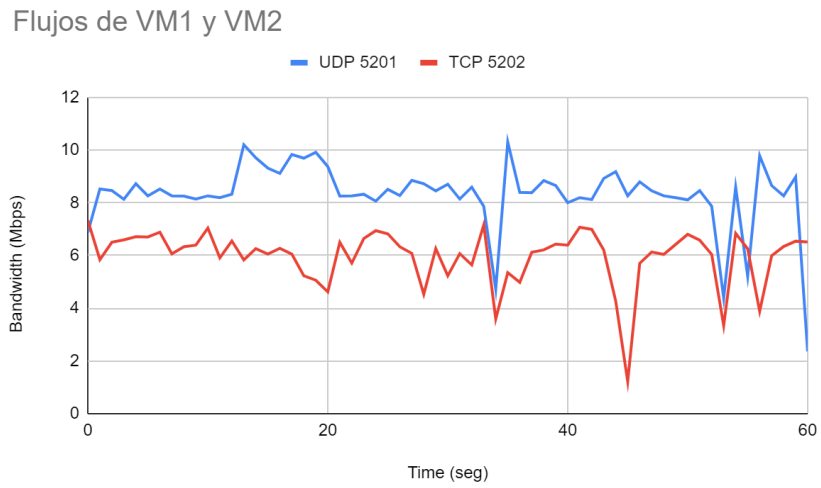



Figura 5.6. Ancho de banda utilizado por flujos UDP y TCP respectivamente
Fuente propia

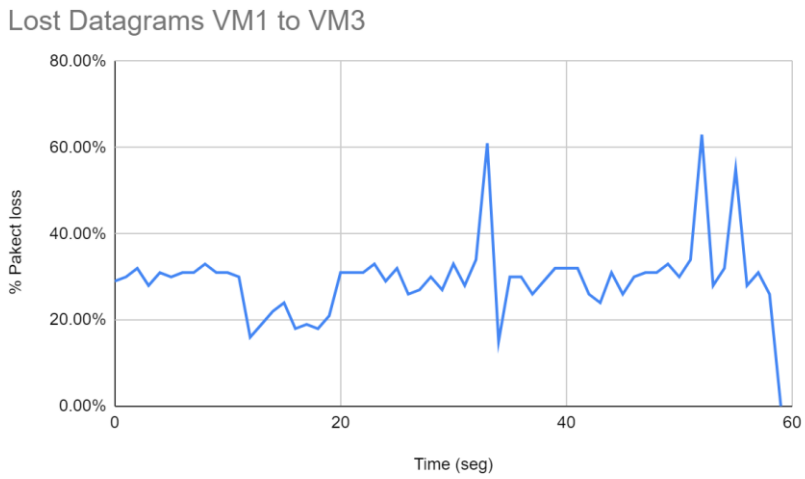



Figura 5.7. Pérdida de paquetes en el flujo bajo protocolo UDP. Fuente propia

## Pruebas con Iperf3 Flujos dobles aplicando políticas de Calidad de Servicio (QoS) sobre GNS3 - (Redes Tradicionales – Protocolo OSPF)

Se observó en las pruebas de envío de flujos UDP y TCP sin Calidad de Servicio, una importante pérdida de paquetes en la comunicación, considerando especialmente el flujo enviado a través de protocolo UDP. En la siguiente configuración se probará la corrección de estas pérdidas, aplicando políticas de Calidad mediante Servicios Diferenciados, bajo la política de encolamiento elegida LLQ y CBWFQ.

En esta prueba proporcionó un servicio de entrega preferencial para las aplicaciones que lo necesitan asegurando un ancho de banda suficiente, controlando la latencia y reduciendo la pérdida de datos, según lo explicado anteriormente (pág 12) de este mismo documento.

En la parametrización de toda la topología, para los flujos UDP con puerto destino 5060, se consideró que dichos flujos UDP, contienen en su carga útil VoIP donde es necesario aplicar el marcado DiffServ (QoS). También se realizó la aplicación de idéntico marcado para el flujo TCP, a través del puerto 5201. La configuración de los routers de borde, en particular sobre el manejo de las entradas y salidas de encolamiento de todos los flujos, se realizó de acuerdo a estas políticas de Calidad de Servicio.

La prueba de los dos flujos simultáneos se realizó, como en el caso anterior, desde VM1 a VM3 y desde VM2 a VM3: el primero sobre protocolo UDP, puerto 5060 y el segundo sobre protocolo TCP sobre puerto 5201, respectivamente.

Se explicita, a continuación, el envío de paquetes UDP generado en Iperf desde VM1 a VM3, el cual lo recibe a través del puerto 5060:

```
VM1: iperf3 -c 170.0.20.2 -p 5060 -t 60 -u -b

-----------------------------------------------------------

Server listening on 5060

-----------------------------------------------------------

Accepted connection from 170.0.10.2, port 48296

[  5] local 170.0.20.2 port 5060 connected to 170.0.10.2 port 48296
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   720 KBytes  5.90 Mbits/sec  8.962 ms  3/93 (3.2%)
[  5]   1.00-2.00   sec   840 KBytes  6.87 Mbits/sec  7.920 ms  0/105 (0%)
[  5]   2.00-3.00   sec   864 KBytes  7.08 Mbits/sec  7.056 ms  0/108 (0%)
[  5]   3.00-4.00   sec   832 KBytes  6.81 Mbits/sec  7.840 ms  0/104 (0%)
[  5]   4.00-5.00   sec   872 KBytes  7.15 Mbits/sec  6.913 ms  1/110 (0.91%)
[  5]   5.00-6.00   sec   848 KBytes  6.95 Mbits/sec  8.062 ms  1/107 (0.93%)
[  5]   6.00-7.00   sec   840 KBytes  6.88 Mbits/sec  8.250 ms  0/105 (0%)
[  5]   7.00-8.00   sec   872 KBytes  7.15 Mbits/sec  8.600 ms  0/109 (0%)
[  5]   8.00-9.00   sec   848 KBytes  6.94 Mbits/sec  7.942 ms  0/106 (0%)
[  5]   9.00-10.00  sec   848 KBytes  6.95 Mbits/sec  8.632 ms  0/106 (0%)
……..

ospf[  5]  53.00-54.00  sec   840 KBytes  6.88 Mbits/sec  7.243 ms  0/105 (0%)
[  5]  54.00-55.00  sec   848 KBytes  6.95 Mbits/sec  8.853 ms  1/107 (0.93%)
[  5]  55.00-56.00  sec   840 KBytes  6.87 Mbits/sec  7.931 ms  2/107 (1.9%)
[  5]  56.00-57.00  sec   832 KBytes  6.82 Mbits/sec  9.011 ms  3/107 (2.8%)
[  5]  57.00-58.00  sec   840 KBytes  6.88 Mbits/sec  7.532 ms  1/106 (0.94%)
[  5]  58.00-59.00  sec   848 KBytes  6.95 Mbits/sec  8.901 ms  0/106 (0%)
[  5]  59.00-60.00  sec   832 KBytes  6.80 Mbits/sec  6.157 ms  9/113 (8%)
[  5]  60.00-60.06  sec  8.00 KBytes  1.23 Mbits/sec  6.431 ms  0/1 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-60.06  sec  50.0 MBytes  6.98 Mbits/sec  6.431 ms  140/6399 (2.2%)
```

