PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ESTUDIO DE ALGORITMOS DE ENRUTAMIENTO EN REDES SDN

Trabajo de investigación para obtener el grado académico de BACHILLER EN CIENCIAS CON MENCIÓN EN INGENIERÍA DE LAS TELECOMUNICACIONES

AUTOR:

Julio Ricardo Huamaní Jerí

ASESOR:

Dr. César Augusto Santiváñez Guarniz

Lima, enero de 2021

Resumen

El presente trabajo de investigación presenta una revisión de literatura acerca del impacto en los parámetros de la Calidad de Servicio (QoS) al utilizar diferentes algoritmos de enrutamiento en una solución SDN (Red Definida por Software).

En la introducción se presenta el contexto de las redes SDN, así como los objetivos del presente trabajo de investigación, la motivación para el mismo, mencionando la necesidad de cuantificar la mejora obtenida al utilizar algoritmos de enrutamiento en una red SDN e indicando que la discusión se hará en base a revisión bibliográfica.

Luego, se presenta el marco teórico acerca de una red SDN, sus componentes y las ventajas que ofrecen respecto a una red convencional. Continuando con el marco teórico, se presenta la definición de enrutamiento en una red y la base matemática que fundamenta el impacto que puede generar en el *throughput* y latencia, que son parámetros de la Calidad de Servicio (QoS). Asimismo, en el apartado se hace mención de artículos IEEE relacionados con el tema de enrutamiento y el impacto generado en la red, cuyos resultados muestran la diferencia del *throughput* y/o latencia al hacer uso de distintos algoritmos de enrutamiento.

Finalmente, se hace un análisis de los resultados que dichos artículos reportaron, teniendo en cuenta el contexto de pruebas, el tipo de topología utilizada, la cantidad de tráfico generada, entre otros factores, para realizar una discusión y hallar conclusiones sobre el impacto de los algoritmos de enrutamiento en los parámetros de QoS en redes SDN.

Tabla de contenidos

Índice de Figuras

Índice de Tablas

Introducción

El paradigma de Software Defined Network (SDN) se viene aplicando como solución empresarial en los últimos 10 años y ha estado obteniendo cada vez más una inserción en el mercado debido a la reducción de costos respecto a una solución de red usual y a sus múltiples aplicaciones: Enrutamiento, ciberseguridad, conexión con nube, etc. Sin embargo, la naturaleza de una red SDN también trae consigo nuevos tipos de tráfico a tener en cuenta (tráfico entre dispositivos de red y controladores, cuellos de botella, tiempos de convergencia para enrutamiento, etc.), lo que lleva a la reflexión de analizar de cuáles son los beneficios y las desventajas que se adquieren al utilizar una solución de SDN para una determinada topología.

La presente investigación tiene enfocado el problema de analizar parámetros de Calidad de Servicio como el *throughput* y latencia obtenidos en una red SDN al utilizar distintos algoritmos de enrutamiento. Con la reciente popularidad de redes que implementa esta tecnología, se hace imperativo el conocimiento de las ventajas que se obtienen a comparación de una red normal. Si se lograra encontrar una cuantificación adecuada de la mejora de la red, se podrían implementar mejores algoritmos en una red de modo que se obtenga una mejora en términos de rendimiento y latencia. Por lo tanto, la presente investigación tiene como objetivos: indagar sobre los algoritmos que se puedan implementar en un controlador de SDN, identificar las técnicas (si existen) para desarrollar algoritmos dada una topología y cuantificar la mejora en función de parámetros de Calidad de Servicio. Para lo anterior, se realizará una investigación bibliográfica y se analizarán los resultados mostrados en los artículos para dar paso a una discusión y llegar a conclusiones respecto al impacto de la métrica de QoS al utilizar distintos algoritmos de enrutamiento.

1. Beneficios de SDN Respecto a las Soluciones Convencionales

En este apartado, se hará una presentación junto a los beneficios que se tiene al utilizar una Red Definida por Software en lugar de una solución de red convencional "legacy" con el fin de justificar el porqué del estudio de los algoritmos que se pueden implementar para una red SDN.

1.1 Beneficios de SDN

1.1.1 Desacoplo del plano de control con el plano de datos.

La premisa principal de una arquitectura de SDN es la separación entre el Plano de Control y el Plano de Datos, esto significa que no se utilizarán los mismos enlaces para el flujo de datos y el flujo de paquetes de control simultáneamente, sino que se utilizarán

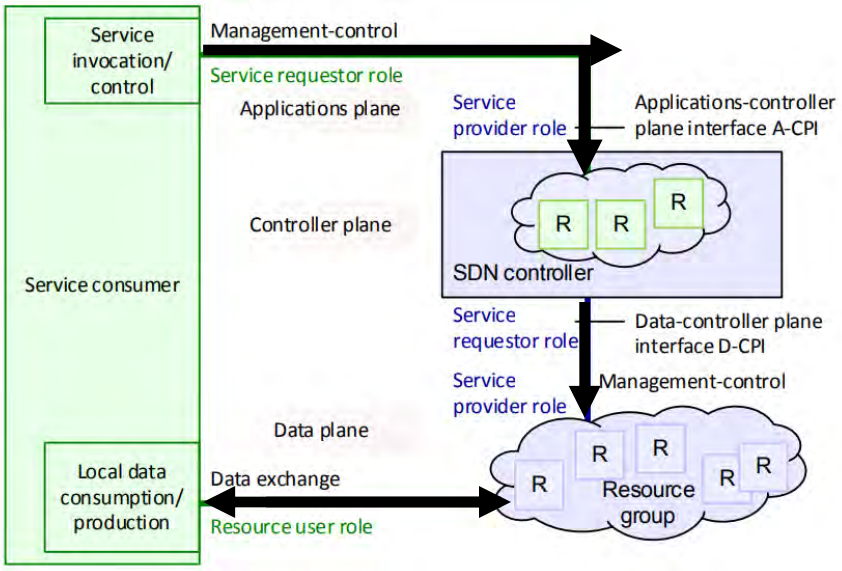

Figura 1-1 Modelo básico de interacción entre la capa de datos y la capa de control, los enlaces se encargan de un flujo exclusivo de datos o de control y el dispositivo central es el controlador SDN.

SDN Architecture Issue [25]

enlaces dedicados tanto para el flujo del tráfico (Switch a Switch) como del flujo de control (Switch a Controlador).

De la misma manera, la arquitectura SDN se define en 3 capas: Capa de Control, Capa de Datos y Capa de Aplicación, mostrando la marcada diferenciación que hace esta tecnología al momento de interactuar con los flujos de paquetes:

-Capa de Aplicación: Consta de los programas y entidades que operan a nivel de software, aquí se ubican las aplicaciones de negocios.

-Capa de Control: Consta de los paquetes de control que se encargan de darle instrucciones al Controlador respecto a qué hacer cuando se detecten nuevos flujos, aquí se ubican los servicios HTTP, API REST, entre otros que el controlador SDN utilice.

-Capa de Datos: Consta de los paquetes de datos que viajan a través de la red, son usualmente enviados desde un nodo terminal a otro.

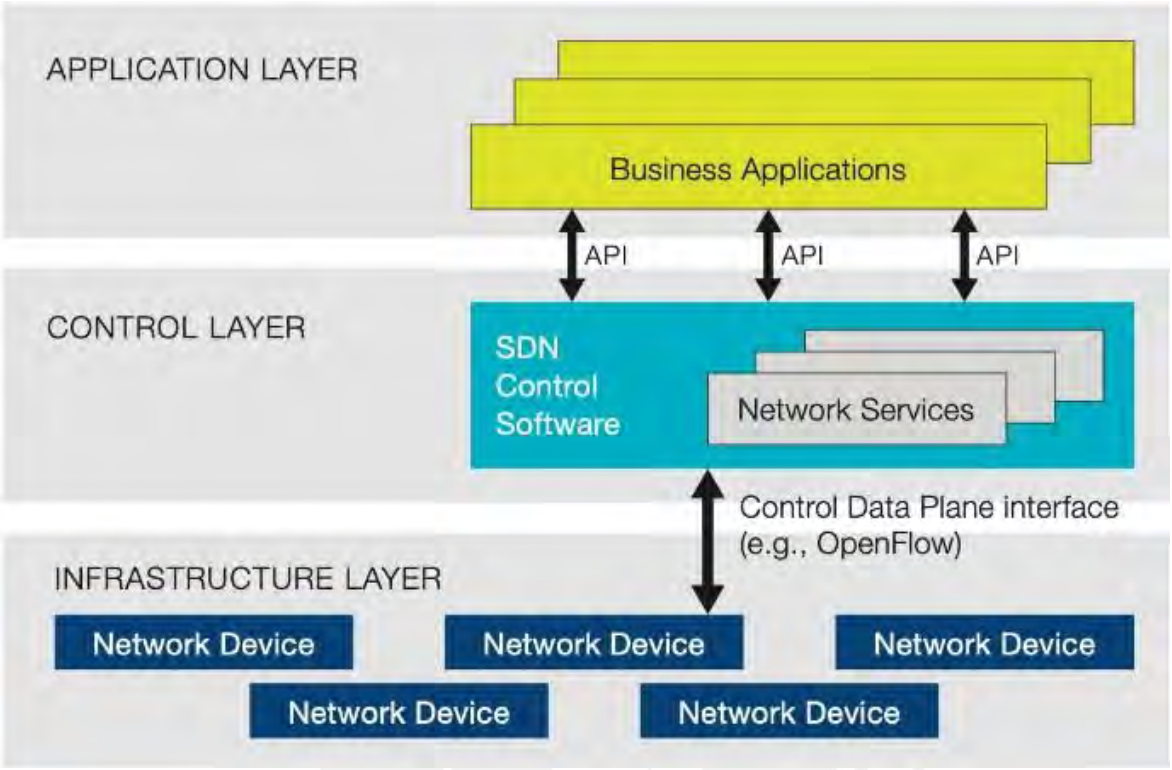

Figura 1 - 2 Arquitectura básica de una red SDN. Se aprecia la diferenciación entre las 3 capas mencionadas

What is Software Defined Networking? Definition [11]

1.1.2 Programación del comportamiento de la red SDN.

El componente principal que muy probablemente defina a una solución SDN respecto a otra red es el Controlador, este dispositivo se ubica en una posición lógicamente central en toda la red y posee una visión completa de todos los dispositivos, a la vez que permite configurarlos a todos de forma simultánea. Las interfaces son el concepto clave que permite interconectar una y otra capa de la arquitectura SDN. Finalmente, la gran ventaja que se tiene al unir estas capas, es la capacidad de poder configurar el comportamiento de los flujos que viajan a través de la red haciendo uso de la programación (software).

1.1.3 Estadísticas de la red SDN.

Una red SDN posee módulos que permiten tener estadísticas de los flujos y los dispositivos de red en tiempo real, de este modo, utilizando la capacidad de programación del controlador sumado a la obtención monitoreo constante, se puede lograr la optimización de la red así como las mejoras de sus parámetros como el *throughput* y latencia.

A continuación, se mostrará una tabla simple mostrando las principales diferencias que se tienen al utilizar SDN a comparación de una red convencional:

Tabla 1 - 1 Resumen de diferencias entre una red SDN y una red convencional ELABORACIÓN PROPIA

Características	**SDN**	**Red convencional**
Funciones	Datos y control desacoplados, capacidad de programación	Uso de protocolos nuevos por problema, manejo complejo del control de la red
Configuración	Configuración automatizada con validación centralizada	Error inducido en caso de problemas de configuración manual
Rendimiento	Control dinámico a nivel de toda la red con información de distintas capas	Información limitada, y configuración relativamente estática, no hay uso de programación
Innovación	Fácil implementación en caso de nuevas ideas. Pruebas solo requieren un entorno aislado	Difícil implementación de nuevas ideas. La realización de pruebas es limitada

1.1.2 Retos de SDN.

Si bien es cierto que SDN fomenta la innovación e investigación debido a que ofrece una mayor programación y control de la red por parte del usuario, es una tecnología relativamente nueva, por lo que también presenta varios problemas que necesitan ser abordados y corregidos con el fin de obtener una tecnología con bases lo suficientemente sólidas como para participar en el mercado laboral como el caso de la escalabilidad (El tráfico de control se incrementa mientras existen más dispositivos en la red, así como también el aumento de controladores), la seguridad (el hecho de tener acceso al controlador significa que el atacante puede tener acceso a toda la red), entre otros.

1.1.3 Factores de cuantificación de mejora entre una red SDN y una solución convencional.

- Retardo punto a punto: Se define como el tiempo promedio que le toma a un paquete llegar a su destino. Esto incluye todos los retardos acumulados que el paquete acumula a lo largo de su viaje por la red. El cálculo de este parámetro se hace restándole el tiempo al que el primer paquete fue emitido por una fuente al tiempo que le toma al primer paquete llegar a su destino.

Matemáticamente se define como S/N donde S representa la suma del tiempo utilizado en entregar paquetes para cada destino mientras que N representa el número de paquetes recibidos por todos los nodos de destino.

