Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias en Electrónica y Telecomunicaciones con orientación en Telecomunicaciones

---

Modelado de una arquitectura de red definida por software (SDN) para el aprovisionamiento de recursos utilizando Cross-Layer Design (CLD)

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de
Maestro en Ciencias

Presenta:

Ernesto Cosío Velázquez

Ensenada, Baja California, México
2017

Tesis defendida por

Ernesto Cosío Velázquez

y aprobada por el siguiente Comité

Dr. Raúl Rivera Rodríguez
Director de tesis

M. C. Jorge Enrique Preciado Velasco
Co-director de tesis

Miembros del comité

Dr. Salvador Villareal Reyes

Dr. Raúl Tamayo Fernández

Dr. Álvaro Armenta Ramade

M.Ing. José Eleno Lozano Rizk

Dr. Miguel Ángel Alonso
Coordinador del Posgrado en Electrónica y Telecomunicaciones

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta **Ernesto Cosió Velázquez** como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS en Electrónica y Telecomunicaciones con orientación en Telecomunicaciones.

Modelado de arquitectura de red definida por software (SDN) para el Aprovisionamiento de recursos utilizando Cross-Layer Design

Resumen aprobado por:

_____	_____
Dr. Raúl Rivera Rodríguez	**M. C. Jorge Enrique Preciado Velasco**
Director de Tesis	**Co-Director de Tesis**

Las redes de datos se han convertido en la espina dorsal de las telecomunicaciones, convergiendo las redes inalámbricas, redes de telefonía fija y celular, entre otras, en una sola. Esto ha hecho que las redes de datos hayan llegado a tener un uso cotidiano e indispensable, de manera que se pueden emplear tanto en aplicación de entretenimiento y comunicación, como en aplicaciones industriales, médicas y educativas. Las redes de datos convencionales al igual que las redes definidas por software, no utilizan ni contemplan mecanismos de calidad de servicio en sus arquitecturas, a pesar de que existen diferentes opciones y herramientas. Uno de los principales retos de la prestación de servicios en la nube es desplegar un control coordinado de recursos para las aplicaciones y la red, con el fin de proporcionar una entrega adaptativa del servicio de datos, basados en el conocimiento recíproco entre los requisitos de las aplicaciones y las capacidades de la red. Los servicios y aplicaciones requieren recursos de la red para operar de manera eficiente, lo cual no siempre se logra por las limitaciones que se tienen en la red. La aportación de este documento se basa en la aproximación de las latencias y retardos en la red como monitoreo de bajo nivel que utiliza la mensajería OpenFlow para estimar estos parámetros. Así como el optimizador CLD que hace las funciones de abstracción de los parámetros en la red y los requerimientos de las aplicaciones. Además de un mecanismo simple de selección de trayectoria con base en las métricas de QoS en la red y lo requerido por la aplicación, que evita la selección de la ruta más corta por omisión. El algoritmo propuesto es funcional, mejora el desempeño de la aplicación, dando mayor agilidad al flujo de información, reduce las pérdidas de paquetes, por lo tanto reduce el tiempo que se requiere para realizar una tarea. La selección de la ruta se realiza con base en las condiciones de la red, así como los requerimientos de las aplicaciones y poder proveer los recursos para el desempeño óptimo de las mismas.

Palabras clave: SDN, Cross-Layer, QoS.

Esta página se ha dejado intencionalmente en blanco.

Abstract of the thesis presented **by Ernesto Cosío Velázquez** as a partial requirement to obtain the Master of Science degree in Electronics and Telecommunications with orientation in Telecommunication.

Modeling software-defined network (SDN) architecture for resource provisioning using Cross-Layer Design

Abstract approved by:

_____	_____
Dr. Raúl Rivera Rodríguez	**M. C. Jorge Enrique Preciado Velasco**
Thesis Director	**Thesis Co-Director**

Data networks have become the backbone of telecommunications, converging wireless networks, fixed and cellular networks, among others, into one. This has made the data networks have a daily and indispensable use, so that they can be used both in entertainment application and communication, as in industrial, medical and educational applications. Conventional data networks, like software-defined networks, do not use or contemplate quality-of-service mechanisms in their architectures, although different options and tools exist. One of the major challenges of delivering cloud services is to deploy coordinated resource control for applications and the network to provide adaptive delivery of the data service, based on reciprocal knowledge among Applications and network capabilities. Services and applications require network resources to operate efficiently, which is not always achieved due to network limitations. The contribution of this document is based on the approximation of the latencies and delays in the network as low level monitoring that uses the OpenFlow messaging to estimate these parameters. As well as the CLD optimizer that does the abstraction functions of the parameters in the network and the requirements of the applications. In addition to a simple path selection mechanism based on the QoS metrics in the network and what is required by the application, it avoids the selection of the shortest path by default. The proposed algorithm is functional, improves the performance of the application, giving greater agility to the information flow, reduces packet losses, thus reducing the time required to perform a task. The selection of the route is made based on the conditions of the network, as well as the requirements of the applications and to be able to provide the resources for the optimal performance of the same ones.