Se explicita, a continuación, el envío de flujo TCP generado en Iperf desde VM2 a VM3, el cual lo recibe a través del puerto 5201:

```
VM2: iperf3 -c 170.0.20.2 -p 5201 -t 60


-----------------------------------------------------------

Server listening on 5201

-----------------------------------------------------------

Accepted connection from 170.0.10.3, port 45110

[  5] local 170.0.20.2 port 5201 connected to 170.0.10.3 port 45110
```

```
[ ID] Interval           Transfer     Bandwidth

[  5]   0.00-1.00   sec  1.50 MBytes  12.6 Mbits/sec
[  5]   1.00-2.00   sec  1.21 MBytes  10.1 Mbits/sec
[  5]   2.00-3.01   sec  1.17 MBytes  9.73 Mbits/sec
[  5]   3.01-4.00   sec  1.17 MBytes  9.87 Mbits/sec
[  5]   4.00-5.00   sec  1.17 MBytes  9.80 Mbits/sec
[  5]   5.00-6.00   sec  1.17 MBytes  9.80 Mbits/sec
[  5]   6.00-7.00   sec  1.19 MBytes  10.0 Mbits/sec
[  5]   7.00-8.00   sec  1.18 MBytes  9.87 Mbits/sec
[  5]   8.00-9.00   sec  1.17 MBytes  9.80 Mbits/sec
[  5]   9.00-10.00  sec  1.19 MBytes  9.99 Mbits/sec
[  5]  10.00-11.00  sec  1.17 MBytes  9.80 Mbits/sec
[  5]  11.00-12.00  sec  1.19 MBytes  9.98 Mbits/sec
[  5]  12.00-13.01  sec  1.18 MBytes  9.81 Mbits/sec
.....
[  5]  56.00-57.00  sec  1.19 MBytes  9.97 Mbits/sec
[  5]  57.00-58.00  sec  1.17 MBytes  9.76 Mbits/sec
[  5]  58.00-59.00  sec  1.20 MBytes  10.1 Mbits/sec
[  5]  59.00-60.00  sec  1.18 MBytes  9.89 Mbits/sec
[  5]  60.00-60.08  sec  79.2 KBytes  7.88 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -

[ ID] Interval           Transfer     Bandwidth       Retr

[  5]   0.00-60.08  sec  70.2 MBytes  9.81 Mbits/sec   52             sender

[  5]   0.00-60.08  sec  70.0 MBytes  9.77 Mbits/sec                  receiver
```

**Resultado:** En los flujos lanzados simultáneamente, bajo la política de calidad de servicio con servicios diferenciados se aprecia, a comparación del test anterior, como se regula el flujo TCP y, además, el flujo bajo protocolo UDP evidencia una recuperación importante en la pérdida de paquetes. Drásticamente, se observa una disminución de 40% promedio a un porcentaje de 4%. Esto se observa en la Figura 5.9

También se aprecia que, al asignar 10 Mbps al flujo TCP, los restantes 7 Mbps los aprovecha en su totalidad, la comunicación vía UDP. (Figura 5.8)

NOTA: Los 17 Mbps totales, son los asignados por GNS3 y se encuentran parametrizados.

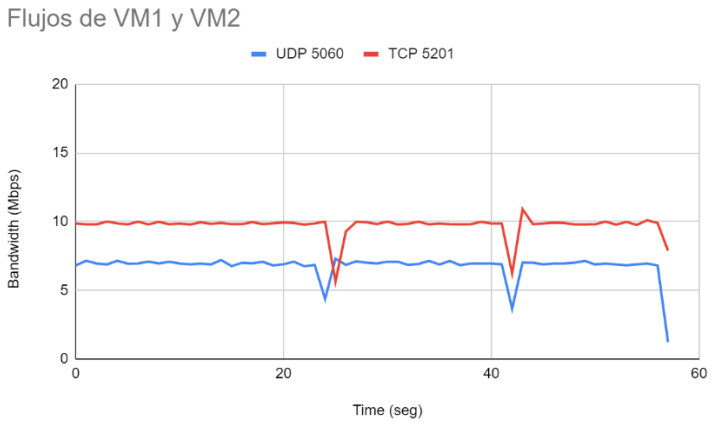


Figura 5.8. Ancho de banda utilizado por flujos UDP y TCP bajo políticas DiffServ Fuente propia

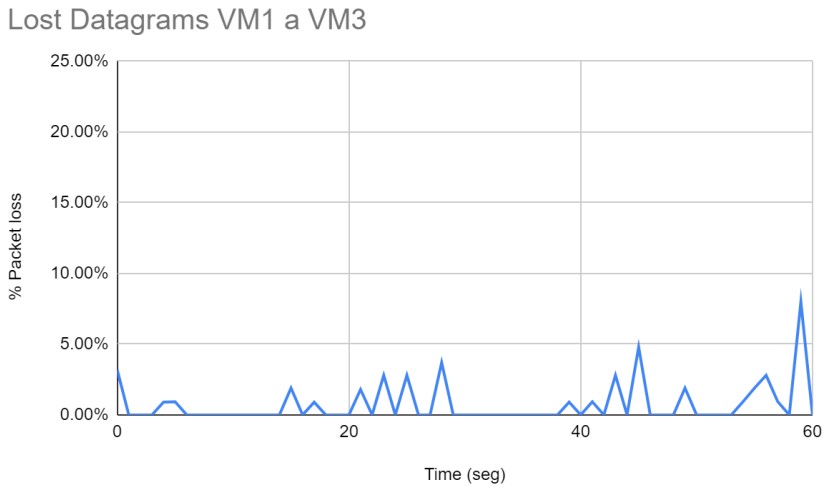



Figura 5.9. Pérdida de paquetes en el flujo bajo protocolo UDP bajo políticas Diff-Serv. Fuente propia

## Observaciones sobre otros protocolos de enrutamiento

1. RIP (Routing Information Protocol) RFC 2453 [81]

En el caso de RIP (Routing Information Protocol), el dispositivo correspondiente enviará su tabla de enrutamiento completa a todos los vecinos conectados cada un intervalo de tiempo que oscila alrededor de los 30 segundos. Por otra parte, las actualizaciones enviadas por eventos alterarán dicho intervalo si, por ejemplo, una interfaz cae antes de que expire el clock de 30 segundos mencionado anteriormente.

Por ser un protocolo basado en el concepto de vector distancia, es sensible a la aparición de bucles de enrutamiento. Esto es consecuencia de la inexistencia de relaciones de vecindad o recálculos de la topología de la red, como ocurre con otros protocolos, como el caso de aquellos basados en estados de enlace, como es el caso de OSPF. Esto afecta directamente la calidad de la información de enrutamiento que proporciona RIP.

Por lo tanto, en el presente trabajo, se ha omitido su utilización, ya que se busca protocolos que permitan establecer la topología de la red y RIP no colecta información de los enlaces. Simplemente se obtiene información de la interfaz por la cual se deberá realizar el envío de la información.

2. IS-IS (Intermediate System to Intermediate System) RFC 3787 [82]

Nuevamente, se justifica el uso de OSPF. Este es compatible con NBMA (No Broadcast MultiAccess Network) y enlaces punto a multipunto, mientras que IS-IS no es compatible. Por otra parte, OSPF admite el enlace virtual, mientras que IS-IS no lo admite. Además, OSPF elige un DR (Designated Router)y un BDR (Backup Designated Router), mientras que IS-IS elige un solo DR llamado DIS (Designated IS).