- Tasa de entrega de paquetes: Se define como el número total de paquetes entregados sobre el tiempo total de simulación. El concepto detrás de este término es hallar un

promedio de los paquetes recibidos por la cantidad de paquetes enviados. Matemáticamente puede definirse como la suma de S1 y S2 donde S1 representa la suma de los paquetes de datos recibidos por cada destino y S2 es la suma de los paquetes generados por cada fuente.

- *Throughput*: Se define como la medida de cuántas unidades de información puede procesar un determinado sistema en una cantidad de tiempo dada. El *throughput* es usualmente medido en bits por segundo, pero también se puede cuantificar en paquetes por segundo o paquetes de datos por slot de tiempo.

1.2 Casos de Éxito de Redes SDN

A continuación se muestra una pequeña compilación donde el uso de redes SDN han resultado provechosos a comparación de una red convencional.

1.2.1 Comparación de una red solución SDN y una solución convencional.

El año 2017 se hicieron pruebas de comparación entre una red SDN y una red "legacy" con dispositivos inalámbricos dentro de la red. Para una mayor generalidad, se dispusieron los nodos en distintas topologías, los resultados que se mostrarán en este apartado son del resultado del análisis en malla, sin embargo, la conclusión en las otras topologías también resultaron similares:

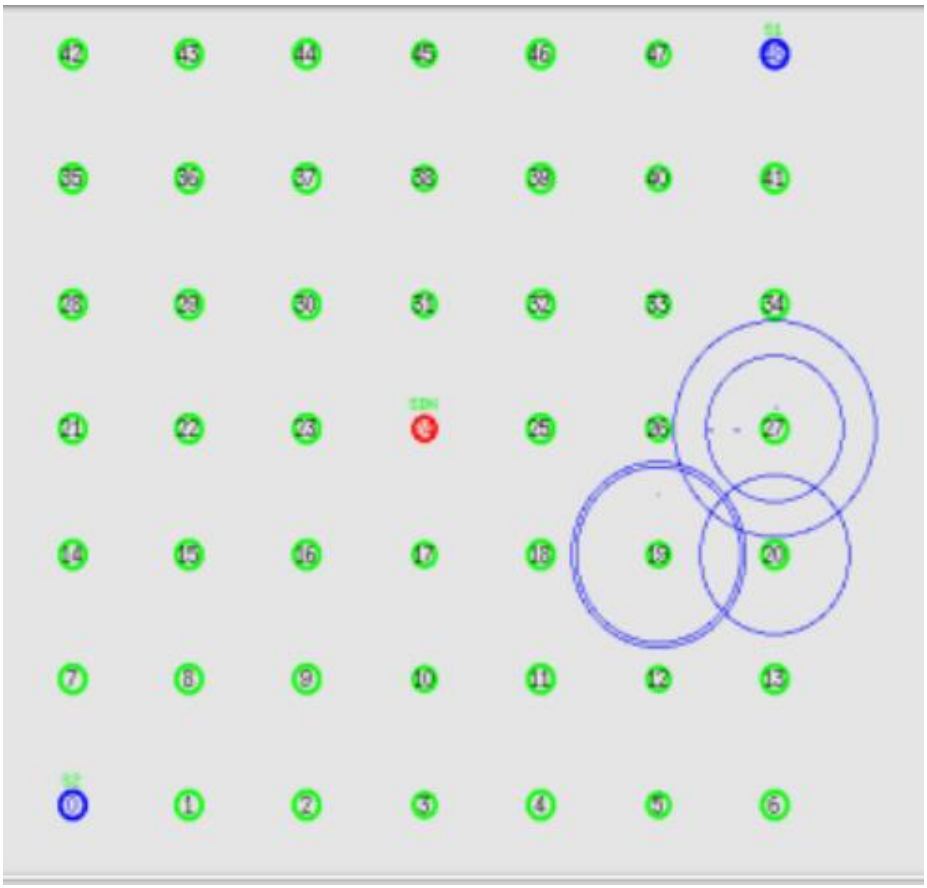

Figura 1 - 3 Topología en malla, nodo rojo como controlador SDN

Performance Analysis of Wireless Trusted Software Defined Networks (2017) [26]

Teniendo en cuenta la topología anterior, se obtuvieron los siguientes resultados en distintos parámetros de la red:

Consumo de energía:

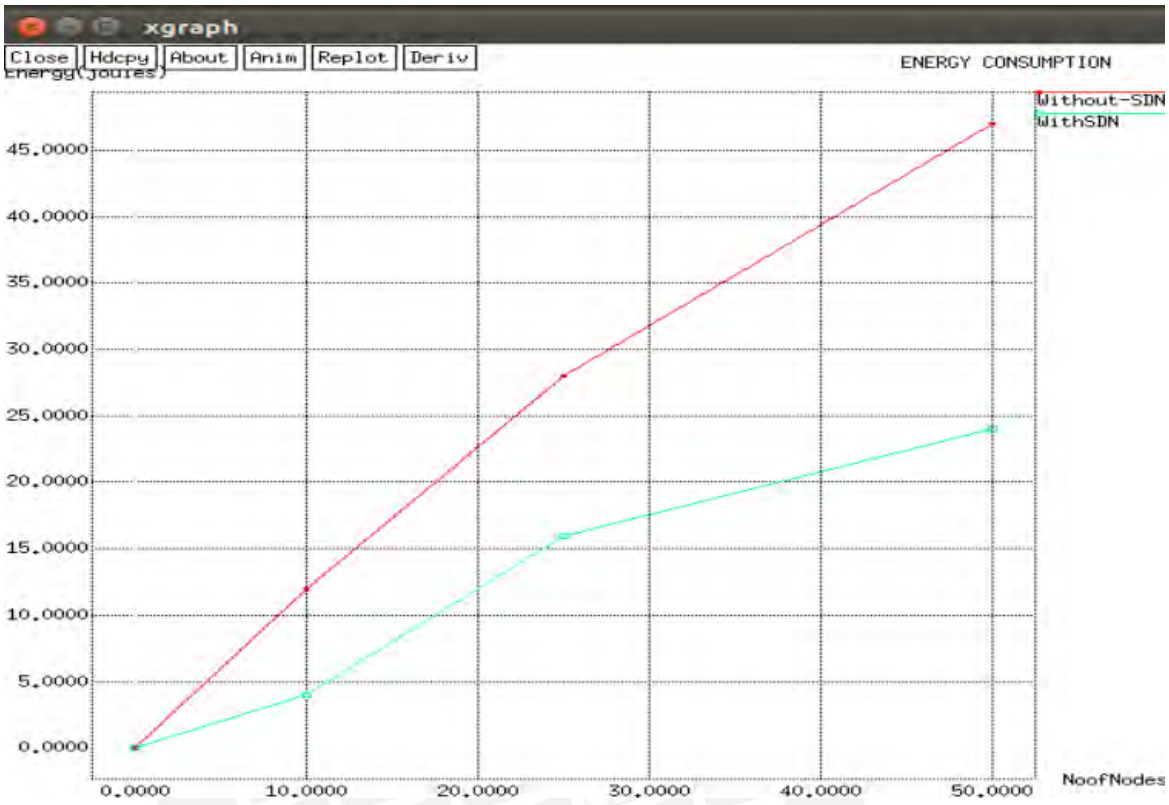

Figura 1 - 4 Comparación de consumo de energía entre una red SDN y una convencional

Performance Analysis of Wireless Trusted Software Defined Networks (2017) [26]

Se puede observar que a lo largo del tiempo, el consumo de energía de una red sin SDN (línea roja) es mucho menor que una solución SDN (línea celeste). De la misma manera, se observa la tasa de entrega de paquetes:

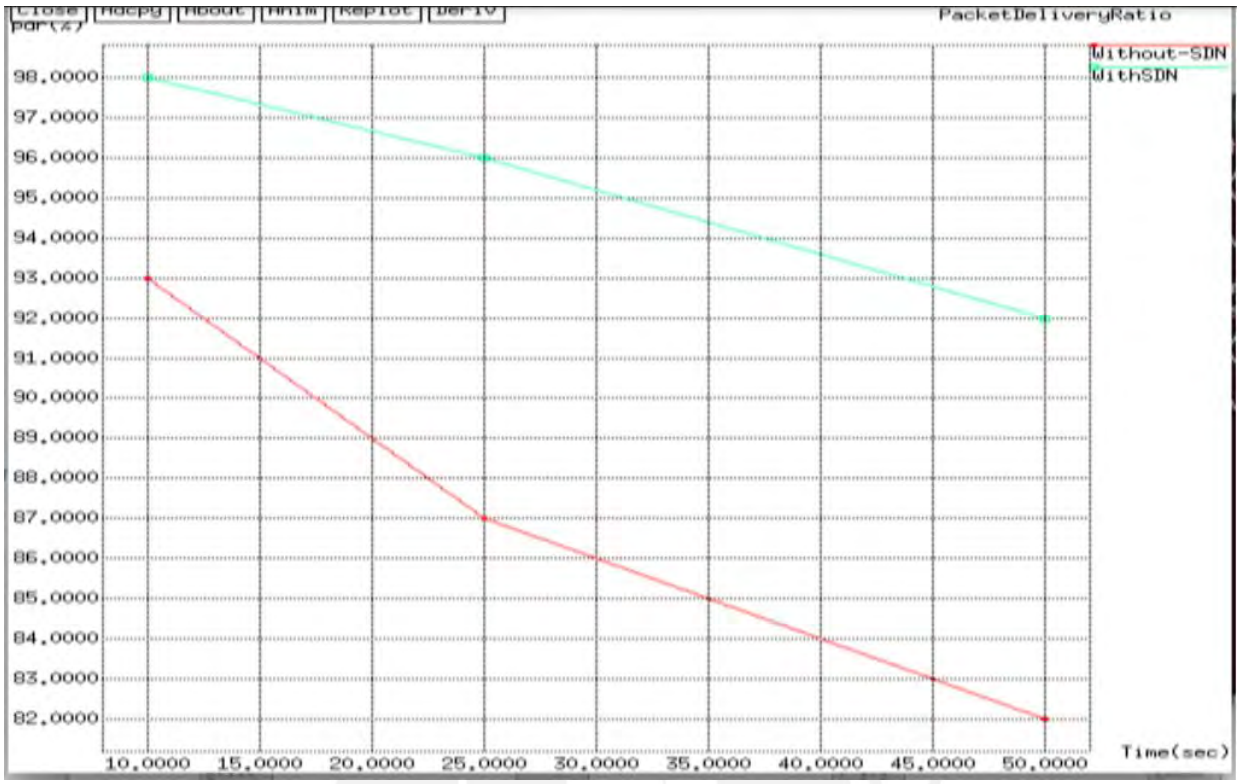

Figura 1 - 5 Comparación de la tasa de entrega de paquetes entre una red SDN y una convencional

Performance Analysis of Wireless Trusted Software Defined Networks (2017) [26]

Observamos que según el gráfico, la tasa de entrega de paquetes es mucho más grande que la solución convencional a lo largo del tiempo. Las pruebas prosiguieron con otros parámetros, en este caso el retardo de terminal a terminal:

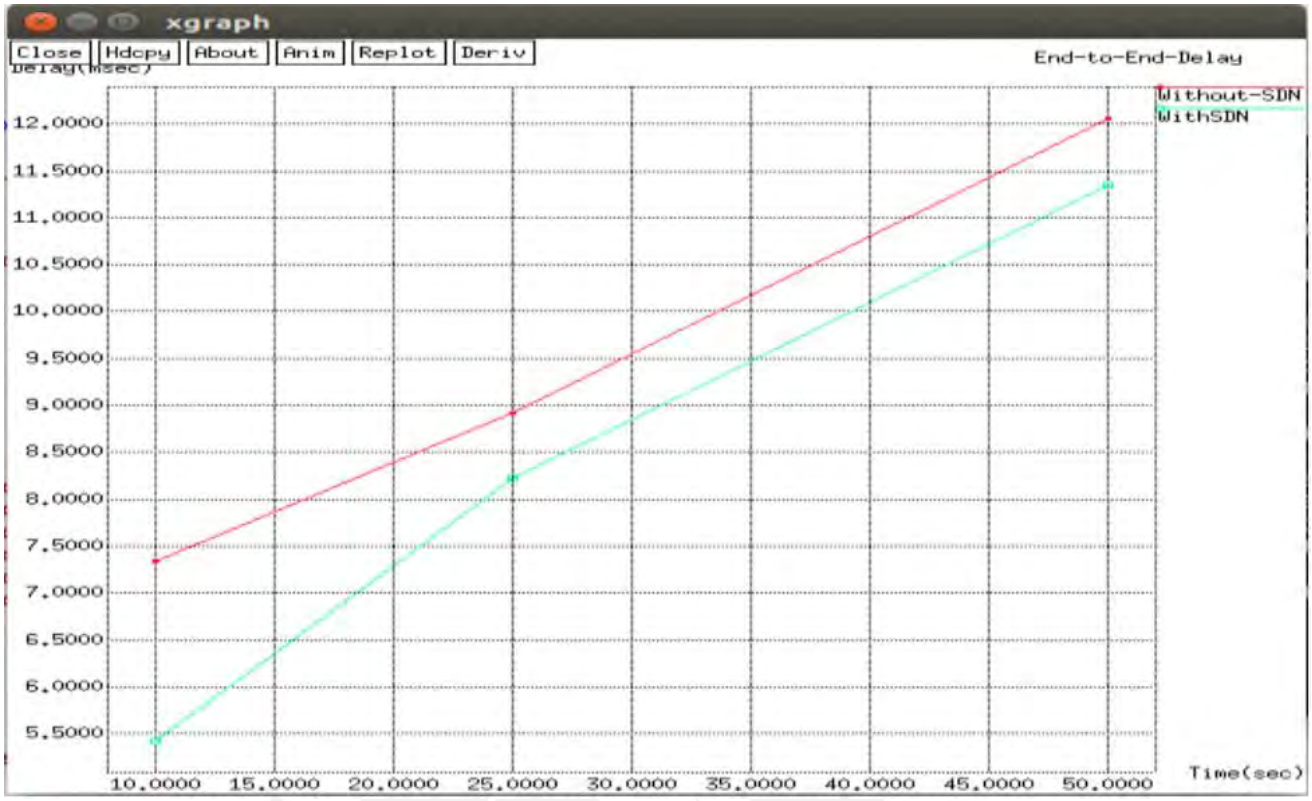

Figura 1 - 6 Comparación del retardo entre terminales de una red SDN y una convencional