Keywords: SDN, Cross-Layer, QoS.

Esta página se ha dejado intencionalmente en blanco.

Dedicatoria

Este trabajo se lo dedico:

A mis padres Ernesto Cosío Ibarra y María Guadalupe Velázquez Torres, por todo el apoyo, comprensión y cariño que siempre me han brindado para alcanzar mis metas.

A mi hermana Erika Cosío Velázquez y su esposo Gustavo Alonso García Trujillo, por su comprensión, apoyo y consejos en esta etapa de aprendizaje.

Esta página se ha dejado intencionalmente en blanco.

Agradecimientos

*A los maestros de **CICESE**, por todos los conocimientos y experiencias compartidas durante los cursos y materias, así como su disponibilidad para atender siempre cualquier duda o problema que se presentara.*

*Se agradece al **Consejo Nacional de Ciencia y Tecnología (CONACYT)** por la beca de estudios de posgrado otorgada con el CVU 529485.*

*Gracias a los miembros del comité de tesis, **Dr. Salvador Villareal**, **Dr. Raúl Tamayo**, **Dr. Álvaro Armenta** y **M.C. Jorge Preciado** por sus valiosas aportaciones y comentarios a este trabajo.*

*Un especial agradecimiento a mi director de tesis al **Dr. Raúl Rivera** y asesor **M. C. José Lozano** por su apoyo, asesoría y sugerencias brindadas durante el desarrollo de esta tesis de maestría.*

*A mis compañeros y amigos, **Guillermo González Martínez**, **Jairo Donlucas Saldivar**, **Juan Ramón Solís Escobedo, Jesús Ismael Casarrubias Garín** y **Andrea Murillo Carrillo**, por su amistad, apoyo y consejos en estos 2 años.*

Esta página se ha dejado intencionalmente en blanco.

Tabla de Contenido

Tabla de Contenido (continuación)

Lista de figuras

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de figuras (continuación)

Lista de tablas

Esta página se ha dejado intencionalmente en blanco.

Capítulo 1. Introducción

Las redes de datos se han convertido en la espina dorsal de las telecomunicaciones, convergiendo las redes inalámbricas, redes de telefonía fijas y móviles, entre otras, en una sola. Esto ha hecho que las redes de datos hayan llegado a tener un uso cotidiano e indispensable, de manera que se pueden emplear tanto en aplicaciones para telecomunicaciones, como en aplicaciones industriales, médicas, educativas y de entretenimiento, por mencionar algunas.

Las redes de datos suelen ser utilizadas por dispositivos de usuarios finales o equipos interconectados a través de una infraestructura de red. Esta infraestructura es compartida y emplea elementos de conmutación, tales como enrutadores (routers) y conmutadores (switches).

Las redes de datos emplean protocolos que se ejecutan de manera distribuida para tomar decisiones, como es el caso de los algoritmos de enrutamiento, en donde la arquitectura de la red utilizada en la Internet trabaja comúnmente en el esquema denominado mejor esfuerzo (Clark & Fang, 1998). Dentro de este esquema, como su nombre lo indica, el algoritmo realiza el mejor esfuerzo para que la información llegue a su destino, sin embargo, no se ofrecen garantías de que el flujo de información en la red cuente con las condiciones necesarias para que las aplicaciones tengan el rendimiento deseado, que permita un funcionamiento sin interrupciones, por ejemplo, en aplicaciones interactivas, como VOIP, videoconferencias etc.

Debido a que en una red de datos pueden transmitirse diferentes tipos de tráfico, es deseable que la infraestructura de la red soporte mecanismos para proporcionar diferentes niveles de calidad de servicio (Quality of Service, QoS) hacia el tráfico de diferentes aplicaciones. Para este efecto, el Internet Engineering Task Force (IETF) ha explorado varias propuestas de calidad de servicio (H.E. Egilmez & Dane, 2012), sin embargo éstas no han sido adoptadas en su totalidad por proveedores de servicio de Internet o incluso al interior de los campus universitarios por mencionar un ejemplo (H.E. Egilmez & Dane, 2012).