Otra diferencia a favor de OSPF, radica en el hecho de que, mientras un enrutador OSPF puede pertenecer a múltiples áreas, un enrutador IS-IS puede pertenecer a una y solamente una área.

Por último, OSPF usa la ID del enrutador, mientras que IS-IS usa el ID del sistema para identificar cada enrutador en la red. Además, ambos son protocolos de estado de enlace y ambos usan el algoritmo Dijkstra para calcular la mejor ruta a través de una red. [83]

3. CSPF (Constrained Shortest Path First)[84]

El algoritmo Restringido de la ruta más corta primero CSPF (Constrained Shortest Path First) se usa con protocolos de enrutamiento de estado de enlace como OSPF e ISIS. Resuelve las consultas de enrutamiento de calidad de servicio, encontrando la mejor ruta (a una dirección de destino IPv4 o IPv6) que cumpla las restricciones especificadas, como, por ejemplo, un ancho de banda mínimo.[85]

En el análisis del trabajo realizado, se observa que agregar restricciones en el cálculo de la ruta, no modificaría los resultados obtenidos aplicando OSPF. Por tal motivo, fue omitida su aplicación.

## Pruebas sobre redes SDN

### Pruebas con Iperf3 Flujo Simple sobre MININET - (Redes SDN – Reglas de cantidad mínima de saltos)

Para la primera prueba, se envió un primer flujo bajo protocolo TCP por el puerto 5201 desde H1 a H3, por la ruta s1, s2, s3, s4 y s7. Dicho test, al igual que el realizado

anteriormente (ver página 53, en este documento) y para sostener la misma configuración que la realizada sobre redes tradicionales, se ha determinado la capacidad del sistema a 17Mbits/sec aproximadamente. Lo que si fue modificado en esta configuración, fue la ruta correspondiente s1, s2, s3, s4 y s7, independizándose del protocolo OSPF. A continuación, se muestra el escenario y la limitación de interfaces:

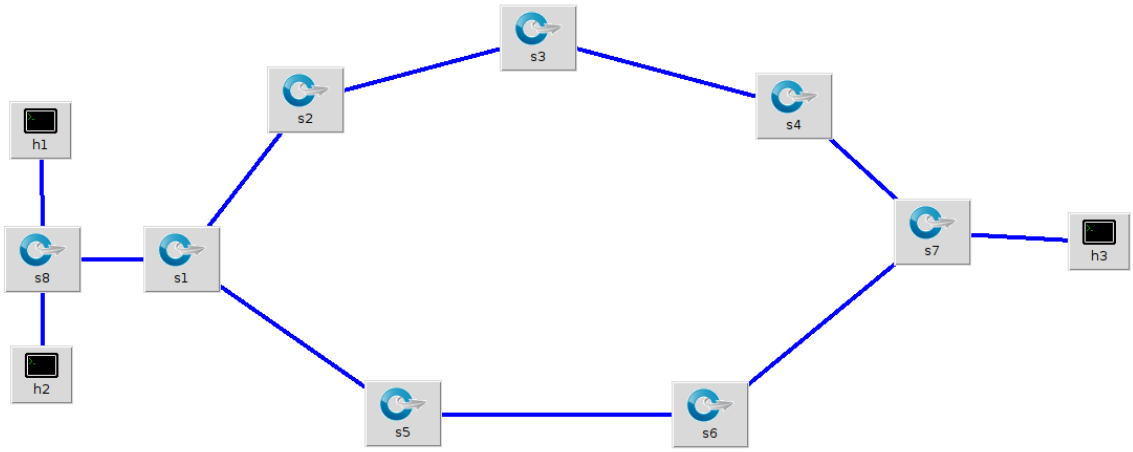



Figura 5.10. Escenario topológico SDN. Fuente propia.

Interfaces limitadas a 17Mbps, según se muestra a continuación

```
for i in {2..6}
do
tc qdisc add dev s$i-eth1 root tbf rate 17Mbit latency 50ms burst 1540
tc qdisc add dev s$i-eth2 root tbf rate 17Mbit latency 50ms burst 1540
done

for i in {1,7,8}
do
tc qdisc add dev s$i-eth1 root tbf rate 17Mbit latency 50ms burst 1540
tc qdisc add dev s$i-eth2 root tbf rate 17Mbit latency 50ms burst 1540
tc qdisc add dev s$i-eth3 root tbf rate 17Mbit latency 50ms burst 1540
done
```

Como en el caso anterior, el límite del ancho de banda será determinante para realizar las siguientes pruebas de rendimiento para el caso de SDN. Se establecen mediante reglas:

Establecimiento de reglas

Reglas con prioridad 99.

Decremento del ttl, modificar dirección ethernet, y salir por interface.

```
#s1
ovs-ofctl -Oopenflow13 add-flow s1 priority=99,eth,ip,in_port=s1-eth1,actions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:12,
                                                mod_dl_dst:00:00:00:00:00:21,
                                                output:2'


ovs-ofctl -Oopenflow13 add-flow s1 priority=99,eth,ip,in_port=s1-eth2,actions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:11,
                                                mod_dl_dst:00:00:00:00:00:81,
                                                output:1'


#s2
ovs-ofctl -Oopenflow13 add-flow s2 priority=99,eth,ip,in_port=s2-eth1,actions='dec_ttl,
                                              mod_dl_src:00:00:00:00:00:22,
                                              mod_dl_dst:00:00:00:00:00:31,
                                              output:2'


ovs-ofctl -Oopenflow13 add-flow s2 priority=99,eth,ip,in_port=s2-eth2,actions='dec_ttl,
                                              mod_dl_src:00:00:00:00:00:21,
                                              mod_dl_dst:00:00:00:00:00:11,
                                              output:1'


……..

# s8
ovs-ofctl -Oopenflow13 add-flow s8 priority=99,eth,ip,nw_dst=172.0.20.2,actions='dec_ttl,
                                            mod_dl_src:00:00:00:00:00:81,
                                            mod_dl_dst:00:00:00:00:00:11,
                                            output:1'


ovs-ofctl -Oopenflow13 add-flow s8 priority=99,eth,ip,in_port=1,nw_dst=172.0.10.2,actions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:82,
                                                mod_dl_dst:00:00:00:00:00:01,
                                                output:2'


ovs-ofctl -Oopenflow13 add-flow s8 priority=99,eth,ip,in_port=1,nw_dst=172.0.10.3,actions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:83,
                                                mod_dl_dst:00:00:00:00:00:02,
                                                output:3'


ovs-ofctl -Oopenflow13 add-flow s8 priority=80,actions=NORMAL
```

**Resultados:** Luego de la realización de esta prueba, se observa que el ancho de banda en función del tiempo, se mantiene en los límites establecidos por las reglas previamente enunciadas y parametrizadas, según Figura 5.11

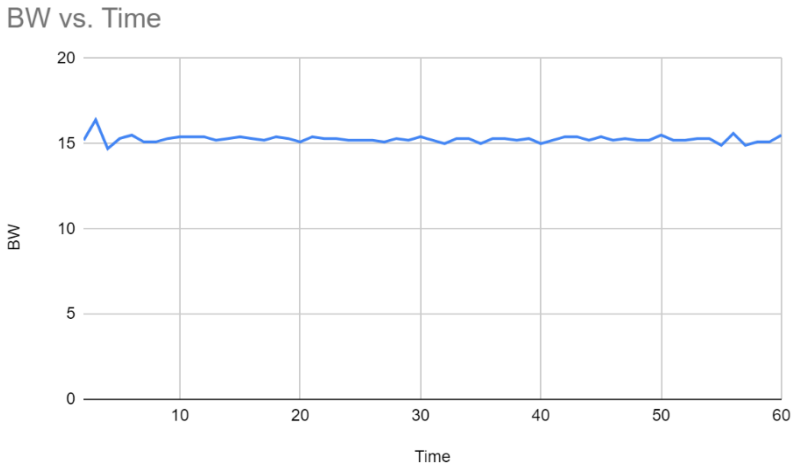



Figura 5.11. Ancho de banda según el tiempo transcurrido de transmisión. Fuente propia.