Performance Analysis of Wireless Trusted Software Defined Networks (2017) [26]

Donde se puede observar que el retardo de terminal a terminal es mucho menor en las solución SDN a comparación de la solución convencional teniendo en cuenta nuestra topología en malla. Finalmente se hacen comparaciones respecto al *throughput* de la red:

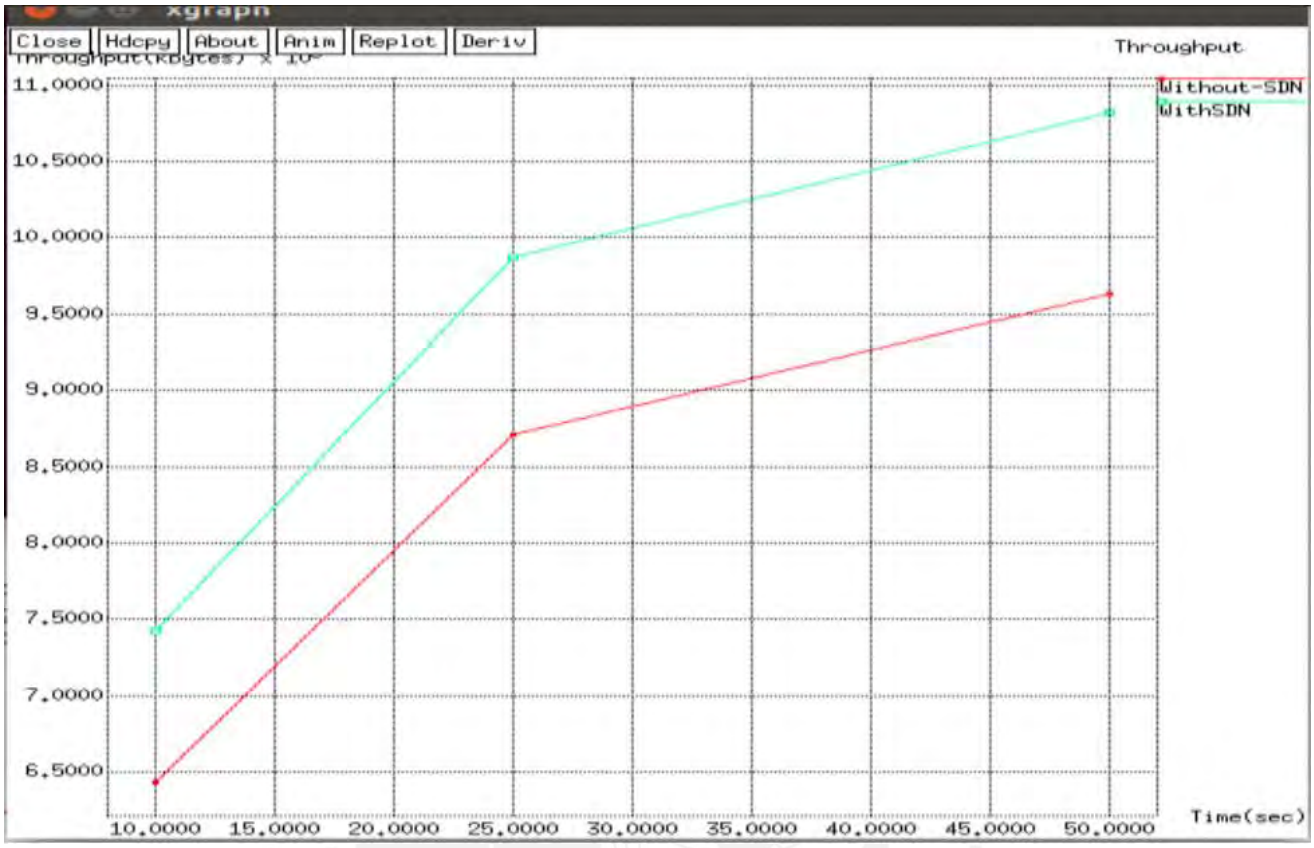

Figura 1 - 7 Comparación del *throughput* entre una red SDN y una convencional

Performance Analysis of Wireless Trusted Software Defined Networks (2017) [26]

De esta manera, se observa que el uso de una Red Definida por Software, trae consigo beneficios en diferentes parámetros de calidad respecto a una red convencional.

1.2.2 Migración de legacy a híbrido en la Universidad de Stanford.

Una de las principales motivaciones de la migración de la Universidad de Stanford fue obtener mejores datos respecto al uso de OpenFlow como tecnología viable. Stanford desplegó una red SDN totalmente funcional utilizando controladores OpenFlow sobre una parte de su campus. La migración fue enfocada al principio para los usuarios inalámbricos que se ubicaban en dos edificios. El resultado fue una red híbrida legacy-SDN que implicaba el uso de VLANs 802.1q interconectadas por un router capa 3.

En un edificio Stanford desplegó un switches OpenFlow de 48 puertos con soporte de enlaces de 1GigabitEthernet de diferentes marcas. El segundo edificio desplegó un solo switch OpenFlow. Un dominio de capa 2 fue utilizado en cada edificio para tener soporte de 34 Access Points. Estos Access Points tenían soporte para interfaces 802.11g que se ejecutaban con software basado en Linux y una estación base WiMAX en uno de los edificios. A continuación, se adjunta el diagrama.

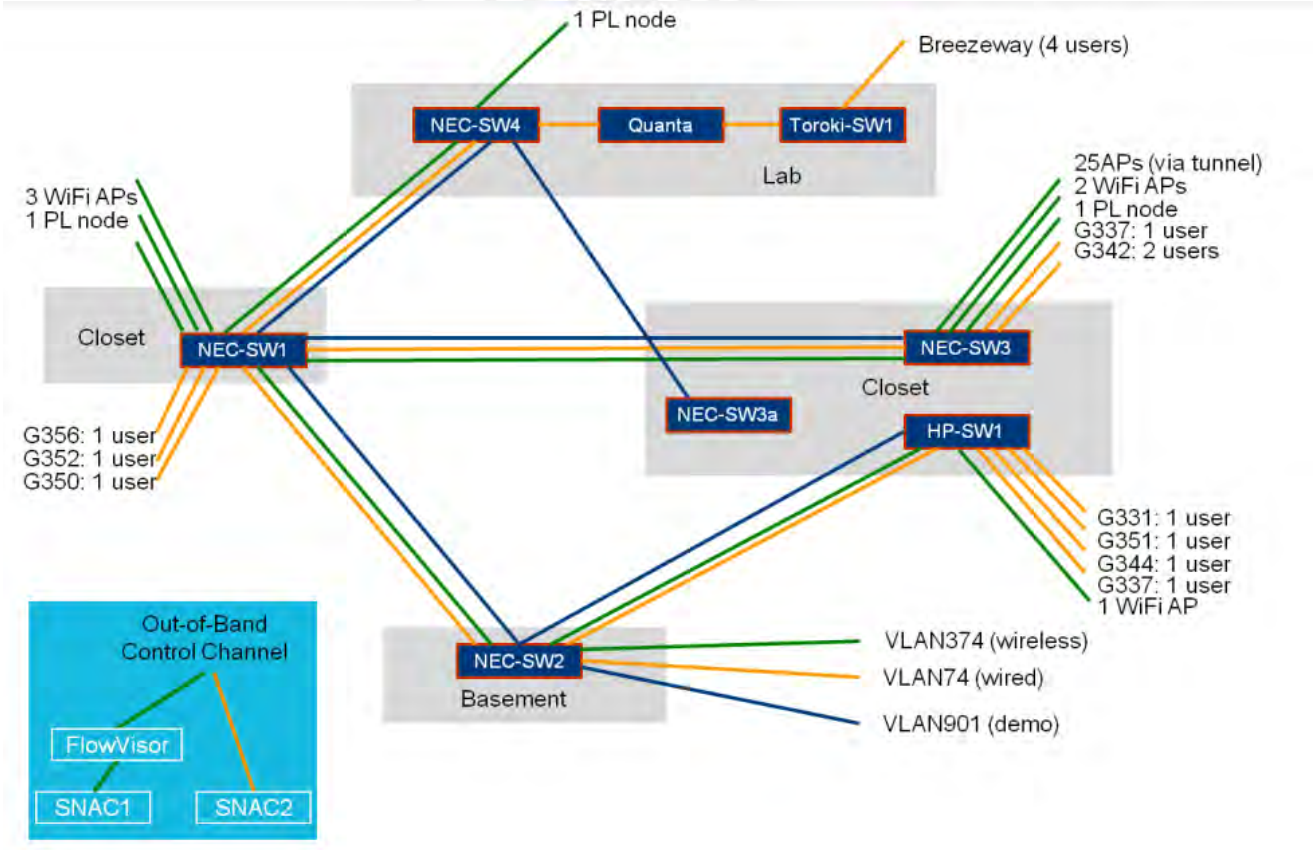

Figura 1 - 8 Esquema del despliegue de la Universidad de Stanford SDN

Migration Use Cases [12]

El despliegue resultó exitoso con una disponibilidad de red que excedía el 99.9% con una arquitectura que permitía volver al sistema legacy en caso de fallas significativas. Además, el rendimiento general de la red fue comparable a la red legacy sin impactos a la experiencia de usuario.

1.2.3 Migración de Google.

La red WAN OpenFlow de Google está organizada como dos backbones distintos, uno lleva tráfico de usuario hacia internet y el otro lleva tráfico interno entre los Data Centers de Google. La gran diferencia entre los requerimientos de los usuarios y la escala del proyecto volvieron a la migración de Google un reto único que demostraba la flexibilidad de OpenFlow.

El objetivo de la migración a SDN fue mejorar la escalabilidad, flexibilidad y la agilidad en gestionar la línea WAN respecto al tráfico de Internet de modo que se tuvieran mejoras en servicios como. Google+, Gmail, Youtube, Google Maps, entre otros.

Ambas redes WAN de Google soportan miles de aplicaciones individuales, grandes cantidades de tráfico y sensibilidades de latencia - siendo gobernados or prioridades en general. La red interna de Google que conecta múltiples Data Centers es una red basada en OpenFlow hasta el día de hoy y un caso de uso de SDN bastante conocido. Esta red entre Data Centers fue construida en una arquitectura de 3 capas: Capa de switch, capa de controlador de zona y capa de control global.

La migración a SDN se realizó en distintas etapas desde una arquitectura de hardware que implicaba un control monolítico-distribuido hasta uno físicamente

descentralizado (aunque lógicamente centralizado por el uso del controlador). La migración en general siguió 3 etapas:

-Red Inicial: En la etapa inicial, la red conectaba Data Centers a través de nodos legacy haciendo uso de enrutamiento E/I BGP y enrutamiento ISIS. Enrutadores de borde se organizaban como clústeres para funcionar como acceso entre la red y los Data Centers.

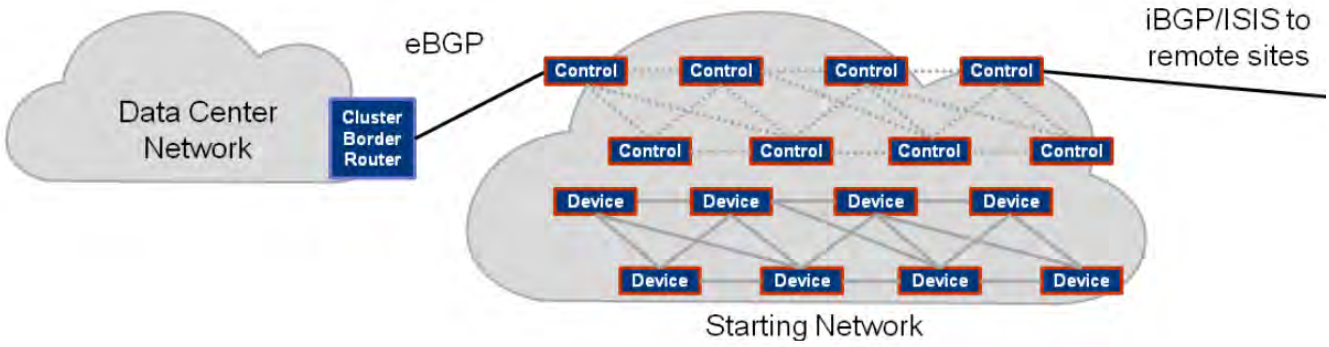

Figura 1 - 9 Fase 1 de la migración de Google

Migration Use Cases [12]

-Despliegue por fases: En esta etapa de red híbrida, una parte de los nodos de la red se habilitó para su uso con OpenFlow y control por parte de un controlador lógicamente centralizado utilizando Paxos, un controlador OpenFlow, además del uso del stack de enrutamiento Quagga que Google adaptó para sus requerimientos.