La falta de implementación se debe a que las propuestas de calidad de servicio, tales como IntServ y Diffserv se construyen en la parte superior de las capas del modelo de Internet (capa de aplicación, presentación, sesión), sin embargo el enrutamiento por saltos que es utilizado carece de una visión amplia de los recursos globales de la red. Existen protocolos alternativos, como la conmutación de etiquetas multiprotocolo (MPLS, por sus siglas en inglés) que ofrece una solución parcial a través de su

capacidad de conmutación ultra-rápida, sin embargo carecen de reconfiguración en tiempo real y adaptabilidad (H.E. Egilmez & Dane, 2012).

Las redes definidas por software (SDN, del inglés Software Defined Network) surgieron como una alternativa para facilitar la innovación de protocolos y permitir un control sencillo al programar las rutas de los datos en la red (Farhady, Lee, & Nakao, 2015), por lo cual en un corto periodo de tiempo se han convertido en el nuevo paradigma y se visualiza como el futuro de las redes. SDN en lugar de hacer cumplir políticas y protocolos que se ejecutan en dispositivos dispersos como se realiza en una red de datos convencional, donde se limita a realizar el simple reenvío de información entre sus dispositivos y el controlador (el cual será definido en la sección 2.2.5) de la red SDN toma las decisiones (Astuto et al., 2014).

OpenFlow (Mckeown et al., 2008), en el protocolo de transporte que se utiliza para mantener la comunicación entre un controlador lógicamente centralizado y el conmutador para el funcionamiento en el esquema de una SDN, además de delinear el comportamiento esperado del conmutador (Göransson & Black, 2014).

Las SDN han tomado tal importancia que la mayoría de las compañías fabricantes de dispositivos de red incluyen soporte para OpenFlow, aún y cuando éste continúa en desarrollo. Sin embargo, la investigación sobre las SDN está todavía en una etapa temprana y hay cuestiones que deben ser investigadas más a fondo. Por ejemplo, el aprovisionamiento de calidad de servicio (QoS) para las redes definidas por software (SDN), es un problema importante que no ha sido estudiado plenamente (Duan, 2014).

1.1 Planteamiento del problema

Uno de los retos de la prestación de servicios en la nube es desplegar un control coordinado entre los recursos de la red y las aplicaciones, con el fin de facilitar una entrega adaptativa del servicio de datos, además de proporcionar una experiencia adecuada del servicio hacia el usuario, basados en el conocimiento recíproco entre los requisitos de las aplicaciones y las capacidades de la red. Las redes definidas por software, al igual que las redes de datos convencionales, no utilizan ni contemplan mecanismos de calidad de servicio por defecto en sus arquitecturas. Para la implementación se requiere mejorar la arquitectura de la red, escalabilidad, robustez y facilidad de modificación/actualización, de manera que la realización en la red de datos responda a las prioridades de la aplicación a través de

diferentes servicios; aunado a esto es importante que se proporcionen garantías de calidad de servicio (QoS), que aseguren un mejor rendimiento de la red desde la perspectiva de la aplicación.

Los servicios y aplicaciones, requieren recursos de la red para operar de manera eficiente, lo cual no siempre se logra por las limitaciones que se tienen en la red, por ejemplo, en la tasa de transmisión, en el retardo extremo a extremo y sobrecarga de rutas, etc.

Los algoritmos de enrutamiento utilizados en los controladores SDN se basan en el algoritmo de Dijkstra (Jammal, Singh, Shami, Asal, & Li, 2014), este algoritmo permite obtener la ruta más corta hacia el destino. Los algoritmos de la ruta más corta implementados no consideran los parámetros de calidad de servicio para generar sus rutas, por lo tanto es necesario un trabajo fuerte en esta área para incorporar estas variables en la toma de decisión de la ruta (Quintana, 2001) (Fortz, B., Rexford, J., & Thorup, 2002).

El objetivo del enrutamiento con calidad de servicio es buscar un camino que cumpla los requisitos de la aplicación (Quintana, 2001). No solo requiere encontrar el camino de costo mínimo con respecto a una determinada métrica. Los enlaces disponen de una serie de propiedades o métricas además de la tasa de transmisión, dicho camino debe de cumplir simultáneamente las restricciones impuestas por los parámetros de QoS. El enrutamiento con calidad de servicio presenta algunas desventajas, la mayor de las cuales es la exigencia de conocer el estado completo de la red para poder determinar un camino válido (Quintana, 2001).