### Pruebas con Iperf3 Flujo doble sobre MININET - (Redes SDN – Envío por ruta más larga)

Se realizó el envío de dos flujos desde H1 a H3 y desde H2 a H3: el primero representado por tráfico UDP por el puerto 5201 y el segundo como tráfico TCP a través del puerto 5202, respectivamente. Se utilizó la misma ruta que en el caso anterior: s1, s2, s3, s4 y s7.

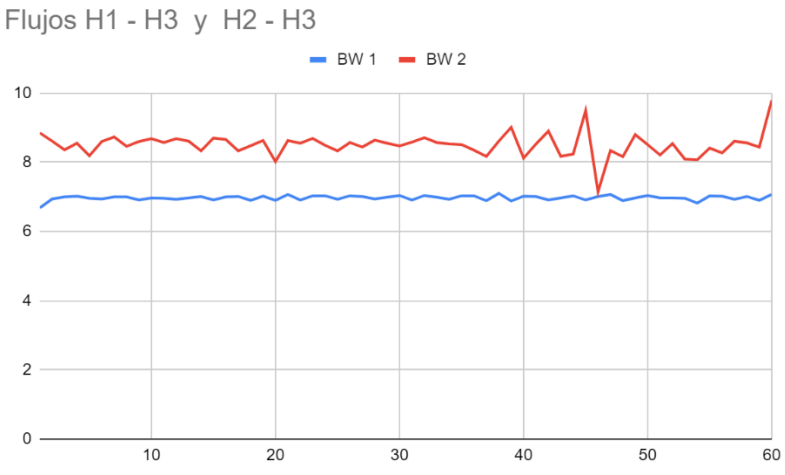



Figura 5.12. Ancho de banda según el tiempo transcurrido de transmisión (Azul H1-H3, tráfico UDP, Rojo H2 -H3, tráfico TCP). Fuente propia.

Se analiza a continuación, la cantidad de paquetes perdidos:

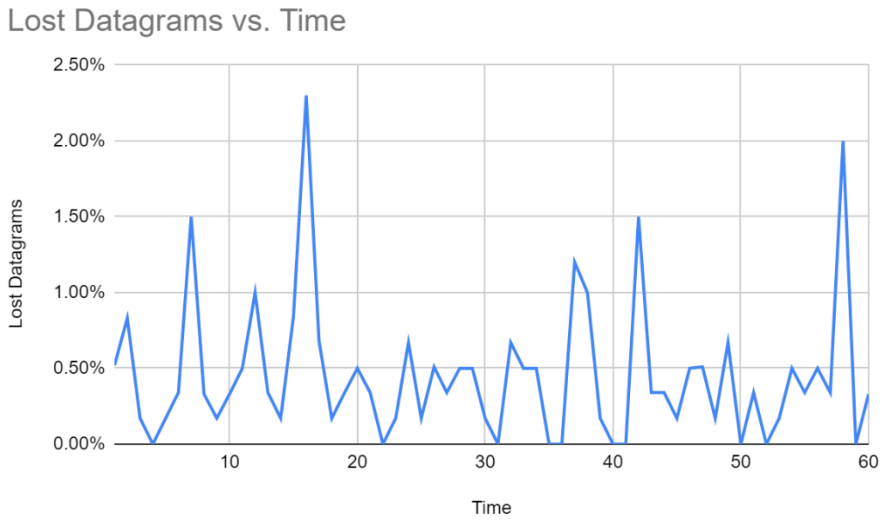



Figura 5.13. Cantidad de paquetes perdidos sobre el flujo bajo protocolo UDP.. Fuente propia.

**Resultado:** La cantidad de paquetes perdidos sobre el flujo bajo protocolo UDP, según Figura 5.13, es considerablemente inferior a la observada en la Figura 5.7, a pesar de que, en este caso, los dos flujos fueron enviados por la ruta más larga.

### Pruebas con Iperf3 Flujo doble sobre MININET - (Redes SDN – Reglas de cantidad mínima de saltos para flujo UDP y envío de flujo TCP por la ruta más larga)

Para este caso, se dividieron los dos flujos, enviándose el flujo bajo protocolo TCP por la ruta s1, s2, s3, s4 y s7 a través del puerto 5202 mientras que para el flujo bajo protocolo UDP, se utilizó la ruta s1, s5, s6 y s7 y el puerto 5201.

Establecimiento de Flujos

```
Flujos prioridad 100 en los switches 1 y 7 para matchear según el puerto tcp/udp
El flujo tcp 5202 toma el camino de arriba (s1, s2, s3, s4, s7)
El flujo udp 5201 toma el camino de abajo (s1, s5, s6, s7)

ovs-ofctl -Oopenflow13 add-flow s1 priority=100,eth,ip,tcp,in_port=s1-eth1,tcp_dst=5202,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:12,
                                                mod_dl_dst:00:00:00:00:00:21,
                                                output:2'
ovs-ofctl -Oopenflow13 add-flow s1 priority=100,eth,ip,tcp,in_port=s1-eth2,tcp_src=5202,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:11,
                                                mod_dl_dst:00:00:00:00:00:81,
                                                output:1'
# 5201 abajo
ovs-ofctl -Oopenflow13 add-flow s1 priority=100,eth,ip,udp,in_port=s1-eth1,udp_dst=5201,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:13,
```

```
                                                    mod_dl_dst:00:00:00:00:00:51,
                                                    output:3'
ovs-ofctl -Oopenflow13 add-flow s1 priority=100,eth,ip,udp,in_port=s1-eth3,udp_src=5201,ac-
tions='dec_ttl,
                                                    mod_dl_src:00:00:00:00:00:81,
                                                    mod_dl_dst:00:00:00:00:00:11,
                                                    output:1'

# s7
# 5202 arriba
ovs-ofctl -Oopenflow13 add-flow s7 priority=100,eth,ip,tcp,in_port=s7-eth2,tcp_dst=5202,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:71,
                                                mod_dl_dst:00:00:00:00:00:03,
                                                output:1'

ovs-ofctl -Oopenflow13 add-flow s7 priority=100,eth,ip,tcp,in_port=s7-eth1,tcp_src=5202,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:72,
                                                mod_dl_dst:00:00:00:00:00:42,
                                                output:2'

# 5201 abajo
ovs-ofctl -Oopenflow13 add-flow s7 priority=100,eth,ip,udp,in_port=s7-eth3,udp_dst=5201,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:71,
                                                mod_dl_dst:00:00:00:00:00:03,
                                                output:1'

ovs-ofctl -Oopenflow13 add-flow s7 priority=100,eth,ip,udp,in_port=s7-eth1,udp_src=5201,ac-
tions='dec_ttl,
                                                mod_dl_src:00:00:00:00:00:73,
                                                mod_dl_dst:00:00:00:00:00:62,
                                                output:3'
```