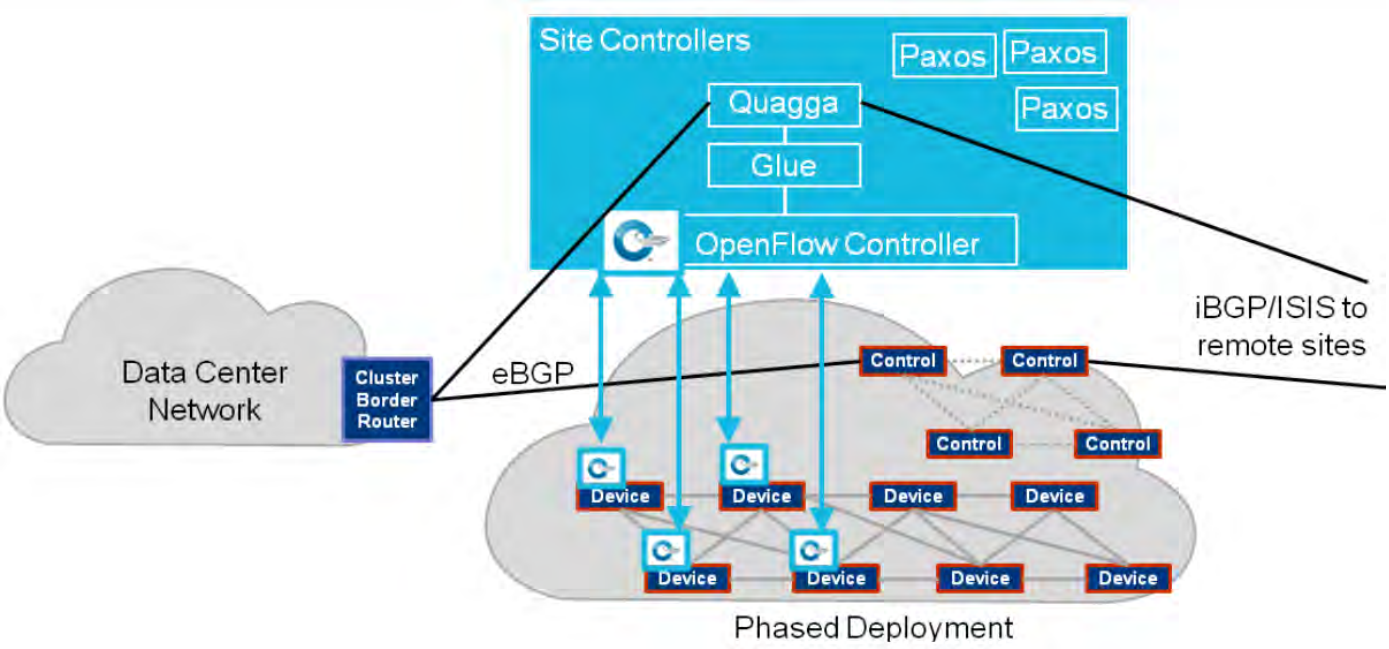

Figura 1 - 10 Fase 2 de la migración de Google, uso de la red híbrida

Migration Use Cases [12]

-Red Objetivo: En esta fase final, todos los nodos se activaron para su uso con OpenFlow. En la red destino, el controlador tiene acceso a toda la red. No hay correspondencia directa entre el Data Center y la red. El controlador también posee un servidor que realiza la ingeniería de tráfico en la red.

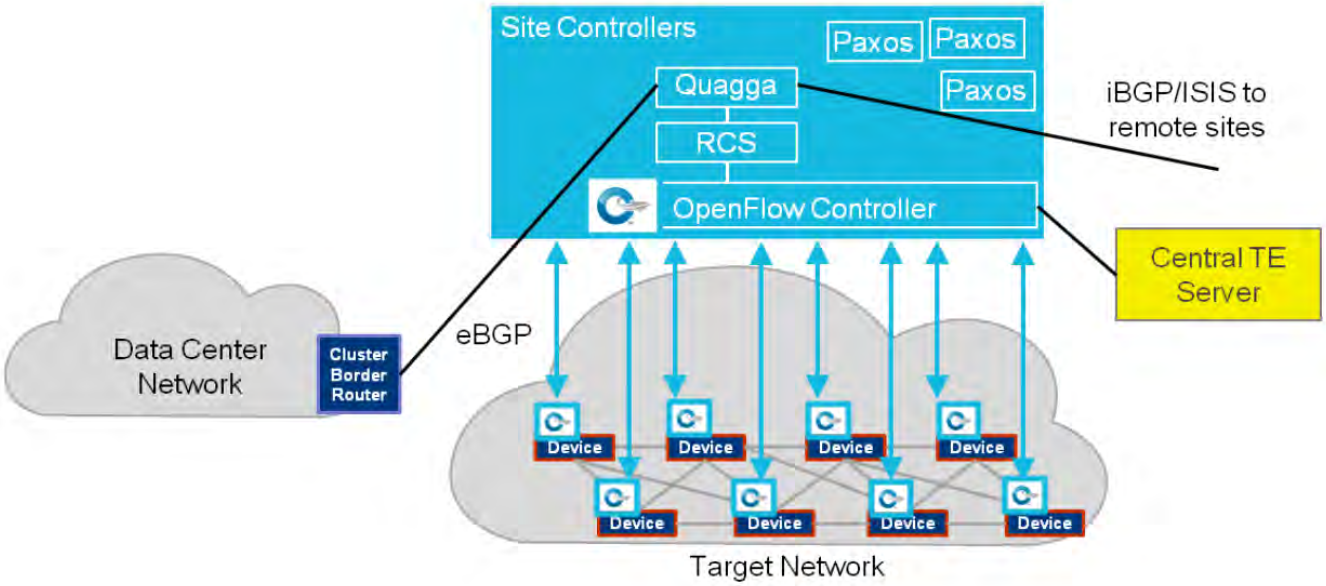

Figura 1 - 9 Fase 3 de la migración de Google, conversión a SDN

Migration Use Cases [12]

El éxito de Google al migrar a una red basada en OpenFlow podría malinterpretarse como un sistema con menor capacidad de tráfico que su red externa. En realidad, sucede lo contrario. No solo la red interna de Google lleva más tráfico que la red WAN externa de Google, también crece a una velocidad cada vez más grande. La red interna de Google soporta despliegue e interacción rápido del control y funcionalidad de la ingeniería de tráfico, la que llegó al 95% de su utilización. También está integrado de forma cercana con aplicaciones de usuario que permiten a la red adaptarse a fallos o a patrones de comunicación cambiantes.

1.2.4 Otros casos de migración.

Existen muchos otros casos de migración a soluciones SDN que se han mostrado de forma pública que involucran diversas capas de red y segmentos de cliente. Por ejemplo, NTT DOCOMO implementó un EPC móvil utilizando una red SDN. Esta migración fue motivada por el terremoto y tsunami en Japón del 2011 y le permitió a NTT DOCOMO mejorar la resistencia de la red contra futuros desastres. Además, la migración soportaba datos evolucionando rápidamente y aplicaciones de usuario intensivas en transacciones así como también le permitió a la empresa automatizar la gestión de políticas y recursos de red mientras mantenía los gastos bajo control.

2. Ingeniería de Tráfico

El presente capítulo contiene la revisión de literatura concerniente a los conceptos manejados en la ingeniería de tráfico en redes como el enrutamiento y la teoría de colas así como la problemática presentada conforme este se aplica a redes cada vez más extensas y complejas como las que se utilizan en soluciones empresariales o campus universitarios. Finalmente, se muestra un caso de implementación de cambio en el algoritmo de enrutamiento utilizado de forma predeterminada para el controlador Floodlight.

2.1 Algoritmos de Enrutamiento

Un algoritmo de enrutamiento suele definir como un protocolo de la capa de red que se encarga de guiar a los paquetes a su respectivo destino. El periodo de tiempo para realizar decisiones de enrutamiento depende si la red utiliza datagramas o circuitos virtuales. En una red de datagramas, dos paquetes sucesivos del mismo par de usuarios pueden viajar por rutas distintas y una decisión de enrutamiento es necesario cuando cada circuito virtual es encendido.

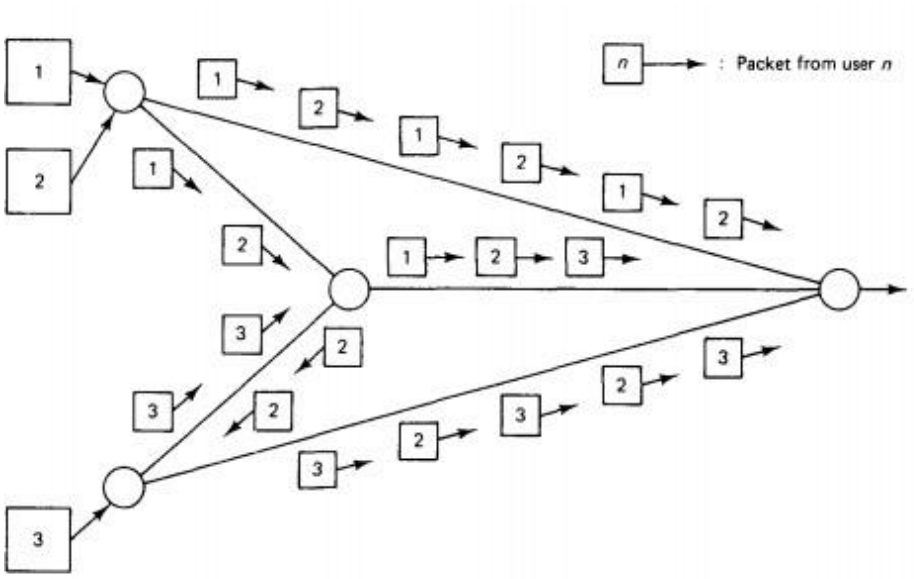

Figura 2-1. Enrutamiento en una red de datagramas.

Dos paquetes del mismo par de usuarios pueden viajar por rutas distintas. Una decisión de enrutamiento se toma para cada paquete individual.

DATA NETWORKS (1992) [1]

En una red de circuitos virtuales, una decisión es tomada cuando cada circuito virtual es encendido. El algoritmo de enrutamiento es utilizado para escoger una ruta de comunicación para el circuito virtual. Luego, todos los paquetes del circuito virtual usan esta ruta hasta el momento que se finaliza el circuito o si se crea una nueva ruta por alguna razón.

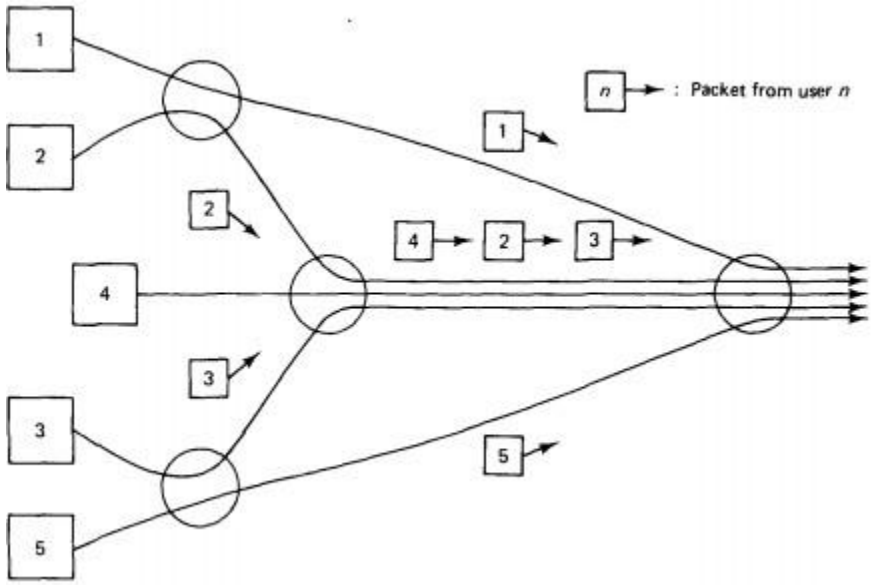

Figura 2-2. Enrutamiento en una red de circuitos virtuales.

Todos los paquetes utilizan la misma ruta todo el tiempo. Una decisión de enrutamiento aparece cuando un circuito virtual nuevo aparece.