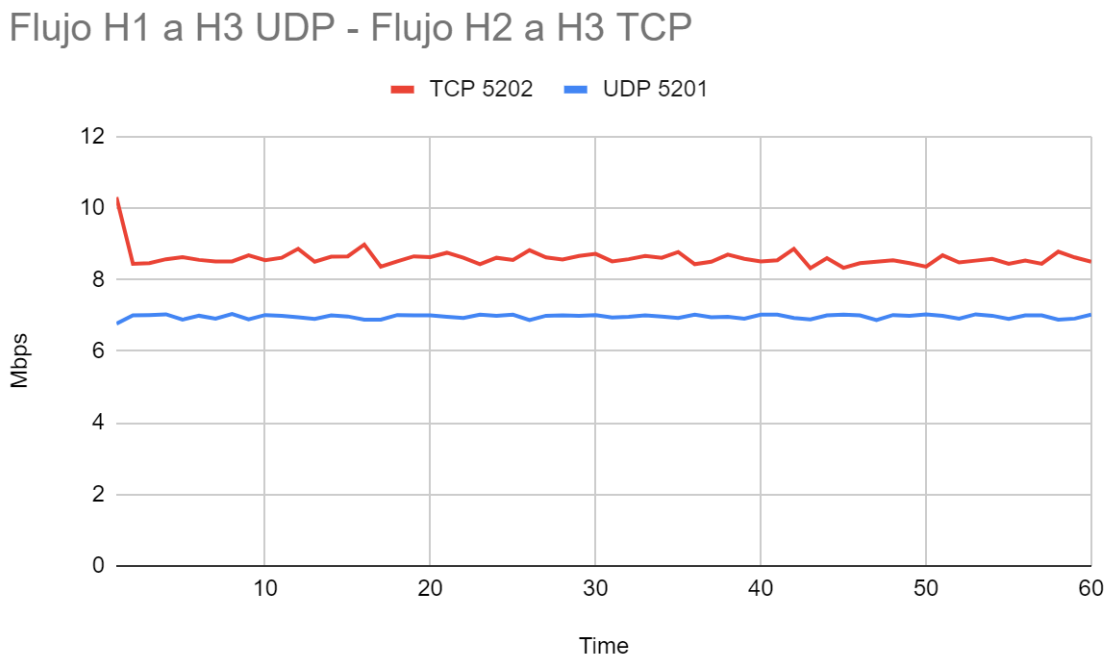


Figura 5.14. Ancho de banda según el tiempo transcurrido de transmisión (Azul H1-H3, tráfico UDP, Rojo H2 -H3, tráfico TCP). Fuente propia.

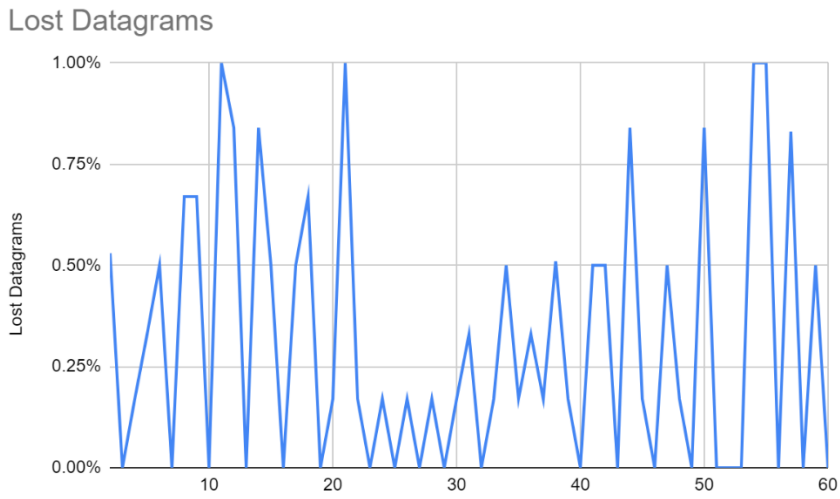



Figura 5.15. Paquetes perdidos para el flujo bajo protocolo UDP. Fuente propia

**Resultado:** En la figura 5-14 se observa que el ancho de banda no se encuentra afectado en ninguno de los dos casos, Además, en la Figura 5-15, la pérdida de paquetes para el flujo bajo protocolo UDP se reduce drásticamente al 1%.

Por otra parte, la infraestructura de red heterogénea aumenta la complejidad de las redes y plantea una serie de desafíos para organizar, administrar y optimizar los recursos de la red de manera efectiva. Implementar más inteligencia en las redes es una forma posible de resolver estos problemas. Hace unos años, se propuso un enfoque de Knowledge Plane (KP) [86] para llevar la automatización, la recomendación y la inteligencia a Internet, mediante la aplicación de Machine Learning (ML) y técnicas cognitivas. Sin embargo, el KP no ha sido prototipado o implementado hasta el momento de escribir este trabajo. Una de las principales razones es la característica inherentemente distribuida de los sistemas de red tradicionales, donde cada nodo, como enrutador o conmutador, solo puede ver y actuar sobre una pequeña porción del sistema. Aprender de los nodos que solo tienen una pequeña vista parcial del sistema completo para realizar el control más allá del dominio local es muy complejo. Se espera entonces, que los próximos avances en redes definidas por software (SDN) faciliten la complejidad del aprendizaje. [87][88]

# Capítulo 6: Conclusiones

Para sustentar la calidad de servicio (QoS) extremo-extremo es importante profundizar en el comportamiento dinámico de las redes, lo que se hace a través de parámetros que pueden ser medidos y monitoreados. Entre los parámetros más importantes, que determinan si se cumple con el nivel de servicio que se ofrece, se encuentran, entre otros, el jitter y la pérdida de paquetes [89][90][91].

De acuerdo a estas consideraciones, en este trabajo se han encontrado conclusiones que confirman el objetivo planteado, inclusive simplificando los esquemas planteados en el comienzo.

Se detallan, entonces, las conclusiones mencionadas:

- Los protocolos de enrutamiento distribuidos basados en el destino no son una buena solución en la aplicación del enrutamiento centralizado puro (pizarra limpia), pero como mencionamos antes (pag. 60 del presente trabajo) estas redes deberán interconectarse con redes tradicionales que mantendrán protocolos de enrutamiento estandarizados que definirán caminos y formarán parte de la información almacenada en la RIB de nuestro controlador.

- La idea de un enrutamiento basado en la búsqueda de una ruta óptima parece dejar paso a un encaminamiento balanceado que cumpla con parámetros comprometidos de calidad de servicio en todos los flujos cursados a través de la red utilizando todos sus enlaces y los recursos disponibles.

- Las redes podrán adaptarse a los cambios que se produzcan utilizando métricas que permitan controlar el ancho de banda disponible, los paquetes perdi-

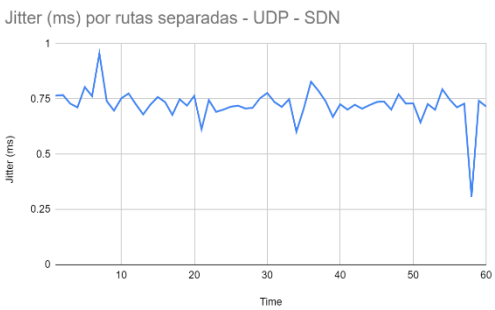
dos de los flujos priorizados o el jitter necesario establecido mediante aplicaciones que establezcan condiciones modificando dinámicamente las reglas en las tablas de flujo, este comportamiento de los flujos podrá programarse estáticamente o bien en forma dinámica siguiendo patrones de comportamiento.

- Las redes SDN poseen una característica intrínseca: sobre ellas se puede aplicar cualquier tipo de algoritmo, sin atarnos a protocolos de enrutamiento, como por ejemplo OSPF (aunque sí permiten implementar el algoritmo de Dijkstra, entre otros). Por lo tanto y según las reglas que se establezcan se pudieron alterar arbitrariamente tanto los algoritmos que ofrecen las librerías como la aplicación de uno propio, creado ad hoc. En el caso del presente trabajo, el envío por diferentes rutas favoreció el mejoramiento de la pérdida de paquetes en las redes SDN, mientras que no se obtuvieron los mismos resultados satisfactorios en los esquemas de redes tradicionales, inclusive aplicando políticas de Servicios Diferenciados.

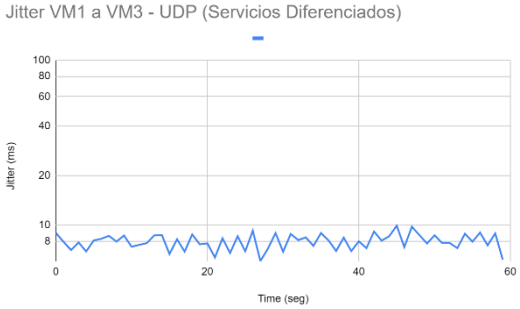
- En este trabajo, se evidencia que la clave de SDN es desarrollar redes de comunicaciones en las cuales se desacopla el plano de control de los elementos de hardware. El controlador asume y centraliza las funciones de control, de tal manera que se puede administrar el tráfico de la red sin tener que tocar los elementos individualmente, diferenciando a estas redes de las tradicionales y simplificando las tareas de administración de flujos.

- De acuerdo con unas pocas reglas en SDN se obtuvo un aprovechamiento eficaz del ancho de banda para cada flujo, juntos o separados. Esta tarea fue sencilla y no requirió la aplicación de protocolos específicos, tramo a tramo.

- Es de destacar el aumento de la performance en SDN con una simple división de tráficos. En redes tradicionales se aplicó OSPF y Servicios Diferenciados. Esto se evidencia en que la variación de retardo para SDN es de 0,75mseg promedio contra un promedio de 8 mseg para redes tradicionales.




a) Jitter en SDN – UDP




b) Jitter Redes tradicionales con aplicación de servicios diferenciados - UDP

- La pérdida de los paquetes disminuyó drásticamente, como se indicara anteriormente. En redes tradicionales, se redujo de 45% en promedio a 4% con la aplicación de Servicios Diferenciados. En el caso de SDN, para el tráfico por la ruta más larga el porcentaje fue de 2% disminuyendo el mismo a 1% cuando se dividieron los flujos por las sendas rutas.

- En el presente trabajo, se pone en evidencia la facilidad de modificar el camino de los flujos de la red y sus parámetros cuando de SDN se trata, claramente a través de las aplicaciones, lo que redunda en una eficaz re-distribución de los flujos.

- Es importante destacar la versatilidad de SDN que, con el tipo de control centralizado que ofrece, muestra claramente la facilidad en la administración y cambio de políticas de ancho de banda, las cuales pueden ser muy inteligentes y optimizables.

- La programabilidad de SDN permite que las soluciones de red se optimicen a través de modelos ML (Machine Learning) (por ejemplo, configuración y asignación de recursos, técnicas de aprendizaje automático para

mejorar el control del tráfico de la red, entre otros). Dicha programabilidad puede optimizarse mediante algoritmos de aprendizaje automático, los cuales pueden ser ejecutados sobre la red en tiempo real. Las técnicas de aprendizaje automático de última generación que se pueden desarrollar y aplicar en SDN están siendo construidas y analizadas al momento de escribir el presente trabajo. Esto implica que se encuentran en marcha investigaciones sobre técnicas de aprendizaje de máquinas para mejorar el rendimiento, la inteligencia y la eficiencia de SDN. [92][93]

- A partir de SDN, con su división de planos y su control centralizado se ha abierto la puerta a distintos conceptos que posibilitan la automatización del comportamiento de las redes y nos alejan de la compleja, difícil y poco eficaz administración de los mecanismos utilizados por las redes tradicionales. Ideas tales como las redes basadas en la intención (IBN o Intent Based Networks) [94]donde través del control del comportamiento de la red se puede cumplir con los objetivos comerciales generales de la organización. Al igual que SDN, IBN está creando software que controla la red como un todo, en lugar de dispositivo a dispositivo.

- Otro concepto importante es la centralización de las acciones a realizar en la red, en contra posición a la importancia en la configuración de los dispositivos en una red descentralizada. Las IBN verifican en tiempo real si se cumple la intención original ya que el sistema puede tomar medidas correctivas, como modificar una política de QoS, redistribuir las VLAN o alterar ACL para cumplir con los objetivos de control.

- Para finalizar, la necesidad de redes automatizadas y predecibles determina la utilización de nuevas herramientas, ya que las existentes parecen no permitir el descubrimiento de todas las interacciones entre el tráfico modelado y los obtenidos en los dispositivos reales de toda la red. Para esto, el sistema construiría modelos lógico-matemáticos de la red y luego verificaría que cumpla con la intención propuesta.

# Bibliografía

## Referencias


[1] OvS – OpenVswitch
https://www.openvswitch.org/
Accedido: 03-08-2019

[2] PROPUESTA PARA LA EVALUACIÓN DE PARÁMETROS DE QOS EN SDN - Ing. Cesar Alonso Irizar - Dr. C. Caridad Anías Calderon - Revista Telem@tica. Vol. 16. No. 2, mayo-agosto, 2017, p.12- 24 ISSN 1729-3804- Accedido: 24-07-2019

[3] ZDNet - What is SDN? How software-defined networking changed everything
By Scott Fulton – Mayo 22 – 2018
https://www.zdnet.com/article/software-defined-networking-101-what-sdn-is-and-where-its-going
Accedido: 15-01-2019

[4]RFC 4271 - A Border Gateway Protocol 4 (BGP-4)
https://tools.ietf.org/html/rfc4271
Accedido: 13-02-2020

[5]RFC 2453 - RIP Version 2
https://tools.ietf.org/html/rfc2453
Accedido: 13-02-2020

[6]Introduction to EIGRP
https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/13669-1.html
Accedido: 13-02-2020

[7] R. Hernández Sampieri, C. Fernandez Collado and P. Baptista Lucio, “Metodología de la investigación”, México: Mc Graw Hill, 2010.