DATA NETWORKS (1992) [1]

El enrutamiento en una red involucra usualmente una compleja colección de algoritmos que funcionan más o menos de forma independiente y se comunican el uno al otro utilizando servicios de la información. La complejidad viene debido tres principales razones:

-Primero, para realizar enrutamiento debe existir coordinación entre cada nodo que compone la red en lugar de pensar solamente en dos módulos como protocolos de transporte y de enlace de datos.

-Segundo, el sistema de enrutamiento debe estar preparado en caso de fallas de enlaces y nodos, para lo cual todos los componentes deben estar informados constantemente del estado actual de la topología.

-Tercero, para que el protocolo alcance un alto rendimiento, el algoritmo de enrutamiento puede necesitar modificar sus rutas cuando algunas áreas se vuelvan congestionadas.

2.1.1 Parámetros de calidad de servicio.

La calidad de servicio describe la calidad de la comunicación percibido desde el punto de vista del usuario utilizando la red. Esta se puede cuantificar en función de distintos parámetros que se pueden medir de formas distintas. Además las mediciones se pueden obtener vía simulación o pueden ser modeladas con la matemática de la teoría de colas. Las siguientes medidas son usualmente consideradas importantes:

Ancho de banda: Usualmente medido en bits/segundo es la máxima tasa por la cual se puede transmitir información.

Rendimiento (*Throughput*): Es la tasa real a la que la información se transmite.

Latencia: Es el retardo entre el emisor y el receptor decodificándolo, usualmente es una función de las señales respecto al tiempo y el tiempo de procesamiento en cada nodo que atraviese la información.

***Jitter*:** Es la variación en el retardo de paquetes en el receptor de la información.

***Error rate*:** El número de bits corruptos expresados como una fracción del total enviado.

2.1.2 Problemas que los algoritmos de enrutamiento solucionan.

Principalmente existen dos parámetros de calidad de servicio que son directamente

afectados según el algoritmo de enrutamiento a utilizar: El *throughput* y el retardo promedio de paquetes (latencia). Cuando la carga de tráfico ofrecida por los sitios externos a la red es baja, el tráfico será completamente aceptado en la red, por lo que se puede resumir como:

***Throughput* = Carga ofrecida**

Cuando la carga ofrecida es excesiva, una porción será rechazada por el algoritmo de control de flujo por lo que el *throughput* se define por la siguiente definición:

***Throughput* = Carga ofrecida – Carga rechazada**

El tráfico experimenta una latencia promedio por paquete que dependerá en la ruta escogida por el algoritmo de enrutamiento. Además de esto, el *throughput* será afectado en gran medida por el algoritmo de enrutamiento debido al típico esquema de control de flujo, el cual se encarga de realizar un balance entre *throughput* y retardos. Por ejemplo, si un algoritmo percibe enormes cantidades de latencia en la red, comenzará a rechazar paquetes para aminorar la congestión, de modo que el *throughput* se ve reducido. Por lo tanto, se busca con un algoritmo de enrutamiento mantener bajos niveles de latencia (retardo) para maximizar el *throughput* en lo posible. De esta manera, el efecto de un buen algoritmo de enrutamiento se aprecia al observar una curva favorable de latencia-*throughput* mientras el control de flujo opera bajo situaciones de tráfico intenso.

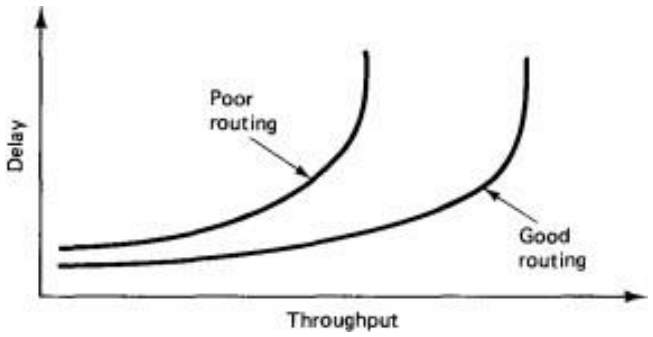

Figura 2-3. Curva *throughput*-delay para el caso de un buen enrutamiento y un mal enrutamiento

DATA NETWORKS (1992) [1]

2.1.3 Enrutamiento de la ruta más corta.

Existen varias aproximaciones para diseñar un algoritmo de enrutamiento, una de las principales se basa en la noción de la menor distancia entre dos nodos cualquiera dentro de una red. En el enrutamiento de la ruta más corta, la distancia se suele definir como un número positivo asignado a un enlace. De este modo, la ruta más corta que el algoritmo obtiene es aquel que contenga el menor valor de distancia para dos nodos determinados. Este tipo de algoritmo enruta cada paquete a una sola ruta seleccionada como la más corta. Los algoritmos más sofisticados incluyen el cambio de los valores de distancia en cada enlace de forma dinámica para situaciones donde se presente un tráfico intenso, de modo que se evita la congestión indicándole al control de flujo una nueva ruta más corta que no utilice el enlace congestionado. En esta familia de algoritmos tenemos a los ya conocidos algoritmos de Dijkstra y de Bellman-Ford, los cuales han sido implementados en varios protocolos conocidos como RIP, IGRP, OSPF y IS-IS.Según el tipo de algoritmo de enrutamiento que un protocolo implementa, estos pueden clasificarse en protocolos de Estado de Enlace (Link

State) y protocolos de Vector Distancia (Distance Vector). El resumen de las diferencias entre estos protocolos se da en la siguiente tabla:

Tabla 2 – 1. Comparación del enrutamiento entre enrutamiento de Vector Distancia y Estado de Enlace

DISTANCE VECTOR VS LINK STATE (2017) [5]

Attribute	Distance Vector Routing Protocol	Link State Routing Protocol
Example	RIP,EIGRP	OSPF, IS-IS
Convergence Time	High	Low
Configuration	Configuration is Simple	Configuration becomes complex for Large network
Routing Update	Sends the Entire routing table in each update. Update timer is 30- 90 seconds	Sends the Update when any changes (Link status) happens. There is no update timer.
Use	Good for Smaller network infrastructure	Good for any size of network infrastructure
Routing Loop	System might suffer routing loop if Routing is not configured properly	System doesn't suffer any routing loop
Routing Metric	Distance is used as Metric • RIP uses Hop Count • IGRP/EIGRP uses a composite of Bandwidth and Delay	Cost is used as Metric • OSPF uses Outgoing interface's link BW to calculate the cost.

2.1.4 Problemática del enrutamiento de ruta más corta.

Existen dos problemas principales en el caso del enrutamiento de la ruta más corta:

-Utiliza solo una ruta para cada par origen-destino, lo cual limita potencialmente el *throughput* de la red, ya que no se aprovechan todas las rutas disponibles a las cuales tenga acceso un nodo.

-Su capacidad de adaptarse a condiciones cambiantes de tráfico es limitado por su susceptibilidad a oscilaciones, esto sucede debido a cambios abruptos en el tráfico cuando

algunas rutas más cortas cambian debido a los cambios en las distancias de los enlaces.

Estos problemas pueden ser solucionados mediante el uso de otra aproximación para el diseño de algoritmos de enrutamiento llamado enrutamiento óptimo.

2.1.5 Enrutamiento óptimo.

El enrutamiento óptimo está basado en la optimización de una medida de performance similar al retardo promedio, de esta manera puede eliminar ambas desventajas mencionadas previamente partiendo el tráfico de cualquier par origen-destino en puntos estratégicos, además cambia el tráfico gradualmente entre rutas alternativas. La metodología para desarrollar este tipo de enrutamiento se basa en un estudio matemático de flujos óptimos y tiene grandes aplicaciones en los protocolos modernos.

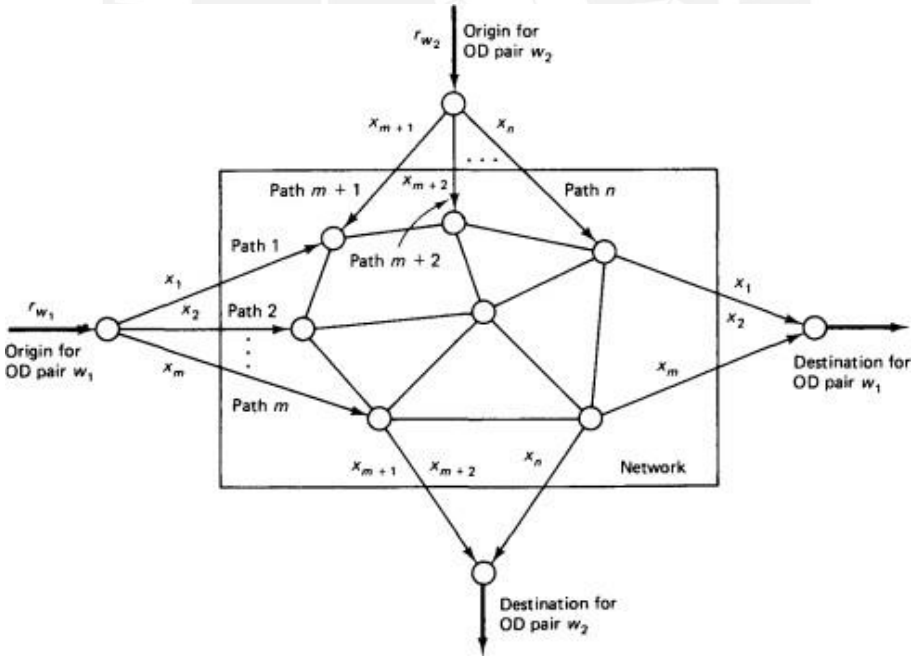

Figura 2-4. Esquema básico de funcionamiento de un algoritmo de enrutamiento óptimo.

El tráfico de diferentes pares origen-destino se divide a través de distintas rutas a lo largo de la red.

DATA NETWORKS (1992) [1]

2.2 Evaluación del Rendimiento de un Algoritmo de Enrutamiento

Para evaluar el rendimiento de un algoritmo de enrutamiento, se necesita cuantificar la noción de congestión de tráfico. Para ello se hace uso de los llamados modelos de flujo, los cuales son usados para formular problemas de enrutamiento óptimo.

Por otro lado, la congestión de tráfico en una red se puede modelar desde el punto de vista estadístico en función de los procesos de llegada de colas de redes. Debido a que no existe una expresión analítica que caracterice por completo los parámetros estadísticos presentes en una red de datos, se recurre a realizar el estudio en una red asumiendo ciertas condiciones.

2.2.1 Condiciones asumidas para el estudio matemático de un algoritmo.

Como se ha mencionado anteriormente, hallar una expresión analítica exacta para una red genérica es muy difícil de realizar por lo que se suelen asumir distintas condiciones:

-La estadística del proceso de llegada de paquetes a cada enlace cambia solo debido a actualizaciones de enrutamiento. Se realiza esta aproximación para medir la congestión en función del tráfico de datos promedio que maneja el enlace.

-La congestión se mide únicamente en función de la tasa de llegada de paquetes, la cual se puede medir en unidades de datos por segundo. Se usa esta aproximación para no hacer el cálculo muy tedioso respecto a distintas variables.

-La estadística del tráfico que ingresa a la red no cambia con el tiempo, esto sucede muy a menudo en redes donde los cambios de tráfico suelen darse de forma lenta, como por ejemplo redes donde un gran número de usuarios utiliza cada par de origen-destino de la red, de modo que el tráfico de cada uno de estos usuarios es pequeño con respecto al tráfico total

en la red.

Con estas condiciones asumidas como ciertas, y utilizando la aproximación adecuada, se puede definir una función costo para la cuantificación del rendimiento de un algoritmo de enrutamiento y el desarrollo de un algoritmo de enrutamiento óptimo.

2.2.2 Teoría de colas.

La teoría de colas es el estudio matemático de líneas de espera o colas, este modelo se suele usar con el fin de predecir el tiempo de espera y el tamaño de una cola y sus resultados son utilizados en distintas ramas de las ciencias.

Para el caso del estudio de redes de paquetes, la teoría de colas sirve para poder cuantificar la pérdida de paquetes, el *throughput*, entre otros parámetros de interés.

La teoría de colas hace uso de la notación de Kendall para poder describir cuantitativamente las condiciones que se asumirán para una cola determinada. Para el estudio de los algoritmos de enrutamiento óptimo, la aproximación para los nodos de una red es, en notación de Kendall, de una cola tipo M/M/1.

2.2.3 Colas M/M/1.

Una cola M/M/1 representa una cola en la que el sistema solo tiene un solo servidor, las llegadas son determinadas por un proceso de Poisson y los tiempos de servicio tienen una distribución exponencial. Este es el caso más simple de la teoría de colas y representa una gran simplificación para los cálculos de costo en algoritmos de enrutamiento.

2.2.4 Teorema de independencia de Kleinrock.

El teorema de independencia de Kleinrock afirma que mezclar distintas colas de tráfico

de paquetes sobre un mismo enlace tiene el efecto de volver independientes a los nodos que componen a la red. Esto significa que para las redes donde se mezcla tráfico sobre un mismo enlace, que es el mecanismo que propone el diseño de un algoritmo de enrutamiento óptimo.