[8] Universidad Internacional de Valencia: ¿En qué se diferencian SDN y NFV?https://www.universidadviu.com/sdn-nfv-que-los-diferencia/
Accedido: 29-07-2019

[9] ONOS Project
https://onosproject.org/ Accedido: 29-07-2019

[10] Open Day Light Project
https://www.opendaylight.org/ Accedido:29-07-2019

[11] VMware - NSX Data Center
https://www.vmware.com/latam/products/nsx.html
Accedido: 29-07-2019

[12] HUAWEI – Agile Controller
https://e.huawei.com/es/products/enterprise-networking/sdn-controller/agile-controller
Accedido: 29-07-2019

[13] ONF – Open Network Foundation
https://www.opennetworking.org/ Accedido: 29-07-2019

[14] Internet Research Task Force
https://irtf.org/Accedido: 29-07-2019

[15] Unión Internacional de Telecomunicaciones
http://www.itu.int/net/ITU-SG/regional-es.aspx Accedido: 29-07-2019

[16] IEEE Xplore Digital Library
https://ieeexplore.ieee.org/Xplore/home.jsp Accedido: 29-07-2019

[17]Estado del arte en redes definidas por software (SDN) -
Miguel Ángel Barrera Pérez; Neider Yampol Serrato Losada; Elisa Rojas Sánchez; Giovani Mancilla Gaona
Visión Electrónica Vol. 13 No. 1 (2019) • January-June • p.p. 178-194 • ISSN 1909-9746 • ISSN-E 2248-4728 • Bogotá (Colombia)

[18]L. Pu W. Wu C. Akyildiz, f. Ahyoung. A roadmap for traffic engineering in SDN-Open-Flow networks.
https://bwn.ece.gatech.edu/projects/sdn-tecs/SDN-TE-survey.pdf Accedido: 29-07-2019

[19] What is MPLS-TE
http://www.ciscopress.com/articles/article.asp?p=30436&seqNum=2
Accedido: 13-02-2020

[20] Razones para usar SDNs- Ing. Esteban Carisimo, Dr. Ing. J.Ignacio Alvarez-Hamelin ,VII Encuentro Nacional de Técnicos, ArNOG - CoNexDat – INTECIN (UBA–CONICET), Buenos Aires, Argentina-
https://cnet.fi.uba.ar/esteban_carisimo/presentaciones/ArNOG_2017_1.pdf
Accedido: 14-02-2020

[21]What is an SLA? How to Use Service-Level Agreements for Success
https://www.process.st/sla-service-level-agreement/
Accedido: 24-02-2020

[22] Reenvío del tráfico en una red con IPQoS: Comportamientos por salto
https://docs.oracle.com/cd/E19957-01/820-2981/ipqos-intro-54/index.html
Accedido: 24-02-2020

[23] Inter-domain Traffic Conditioning Agreement (TCA) Exchange Attribute
https://tools.ietf.org/id/draft-ietf-idr-sla-exchange-11.html
Accedido: 24-02-2020

[24] Interoperación de redes RSVP / Intserv y Diffserv Marzo, 1999. draft-ietf-issll-diffserv-rsvp-01.txt
Accedido: 14-02-2020

[25] Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
https://tools.ietf.org/html/rfc2474
Accedido: 14-02-2020

[26] Cisco – Quality of Service
https://www.cisco.com/c/en/us/products/ios-nx-os-software/quality-of-service-qos/index.html
Accedido: 29-07-2019

[27] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken, The nature of data center traffic: Measurements & analysis, in: Proc. ACM SIGCOMM IMC, Chicago, Illinois, USA, 2009.

[28] A Load Balancing Method Based on SDN. Author(s) Mao Qilin ; Shen Weikang - 7th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2015.
https://ieeexplore.ieee.org/document/7263504
Accedido: 14-02-2020

[29] Aplicación de Balanceo De Carga Dinámico Para Servidores, Basada En Redes Definidas Por Software. Revista Ibérica de Sistemas y Tecnologías de Información. Carlos Enrique MINDA GILCES1, Rubén PACHECO VILLAMAR2.
http://www.scielo.mec.pt/pdf/rist/n32/n32a06.pdf
Accedido: 14-02-2020

[30] Jack Tsai, Tim Moors. A Review of Multipath Routing Protocols: From Wireless Ad Hoc to Mesh Networks. National ICT Austalia / University of New South Wales, Australia, 2006

[31] OSS/BSS: sistemas para operadores pensando en los usuarios-
By Jorge Fernando Negrete - octubre 27, 2014. https://www.mediatelecom.com.mx/2014/10/27/oss-bss-sistemas-para-operadores-pensando-en-los-usuarios/Accedido: 16-01-2019

[32] ¿Qué es Saas? – Microsoft Co.
https://azure.microsoft.com/es-es/overview/what-is-saas/-
Accedido: 16-01-2019

[33]Software-Defined Networks and OpenFlow - The Internet Protocol Journal, Volume 16, No. 1 - Marzo 2013 – William Stallings
https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html. Accedido: 16-01-2019

[34] Software-Defined Networks and OpenFlow - The Internet Protocol Journal, Volume 16, No. 1 - Marzo 2013 – William Stallings
https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html. Accedido: 16-01-2019

[35] NoviWare 400 introduces high-performance OpenFlow 1.5 switching. Fuente: https://novi-flow.com/nw400-2/ Accedido: 04-09-2020

[36]Ching-Hao, Chang and Dr. Ying-Dar Lin OpenFlow Version Roadmap, 2015 Fuente: http://speed.cis.nctu.edu.tw/~ydlin/miscpub/indep_frank.pdf
Accedido: 11-06-2019

[37]Implementación De Un Módulo De Comunicaciones Openflow Para Smartnet. Douglas Alexander Aguacía Fiscó-Pontificia Universidad Javeriana Facultad De Ingeniería Electrónica Departamento De In-Geniería Electrónica Bogotá -2014.
https://repository.javeriana.edu.co/bitstream/handle/10554/16509/AguaciaFiscoDouglasAlexander2015.pdf?sequence=1&isAllowed=y
Accedido: 16-02-2020

[38]OpenFlow v1.3 Messages and Structures- https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html
Accedido: 16-02-2020

[39] What is SDN?
https://www.ciena.com/insights/what-is/What-Is-SDN.html
Accedido: 15-01-2019

[40] What is SDN? - Juniper
https://www.juniper.net/us/en/solutions/sdn/what-is-sdn/
Accedido: 15-01-2019

[41]Andrei Bondkovskii, John Keeney, Sven van der Meer, Stefan Weber, Qualitative Comparison of Open-Source SDN Controllers, School of Computer Science and Statistics. Trinity College Dublin. Disponible: https://pdfs.semanticscholar.org/d220/a081d98277e68d0285cc4a9824d0e427ed76.pdf - Accedido: 31-07-2019

[42] RIP - Overview .
https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_rip/configuration/xe-16-5/irr-xe-16-5-book/irr-cfg-info-prot.html
Accedido: 26-07-2020

[43] ATM – Arquitectura y Servicios
http://repositorio.pucp.edu.pe/index/bitstream/handle/123456789/28691/Redes_Cap16.pdf?sequence=16&isAllowed=y Accedido: 03-08-2019

[44]Requirements for Ethernet VPN (EVPN) – RFC 7209
https://tools.ietf.org/html/rfc7209
Accedido: 03-08-2019

[45] Networks World from IDG
https://www.networkworld.es/networking/como-lidiar-con-redes-de-dispositivos-iot
Accedido: 03-11-2019

[46]Descubriendo CEF- Publicado por David P. en Recursos Educativos (Español) el 07-jun-2019
learningnetwork.cisco.com/.../06/07/descubriendo-cef
Accedido: 24-02-2020

[47]-SDN: Software Defined Networks- Autores Thomas D. Nadeau and Ken Gray-Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Copyright © 2013

[48] ASICs: Only the hard facts count https://www.elmos.com/produkte/special-projects/asic-design.html Accedido: 03-08-2019

[49] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S, Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, Vol.103, No.1, pp.14-76, Jan.2015. Accedido: 26-07-2020

{50} S. Tomovic, N. Prasad, I. Radusinovic, "SDN control framework for QoS provisioning", 22nd Telecommunication Forum TELFOR 2014, pp. 111-114, Belgrade, Serbia, November 2014.

[51} POX software. Disponible: http://noxrepo.org/pox/about-pox/ Accedido: 26-07-2020

[52] Introduction to Segment Routing https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/seg_routing/configuration/xe-3s/segrt-xe-3s-book/intro-seg-routing.pdf Accedido: 26-07-2020

[53] IETF SDN: I2RS uses traditional routing protocols in software networks
https://searchnetworking.techtarget.com/tip/IETF-SDN-I2RS-uses-traditional-routing-protocols-in-software-networksAccedido: 24-10-2019

[54]Una arquitectura para la interfaz con el sistema de enrutamiento (I2RS)RFC 7921 –
https://tools.ietf.org/html/rfc7921
Accedido:24-02-2020

[55] North-Bound Distribution of Link-State and Traffic Engineering (TE) - Information Using BGP
https://tools.ietf.org/html/rfc7752
Accedido: 24-02-2020

[56] Technopedia, Control Plane.
https://www.techopedia.com/definition/32317/control-planeAccedido: 31-07-2019

[57] Juniper – Quienes Somos https://www.ejuniper.com/es/juniper/quienes-somos/
Accedido: 03-08-2019

[58] NetConf - RFC 6241 https://tools.ietf.org/html/rfc6241Accedido: 03-08-2019

[59] SNMP – RFC 1157
https://tools.ietf.org/html/rfc1157
Accedido: 03-08-2019

[60] RED HAT – Telecomunicaciones
https://www.redhat.com/es/technologies/industries/telecommunications/OSS-BSS
Accedido: 03-08-2019

[61] Dynamic Load Balancing Application for Servers, Based on Software Defined Networking RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação versão impressa ISSN 1646-9895 ISTI no.32 Porto jun. 2019. http://dx.doi.org/10.17013/risti.n32.67-82. Autores: Carlos Enrique Minda Gilces1, Rubén Pacheco Villamar2
Accedido: 24-02-2020

[62] LDP Specification – RFC 5036
https://tools.ietf.org/html/rfc5036 Accedido: 03-08-2019

[63] Resource ReSerVation Protocol (RSVP) – RFC 2205
https://tools.ietf.org/html/rfc2205
Accedido: 03-08-2019

[64] Iperf
https://iperf.fr/iperf-download.php Accedido: 04-12-2019

[65] Welcome Page to NUTTCP
https://www.nuttcp.net/Welcome%20Page.html
Accedido: 04-12-2019

[66] NetPerf Manual
http://www.openss7.org/netperf_manual.html
Accedido: 04-12-2019

[67] Introducción a JSON
https://www.json.org/json-es.html
Accedido: 04-12-2019

[68] Disponible UBUNTU 14.04.1 LTS
https://lignux.com/disponible-ubuntu-14-04-1-lts/
Accedido: 04-12-2019

[69] GNS3 – Universidad Complutense de Madrid
https://www.ucm.es/pimcd2014-free-software/gns3 Accedido: 04-12-2019

[70] Qué es Dynamips
https://es.scribd.com/document/171078177/Guia-Dynamips
Accedido: 09-12-2019

[71] Dynagen
https://dynagen.com/
Accedido: 09-12-2019

[72] Institut Puig Castellar
https://elpuig.xeill.net/Members/vcarceler/articulos/qemu
Accedido: 09-12-2019

[73] "Open Networking Summit 2013," Santa Clara, CA, 2013.
Accedido: 04-12-2019

[74]S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," in IEEE Symposium on Computers and Communications (ISCC), Funchal, 2014.
Accedido: 04-12-2019

[75] Introduction to Mininet - InHo Cho edited this page on 26 Sep 2018· 151 revisions - GitHub Introduction to Mininet. (s.f.). Introduction to Mininet. Recuperado de https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

[76] RYU – Controlador SDN
https://osrg.github.io/ryu/ Accedido: 09-12-2019

[77] Cisco Systems: Cisco Networking Academy
https://www.netacad.com/es Accedido: 09-12-2019

[78] Open Shortest Path First Protocol -RFC 2328
https://www.ietf.org/rfc/rfc2328.txt Accedido: 06-12-2019

[79]TRANSMISSION CONTROL PROTOCOL
https://tools.ietf.org/html/rfc793 Accedido: 09-12-2019

[80] User Datagram Protocol
https://tools.ietf.org/html/rfc768 Accedido: 09-12-2019

[81] RIP v2. Routing Information Protocol
https://tools.ietf.org/html/rfc2453 Accedido: 05-01-2020

[82] IS IS Intermediate System to Intermediate System (RFC 3787)
https://www.rfc-editor.org/rfc/rfc3787.txt Accedido: 05-01-2020

[83] Cisco Community
https://community.cisco.com/t5/networking-documents/ospf-and-is-is-differences/ta-p/3126940
Accedido: 05-01-2020

[84] Referencias a CSPF
https://tools.ietf.org/html/rfc4105 Accedido: 05-01-2020

[85] Constrained Shortest Path First
https://www.metaswitch.com/knowledge-center/reference/constrained-shortest-path-first-cspf
Accedido: 05-01-2020

[86] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, “A knowledge plane for the Internet,” in Proc. ACM SIGCOMM’03, Karlsruhe, Germany, 2003, pp. 3–10.

[87] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcon,´M. Sole, V. Munt´ es-Mulero, D. Meyer, S. Barkai, M. J. Hibbett,´G. Estrada, K. Ma’ruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, “Knowledge-defined networking,” SIGCOMM Comput. Commun. Rev., vol. 47, no. 3, pp.2–10, sep. 2017.

[88] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, “Machine learning for networking: Workflow, advances and opportunities,” IEEE Network, vol. 32, no. 2, pp. 92–99, March 2018.

[89] T. Braun, M. Diaz, J. E. Gabeiras, and T. Staub, End-to-end quality of service over heterogeneous networks: Springer Science & Business Media, 2008.

[90] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 AsiaPacific, 2012

[91] Q. Duan, C. Wang, and X. Li, "End-to-End Service Delivery with QoS Guarantee in Software Defined Networks," arXiv preprint arXiv:1504.04076, 2015

[92] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, “Unsupervised machine learning for networking: Techniques, applications and research challenges,” arXiv preprint arXiv:1709.06599, 2017.

[93] G. Xu, Y. Mu, and J. Liu, “Inclusion of artificial intelligence in communication networks and services,” ITU Journal: ICT Discoveries, no. 1, pp. 1–6, Oct. 2017.

[94] Intent-Based Networking (IBN): Bridging the gap on network complexity. Intent-Based Networking was a result of the need for greater network automation.

https://www.networkworld.com/article/3428356/intent-based-networking-ibn-bridging-the-gap-on-network-complexity.html Accedido: 01-10-2020