2.2.5 Función costo para un algoritmo de enrutamiento.

Utilizando las condiciones establecidas en 1.2.1 y la aproximación de Kleinrock, se puede modelar la función costo de la siguiente forma:

$$\boldsymbol{Dij(Fij) = \frac{Fij}{Cij - Fij} + dij * Fij}$$

Figura 2-5. Función costo para una red bajo aproximación de Kleinrock.

DATA NETWORKS (1992) [1]

Donde:

Fij = Retardo de procesamiento entre nodos i y j

Cij = Capacidad de tranmisión del enlace que conecta los nodos

i y j Dij = Retardo de propagación de paquetes entre i y j

Con esta fórmula se obtiene una función costo que es igual al número promedio de paquetes en el sistema asumiendo que cada cola se comporta como una cola de paquetes M/M/1. Aunque para realizar este modelo se ha tenido que asumir varias condiciones, el valor de esta función costo es útil para poder cuantificar el rendimiento de un algoritmo de enrutamiento. En efecto, más allá de la comparación entre algoritmos optimizar esta función costo permite la aproximación matemática para el diseño de un algoritmo de enrutamiento óptimo.

2.3 Diseño de un Algoritmo de Enrutamiento Óptimo

El principal problema para un algoritmo de enrutamiento óptimo el de asignación de capacidad. Esto es, al momento de enviar paquetes desde un nodo a otro, decidir a través de qué enlaces asignar el tráfico a enviar, de modo que se aproveche cada enlace en lo posible y evitar congestionar una sola ruta. Para ello se buscará minimizar la función costo para la red con las asunciones que hemos tomado previamente.

2.3.1 Caracterización del enrutamiento óptimo.

Minimizar la función costo dependerá estrictamente de la cantidad de nodos y de enlaces que se disponga, a modo de ejemplo, se mostrarán los resultados de minimizar la función costo para una red extremadamente simple según la premisa de utilizar un algoritmo de enrutamiento óptimo.

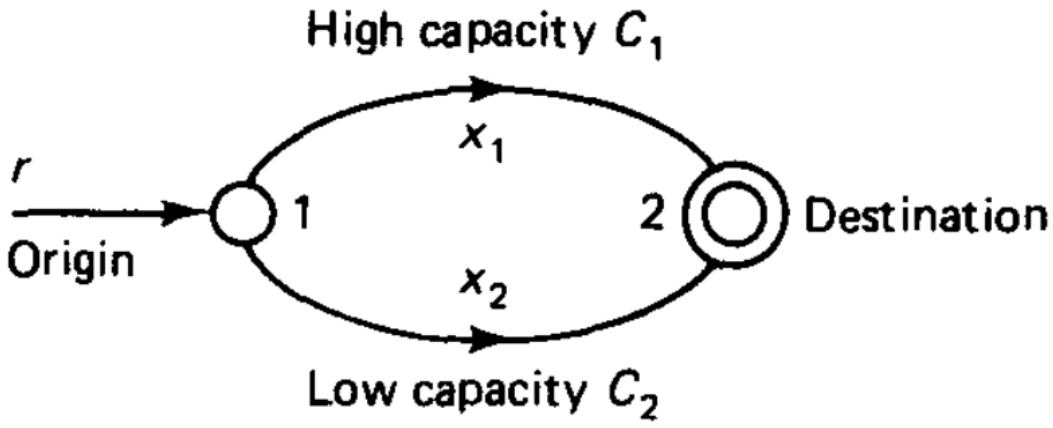

Figura 2-6. Red a analizar utilizando el mecanismo de enrutamiento óptimo minimizando la función costo

DATA NETWORKS (1992) [1]

Se puede observar que la red a analizar tiene un solo origen y un destino, además de ser enlazados a través de un enlace de alta capacidad y un enlace de baja capacidad. El

resultado se muestra a continuación.

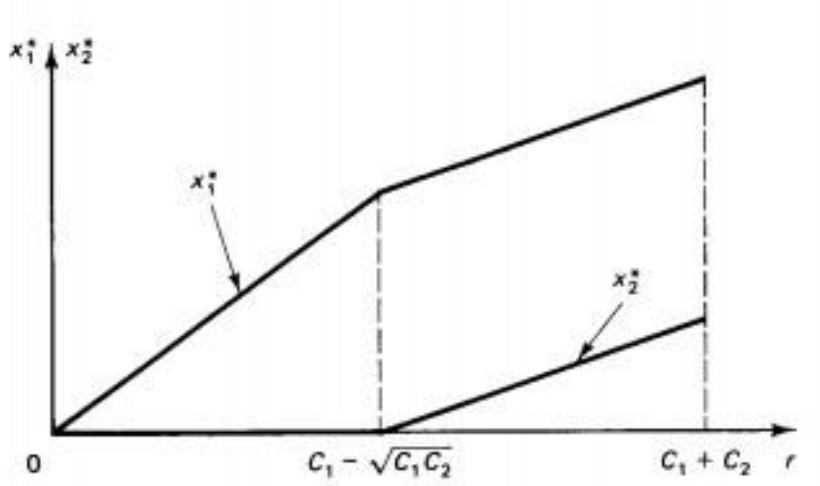

Figura 2-7. Red a analizar utilizando el mecanismo de enrutamiento óptimo minimizando la función costo

DATA NETWORKS (1992) [1]

Donde r es la cantidad de tráfico que ingresa a la red, como se puede apreciar, el flujo escogido por el enrutamiento óptimo muestra que para un tráfico inicial se haga uso del enlace con mayor capacidad, mientras que a partir de cierto valor límite, encontrado como solución matemática particularmente para el caso de la red mostrada, se comienza a utilizar el enlace de menor capacidad y de esta manera se divide el tráfico. Las redes modernas son mucho más complejas que el ejemplo actual mostrado, y el cálculo matemático se vuelve mucho más complicado y tedioso conforme se añaden más pares de origen-destino. Por lo tanto se recurre a soluciones computacionales que forman parte de estos algoritmos. Un ejemplo de implementación del algoritmo óptimo se muestra a continuación, considerando dos fuentes transmitiendo a un solo destino:

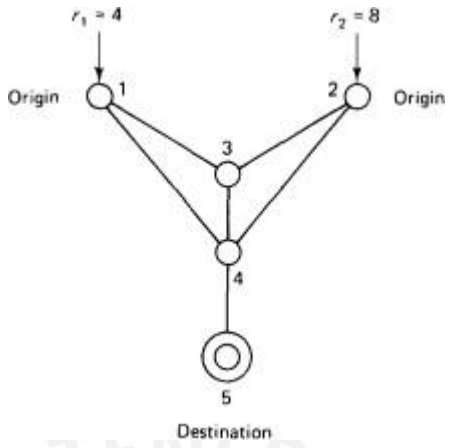

Figura 2-8. Red donde se analizará el efecto de un algoritmo de enrutamiento óptimo

DATA NETWORKS (1992) [1]

A continuación, se muestra una tabla con los resultados de implementar al algoritmo del método de proyección de gradientes (parte de la familia de los algoritmos de enrutamiento óptimo). Nótese que haciendo uso de pasos más pequeños (alfa sub-k), se logra reducir la métrica de los enlaces mientras que al mismo tiempo, existe una tendencia a realizar más iteraciones (k):

Tabla 2-2. Evolución del costo de los enlaces al aplicar un algoritmo de enrutamiento óptimo

DATA NETWORKS (1992) [1]

k	$\alpha^k \equiv 0.5$	$\alpha^k \equiv 1.0$
0	184.00	184.00
1	110.88	104.00
2	102.39	101.33
3	101.28	101.03
4	101.09	101.00
5	101.03	
6	101.01	
7	101.00	
8		
9		
10		

2.4 Enrutamiento Multi-camino

Como se ha mencionado en este capítulo, el enrutamiento óptimo se propone como una mejora de enrutamiento a comparación de los algoritmos utilizados en protocolos de enrutamiento tradicionales. La diferencia principal del enrutamiento óptimo con el enrutamiento tradicional es el de poder enviar paquetes de datos de un origen a un destino a través de diferentes rutas. Esta característica específica corresponde a la familia de algoritmos de enrutamiento denominados multi-camino. Para poder utilizar una red a su máximo potencial, es necesario hacer uso de tecnología multi-camino [22] [23].

2.5 Enrutamiento en SDN

Múltiples formas de implementación de algoritmos de enrutamiento han sido propuestos en SDN dada la utilidad que tiene para la gestión de redes complejas y a gran escala [13], cada uno variando en el tipo de algoritmo implementado y el código utilizado dependiendo de las características del lenguaje de programación sobre el cual el controlador

esté definido.

Para el caso específico de Floodlight v1.2, se tiene que el algoritmo de enrutamiento predeterminado es Dijkstra, sin embargo, este se puede modificar para poder un enrutamiento con características distintas (como por ejemplo el *throughput*). Un caso del implementación de código para el módulo de enrutamiento de Floodlight fue publicado en junio del 2020 [14], reportando cambios en los valores de *throughput* y el costo asignado a los enlaces tanto en el algoritmo de Dijkstra como en el nuevo algoritmo implementado. De esta manera, se pueden analizar las diferentes aproximaciones que se puedan dar a la implementación de un algoritmo de enrutamiento para un controlador SDN.

2.5.1 Módulos de enrutamiento en Floodlight.

Como se ha mencionado, Floodlight realiza el enrutamiento haciendo uso de módulos escritos en lenguaje de programación Java, la modificación del código de estos módulos permite realizar cambios en la lógica utilizada para el enrutamiento así como el procedimiento utilizado por el controlador. Entre los módulos utilizados para poder llevar a cabo el enrutamiento cabe destacar el módulo TopologyManager.java y TopologyInstance.java

2.5.2 TopologyManager.java.

Este módulo se encarga de mantener la noción del controlador acerca del grafo utilizado para la red, así como también se encarga de implementar herramientas para encontrar rutas a través de la topología [15]

2.5.3 TopologyInstance.java.

Según la explicación añadida en el código del repositorio de Github, se trata de una representación de la topología de red, que es utilizada por TopologyManager.java [16]. Sin embargo, en otros artículos de investigación, se le define como el módulo encargado de obtener los caminos con costo mínimo entre los nodos de manera regular [14].

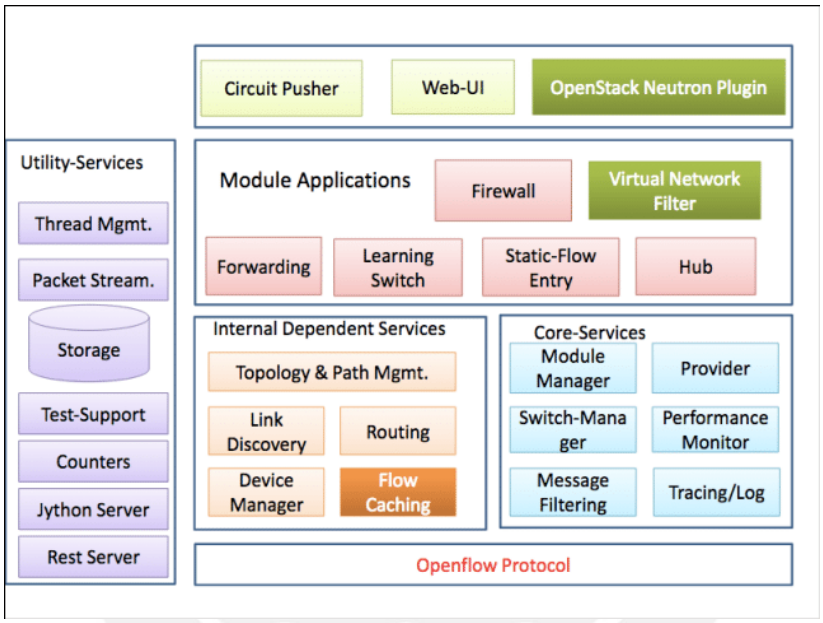

Figura 2-9. Arquitectura de un controlador Floodlight mostrando los módulos utilizados para los diferentes servicios

EXPERIMENTING WITH SCALABILITY OF FLOODLIGHT CONTROLLER IN SDN [17]

2.6 Cambio de Módulo de Enrutamiento en Floodlight

Un artículo publicado en junio del 2020 [14], indicó el procedimiento y análisis posterior luego de cambiar el algoritmo de Dijkstra utilizado de forma predeterminada por Floodlight por un algoritmo Dual Ascent, el estudio de los resultados se mostrará en el

capítulo 3.

2.6.1 Cambio a algoritmo Dual Ascent.

En general se utilizan diferentes variaciones de este algoritmo pero para el artículo mencionado, se hizo uso de una aproximación que puede ser entendida como trabajar sobre el dual de una formulación de programa lineal [18].

En primera instancia, el algoritmo utiliza una variable "grafo" que se va expandiendo en cada iteración conforme va añadiendo los diferentes subconjuntos de nodos que existan en la topología, luego, va reduciendo el costo de los enlaces desde el nodo origen y el nodo destino haciendo uso del camino más corto entre ellos y la tabla de costo de enlaces. Finalmente, se obtiene un grafo permanente con una estructura de datos de árbol, el cual incluye el camino más corto entre los nodos origen y destino. La segunda parte del algoritmo intenta encontrar el camino exacto tomando como punto de partida la última estructura de árbol donde se eliminan los nodos que no tengan enlace común con el resto del árbol.

Para el caso de implementación mencionado, se construyó un módulo DualAscent.java de manera similar a la función de Dijkstra en Floodlight, de modo que hacía uso del mismo número y tipo de variables y obtenía las variables de salida de las mismas formas y estructuras.

2.7 Emuladores de Red

Se debe considerar que los resultados obtenidos para los artículos mencionados a lo largo del presente capítulo se han implementado en redes virtuales con soporte SDN. Existen varias aplicaciones para distintos sistemas operativos que se encargan de emular simular

una red virtualizada, entre las más populares, tenemos a GNS3 para el uso de redes legacy, el mismo que también puede añadir librerías para tener soporte de redes SDN. Por otro lado, Mininet, un programa con distribución en Linux, ha adquirido popularidad como software para pruebas de varias implementaciones SDN en artículos. Su principal ventaja radica en la naturaleza de código abierto pues posee una gran documentación y soporte para diferentes equipos de red gracias a la actividad de la comunidad que lo utiliza [19].

La ventaja de la emulación haciendo uso de emuladores de red es que poseen una curva de aprendizaje menor a la del proceso matemático y que se acercan más a los resultados que se podrían obtener en caso de realizar una implementación de la solución a equipos reales. La desventaja es el gran consumo de recursos que requiere el hardware cuando se desean simular grandes cantidades de tráfico y redes mucho más complejas.

2.8 Enrutamiento Multi-camino en SDN

Entre los algoritmos de enrutamiento que se pueden implementar en SDN, también se encuentra la posibilidad de implementar enrutamiento multi-camino. Un artículo publicado el 2018 mostró que es posible implementar enrutamiento de este tipo en soluciones SDN al implementar en el controlador un mecanismo de partición estocástico con el fin de dividir un flujo de datos en distintos componentes [24]. Asimismo, se procedió a comparar los resultados con el enrutamiento OSPF, que utiliza una sola ruta para enviar datos desde un origen a un destino. Los resultados del análisis del impacto en la red se detallarán en el siguiente capítulo.

3. Análisis de los Resultados

En el presente capítulo se hará un análisis de los cambios en los parámetros de red luego de cambiar el algoritmo de enrutamiento basándose en los resultados obtenidos en los artículos donde se mostraron casos de implementación de algoritmos de enrutamiento en soluciones SDN [14][24].

3.1 Topología Utilizada

La prueba experimental fue llevada a cabo en una topología de tipo malla con el propósito de generar varias rutas desde un nodo a otro en la topología.

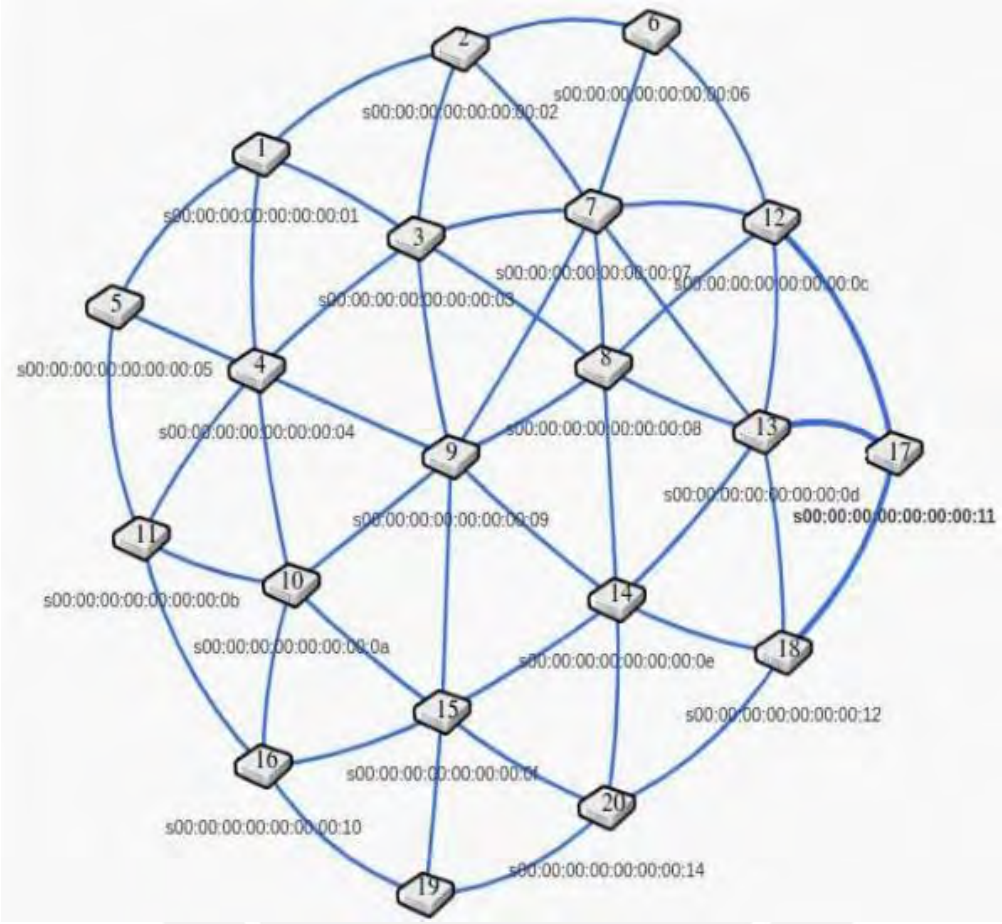

Figura 3-1. Topología de prueba para la implementación del algoritmo Dual Ascent

CHANGING ROUTING MODULE IN FLOODLIGHT [14]

Luego, se generaron 15 segundos de tráfico TCP con el comando Iperf en Mininet. Finalmente se procedió a analizar el costo promedio de los algoritmos así como el *throughput* obtenido.

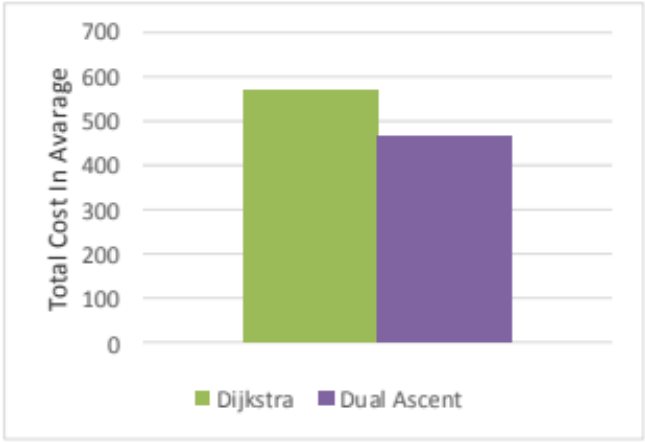

Figura 3-2. Comparación de costos de algoritmos

CHANGING ROUTING MODULE IN FLOODLIGHT [14]

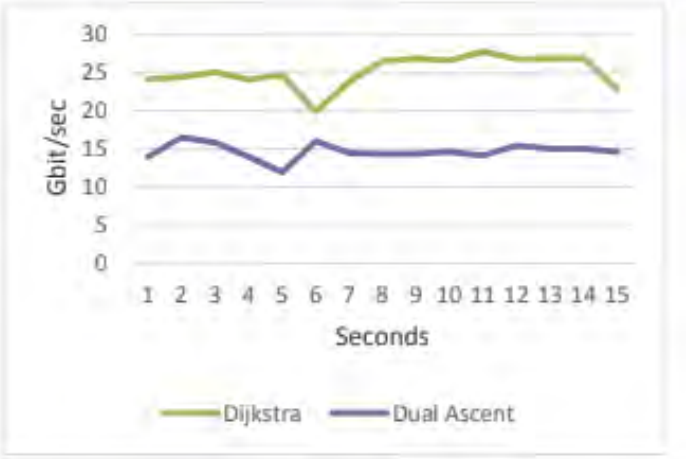

Figura 3-3. Comparación de *throughput* obtenido por los algoritmos

CHANGING ROUTING MODULE IN FLOODLIGHT [14]

3.2 Impacto en la Red

Como se ha podido observar, el algoritmo Dual-Ascent muestra una mejoría respecto al algoritmo Dijkstra respecto al costo del algoritmo, sin embargo, muestra menores niveles de *throughput* respecto al enrutamiento por defecto. Es de tener en consideración que a diferencia del algoritmo Dual-Ascent, el algoritmo de Dijkstra es ya bastante conocido y ha sido optimizado con el paso del tiempo para dar los mejores resultados, por lo que se debe considerar utilizar formas optimizadas del algoritmo Dual-Ascent de modo que se obtengan mejores resultados.

3.3 Resultados de Implementación de Enrutamiento Multi-camino en SDN

Esta sección muestra los resultados del impacto en una red SDN al utilizar enrutamiento multi-camino y compararlo con un protocolo de enrutamiento tradicional como OSPF.

3.3.1 Configuración para la simulación.

La simulación se llevó a cabo utilizando el software Network Simulator (ns-3), donde un switch se configuró para que gestione el mecanismo de separación de tráfico, la formulación de los algoritmos implementados se resolvió utilizando la aplicación GNU Linear Programming Kit (GLPK).

3.3.2 Topología utilizada.

Se utilizaron dos topologías para poder realizar la comparación entre algoritmos de enrutamiento con el fin de evaluar el desempeño de cada uno en diferentes tipos de red.

3.3.2.1 Topología diamante.

Este tipo de topología se usó con el objetivo de demostrar los beneficios de usar todos

los enlaces de la red para poder satisfacer las demandas de un flujo determinado, sin embargo las pruebas en esta topología solo implicaban un flujo de datos y no se podía aprovechar la ventaja de SDN de manejar múltiples flujos.

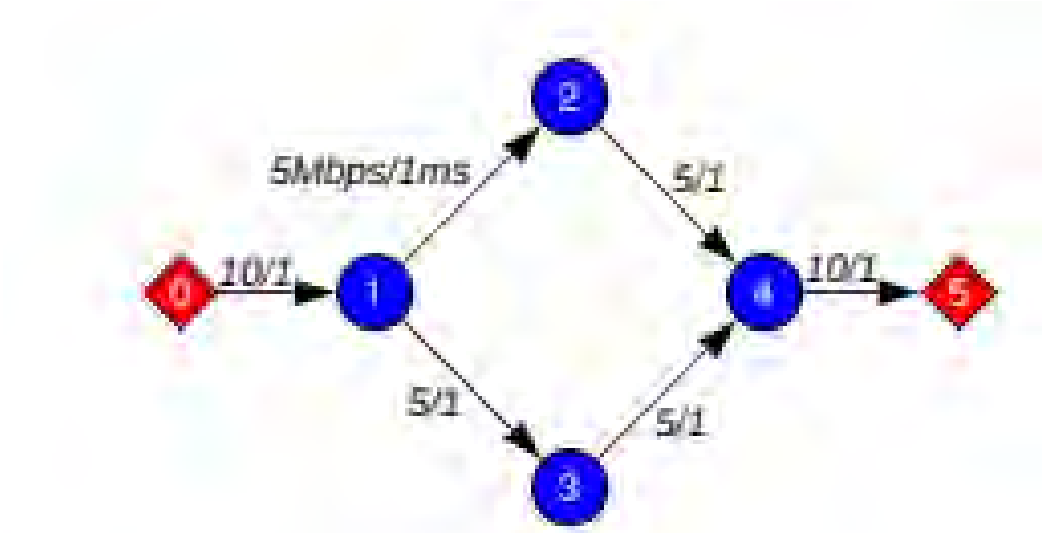

Figura 3-4. Topología diamante, utilizada para la comparación del enrutamiento multi-camino y OSPF, los círculos son switches y los diamantes son terminales.

A GLOBALLY OPTIMISED MULTIPATH ROUTING ALGORITHM USING SDN (2018) [24]

3.3.2.2 Topología mariposa.

Esta topología se propuso con el fin de evaluar el enrutamiento en casos de más de un flujo, esta topología se caracteriza por poseer un enlace cuello de botella y múltiples rutas entre origen y destino.

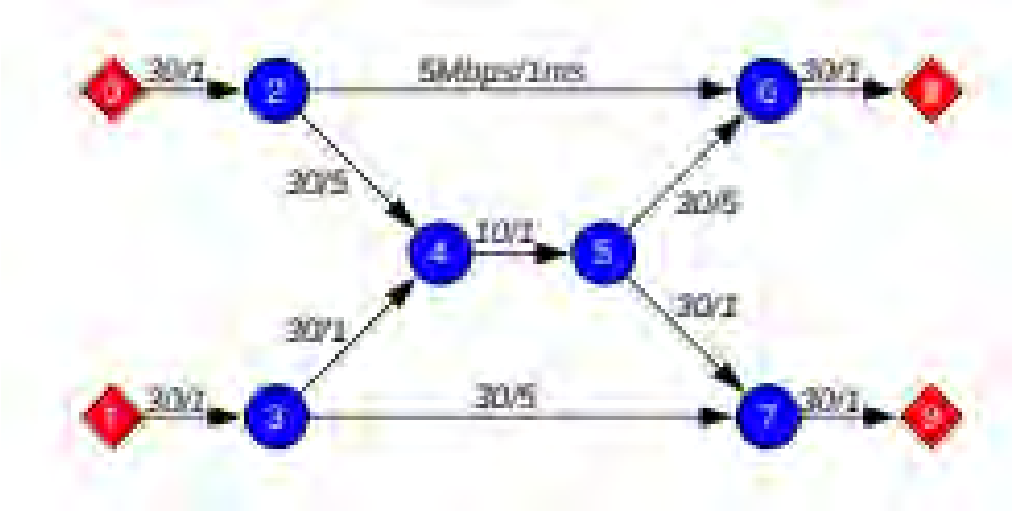

Figura 3-5. Topología mariposa, utilizada para la comparación del enrutamiento multi-camino y OSPF teniendo más de un flujo

A GLOBALLY OPTIMISED MULTIPATH ROUTING ALGORITHM USING SDN (2018) [24]

3.4 Comparación de los Algoritmos de Enrutamiento en Topologías

A continuación, se mostrarán los resultados de la comparación del enrutamiento multi-camino vs OSPF en las diferentes topologías:

Tabla 3-1. Resultados obtenidos en topología diamante

ELABORACIÓN PROPIA, BASADO EN EL ARTÍCULO "A GLOBALLY OPTIMISED MULTIPATH ROUTING ALGORITHM USING SDN" [24]

Tasa de tx (Mbps)	*Throughput* OSPF (Mbps)	*Throughput* Algoritmo multi-camino (Mbps)	*Jitter* OSPF (ms)	*Jitter* Algoritmo multi-camino (ms)	Latencia OSPF (ms)	Latencia Algoritmo multi-camino (ms)
5	5	5	0	0	11.2	11.2
10	5	9.99	1.2	106	60000	152

Como se puede observar, el enrutamiento multi-camino muestra una mayor ventaja frente al enrutamiento OSPF tanto en nivel de *throughput* como latencia al costo de cierto nivel de *jitter*. Esto es de esperar puesto que OSPF solo hace uso de una sola ruta, creando un cuello de botella fijo. Por otra parte, el enrutamiento multicamino hace que exista contribución de la latencia a través de los distintos caminos utilizados para enviar los paquetes, por lo que es esperable.

De la misma manera, se adjunta a continuación los resultados de la evaluación de los algoritmos en la topología mariposa.

Tabla 3-2. Resultados obtenidos en topología mariposa

ELABORACIÓN PROPIA, BASADO EN EL ARTÍCULO "A GLOBALLY OPTIMISED MULTIPATH ROUTING ALGORITHM USING SDN" [24]

Throughput OSPF (Mbps)	*Throughput* Algoritmo multi-camino (Mbps)	*Jitter* OSPF (ms)	*Jitter* Algoritmo multi-camino (ms)
5	9.95	1.2	19
20	20	0	7.57

Una vez más, se observa que un algoritmo multi-camino obtiene mejores niveles de *throughput* a comparación de OSPF en situaciones donde sería necesario agregación de enlaces solo para mejorar el servicio.

4. Aplicabilidad de Algoritmos

En el presente capítulo se mostrarán algunas recomendaciones respecto a los resultados observados en la bibliografía observada, así como una orientación a los planes de trabajo futuro.

4.1 Reflexión

En un entorno donde las redes crecen cada vez más y se hace imperativa la necesidad de gestionar mejor los recursos de red así como aprovechar la capacidad de los enlaces y mejorar la calidad del servicio desde el punto de vista del cliente, el uso de las soluciones SDN se va popularizando con el paso del tiempo por el control que brinda y la facilidad de modificar diversos aspectos de red desde el punto de vista del administrador. Además, a lo largo del presente trabajo de investigación se ha observado que parámetros de calidad de servicio como el *throughput* y latencia quedan influenciados fuertemente por los algoritmos de enrutamiento utilizados en el controlador SDN, por lo que es deseable realizar la implementación de un algoritmo que pueda optimizar estos valores, como es el caso del algoritmo óptimo, el cual aprovecha el envío de paquetes a través de distintas rutas con el fin de aprovechar los enlaces disponibles en la red.

Cabe resaltar que en los artículos de investigación, el uso de algoritmos de enrutamiento multi-camino no son obligatorios pero sí se trata de un tema en investigación constante. Un ejemplo de un artículo de investigación de implementación de un algoritmo es el de Dual-Ascent [14] que a pesar de no ser un algoritmo multi-camino, muestra resultados de mejora a nivel de *throughput* además de dar paso a futuras investigaciones optimizando este tipo de algoritmos.

4.2 Aplicabilidad y Futuro Trabajo

El trabajo de investigación presentado da paso a la evaluación y cuestionamiento de los resultados que se pueden obtener cambiando el algoritmo de enrutamiento o la topología sobre la cual se evalúe. Entre los casos de algoritmos presentados en la bibliografía revisada, el algoritmo de enrutamiento multi-camino resulta el más llamativo pues se acercan al concepto de

enrutamiento óptimo, que es el enrutamiento deseado para poder aprovechar al máximo los recurso de una red/topología dada.

Para realizar ello se debe tener en cuenta una topología determinada sobre la cual realizar la implementación de la solución SDN así como el tipo de algoritmo de enrutamiento a utilizar. De la misma manera, se necesita definir la plataforma de simulación/emulación de red, la capacidad de los enlaces de la topología, el tipo de tráfico que los terminales generarán, etc. El Grupo de Investigación de Redes Avanzadas (GIRA-PUCP) cuenta con los recursos necesarios para poder realizar este tipo de investigación tanto a nivel virtual como equipos físicos, permitiendo altos niveles de fidelidad y realismo.

Conclusiones

- Las soluciones SDN permiten al administrador de red tener mayor visibilidad y control sobre los recursos de red que se gestionan, puesto que la configuración de todos los equipos de red se ubican en un dispositivo central llamado controlador.
- Los algoritmos de enrutamiento utilizados en mecanismos de enrutamiento tienen una fuerte correlación con los parámetros de Calidad de Servicio (QoS) como el *throughput* y latencia percibidos desde el punto de vista del cliente.
- El enrutamiento óptimo es el tipo de enrutamiento que se busca para poder obtener el mejor provecho de los recursos de una red determinada. La característica principal de este tipo de enrutamiento es el poder enviar paquetes a través de distintas rutas dado un flujo de origen-destino.
- Los algoritmos de enrutamiento multi-camino, son los algoritmos más atractivos para poder realizar una aproximación al algoritmo de enrutamiento óptimo. Los resultados obtenidos al compararlo con un protocolo de enrutamiento que hace uso de una sola ruta para enviar paquetes como OSPF en un entorno virtual SDN han demostrado que se alcanzan mejores valores de *throughput* y *jitter* para diferentes topologías.

Bibliografía

[1] R. Gallagher y D. Bertsekas, "Routing," en Data Networks, 2da ed, New Jersey, NJ, USA: Prentice Hall Inc., 1992, cap. 5 [En línea]. Disponible: https://web.mit.edu/dimitrib/www/datanets.html

[2] G. Cuba y J. Becerra, "Diseño e implementación de un controlador SDN/Openflow para una red de campus académica, " tesis de título profesional, Dep. Ing. Telecom. - PUCP, Lima, Lima, Perú, 2015 [En línea]. Disponible: : http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/7149

[3] N. Urena. "Advantages of Software Defined Networking (SDN)." FIDELUS.com https://www.fidelus.com/software-defined-networking-advantages/ (Consultado Ene. 23, 2021)

[4] R. Margaret. "What is Software-Defined Networking (SDN)?" TECHTARGET.com https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN (Consultado Ene. 23, 2021)

[5] Ariq. "Distance Vector vs Link State." NETWORKCHEFBD.com https://networkchefbd.com/routing-protocols/distance-vector-vs-link-state/ (Consultado Nov. 20, 2020)

[6] L. Mamushiane, J. Mwangama, A. Albert, "Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?,"

presentado el 2018 en la conferencia IEEE NFV-SDN. Verona, Italia, Italia, Nov 27-29, 2018, Paper 8725710

[7] D. Tatang, et al., "SDN-Guard: Protecting SDN Controllers Against SDN Rootkits, " presentado el 2017 en la conferencia IEEE NFV-SDN. Berlín, Alemania, Dic 11, 2017, Paper 8169856

[8] M. Alsaeedi, M. Mohamad, A. Al-Roubaiey, "Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey, " en IEEE Access, Ago. 2019. [En línea]. Disponible: https://ieeexplore.ieee.org/document/8784036

[9] C. Park, C. Hwang, K. Jeong, "A telco's perspectives: Open and flexible transport SDN, " presentado el 2015 en la 17va conferencia APNOMS. Busan, Surcorea, Ago. 19 - 21, 2015, Paper 7275413

[10] C. Tselios, I. Politis, S. Kotsopoulos, "Enhancing SDN security for IoT-related deployments through blockchain, " presentado el 2017 en la conferencia IEEE NSV-SDN. Berlín, Alemania, Nov. 6 - 8, 2017, Paper 8169860

[11] SDx Central Staff. "What is Software Defined Networking? Definition." SDXCENTRAL.com https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/ (Consultado Ene. 23, 2021)

[12] Open Networking Foundation. "SDN Migration Considerations and Use Cases ." OPENNETWORKING.org https://www.opennetworking.org/wp-content/uploads/2014/10/sb-sdn-migration-use-cases-pdf (Consultado Ene. 23, 2021)

[13] M. Fatih, E. Karaarslan, C. Güngor. (Set. 2016). "A Preliminary Survey on the Security of Software-Defined Networks, " Presentado en la conferencia ICAT '16. [En línea] Disponible: https://www.researchgate.net/publication/320226086_A_Preliminary_Survey_on_the_Security_of_Software-Defined_Networks

[14] H. T. Ilhan y D. Yiltas-Kaplan, "Changing Routing Module in Floodlight," Conferencia Internacional ICECCE. Estambul, Turquía, 2020, pp. 1-5, doi:10.1109/ICECCE49384,2020.9179298.

[15] GitHub, "TopologyManager.java " GITHUB.com https://github.com/floodlight/floodlight/blob/master/src/main/java/net/floodlightcontroller/topology/TopologyManager.java (Consultado Ene. 23, 2021)

[16] GitHub. "TopologyInstance.java. " GITHUB.com https://github.com/floodlight/floodlight/blob/master/src/main/java/net/floodlightcontroller/topology/TopologyManager.java (Consultado Ene. 23, 2021)

[17] S. Asadollahi y B. Goswami (Dic. 2017). "Experimenting with

scalability of floodlight controller in software defined networks, " Presentado el 2017 en ICEECCOT. [En línea] Disponible: https://www.researchgate.net/publication/323057223_Experimenting_with_scalability_of_floodlight_controller_in_software_defined_networks

[18] M. X. Goemans y D. P. Williamson, "The primal-dual method for approximation algorithms and its application to network design problems", In Approximation algorithms for NP-hard problems, pages 144-191. PWS Publishing Co., Boston, MA, USA, 1996. [En línea]. Disponible: https://math.mit.edu/~goemans/PAPERS/book-ch4.pdf

[19] Mininet Team. "Mininet Credits/Story. " MININET.org http://mininet.org/credits/ (Consultado Ene. 23, 2021)

[20] Project Floodlight. "Module applications. " FLOODLIGHT.ATLASSIAN.net https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343509/Module+Applications (Consultado Ene. 23, 2021)

[21] E. Akin y A. Turgay, "Comparison of Routing Algorithms with Static and Dynamic Link Cost in Software Defined Networking (SDN)" en IEEE Access, Oct. 2019. [En línea] Disponible: https://www.researchgate.net/publication/336449420_Comparison_of_Routing_Algorithms_With_Static_and_Dynamic_Link_Cost_in_Software_Defined_Networking_SDN

[22] S. K. Singh, T. Das y A. Jukan, "A Survey on Internet Multipath Routing and Provisioning," IEEE Communications Surveys Tutorials, vol. 17, num. 4, pp. 2157-2175, 2015

[23] S. Habib, J. Qadir, A. Ali, D. Habib, M. Li y A. Sathiaseelan,, "The Past, Present, and Future of Transport-Lay"

[24] N. Farrugia, V. Buttigieg y J. A. Briffa, "A globally optimised multipath routing algorithm using SDN," 21va Conferencia ICIN del 2018, Paris, 2018, pp. 1-8, doi: 10.1109/ICIN.2018.8401633

[25] Open Networking Foundation, "SDN Architecture. " OPENNETWORKING.org https://opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf (Consultado Ene. 23, 2021)

[26] A. Keziah, K. Murugan y K. Rajarajeshwari. "Performance analysis of wireless trusted software defined networks" en IRJET, Set. 2017. [En línea]. Disponible: https://www.researchgate.net/publication/323057223_Experimenting_with_scalability_of_floodlight_controller_in_software_defined_networks (Consultado Ene. 23, 2021)