El monitoreo de parámetros de operación en la red generalmente se realiza mediante aplicaciones en la capa superior (capa de aplicación), de manera que los parámetros obtenidos de este monitoreo pasan por toda la pila de protocolos; este procesamiento causa la generación de un retardo, el cual no permite que el monitoreo se mantenga actualizado. Por lo que es necesario un monitoreo de bajo nivel soportado por los equipos de red mediante OpenFlow, con el cual se logre reducir este retardo generado por el procesamiento de las capas (Kritzinger, 1986).

Aunque no ha sido estandarizado, el protocolo de transporte OpenFlow se utiliza como estándar para las SDN en la interface que comunica los dispositivos de la red con el controlador, por un lado proporciona un soporte muy limitado para conocer las condiciones de la red y de calidad de servicio (ONF, 2012) (Kreutz et al., 2014), por lo cual se utiliza un monitoreo de la red con un agente externo, por otro lado la información que recolecta no es utilizada completamente por el controlador SDN (Azizi, Benaini, & Mamoun, 2015).

La limitante de las redes definidas por software (SDN) y la especificación del protocolo OpenFlow es que proporciona un apoyo muy limitado para proporcionar calidad de servicio (QoS), por lo que es necesaria una adaptación a la arquitectura SDN que incluya la calidad de servicio con mayor importancia. Las SDN requiere un amplio trabajo para una implementación de calidad de servicio, que involucra diferentes enfoques y herramientas para proporcionarlo, cada enfoque es una área de oportunidad que tiene SDN para trabajar, como el monitoreo de parámetros, algoritmos de enrutamiento, entre otros.

Los principales problemas en las SDN, así como las redes tradicionales, es que no se seleccionan las rutas con base a las métricas de QoS para las aplicaciones que lo requieren. Los recursos no son aprovechados se requiere una modificación de la arquitectura que permita al controlador SDN tener el conocimiento del comportamiento de la red y el requerimiento de las aplicaciones.

Además el monitoreo de la red para soportar el enrutamiento con QoS, se realiza desde la capa de aplicación por un agente externo que genera tráfico adicional en la red y un retardo por el procesamiento de las capas. Donde el monitoreo debería ser de bajo nivel y soportado directamente por OpenFlow hacia el controlador.

1.2 Objetivos generales

Diseñar y proponer una solución de calidad de servicio aplicado a redes definidas por Software (SDN) con la utilización del diseño de capas cruzadas (Cross-Layer Design, CLD) para redes de cobertura amplia.

1.3 Objetivos particulares.

1. Investigar parámetros de calidad de servicio y cuáles son sensibles en aplicaciones de alto desempeño.
2. Utilizar el diseño de capas cruzadas (CLD) para la incorporación de calidad de servicio en la arquitectura de red definida por software (SDN).
3. Diseñar y Proponer una solución para las redes definidas por software (SDN) que incorpore calidad de servicio para el aprovisionamiento de recursos orientado a aplicaciones de cómputo de alto desempeño.
4. Desarrollar una metodología de emulación de la solución propuesta.
5. Evaluar la solución propuesta, para el análisis de desempeño.

1.4 Hipótesis

Es posible diseñar un esquema para proporcionar calidad de servicio a las aplicaciones que lo requieran, como la aplicación de alto desempeño (HPC), soportado por las redes definidas por software, proveyendo la ruta y los recursos necesarios para mantener el rendimiento de la red en condiciones favorables.

1.5 Contenido de la tesis

Este documento se encuentra organizado de la siguiente manera: en el capítulo 2 se hace una descripción detallada de los conceptos para las redes definidas por software, calidad de servicio para aplicaciones de alto desempeño y el diseño de capas cruzadas; en el capítulo 3 se describen las diferentes alternativas que existen para SDN en sus diferentes planos y algunas de las soluciones propuestas para incorporar calidad de servicio; en el capítulo 4 se muestra el enrutamiento utilizado en las SDN; en el capítulo 5 se introduce la metodología de la solución propuesta para el presente trabajo de investigación, donde se describen sus componentes y se presenta su diseño; en el capítulo 6 se presentan los resultados obtenidos de la emulación realizada con base en la propuesta; en el capítulo 7 se concluye mencionando el análisis de las ventajas y desventajas, así como la aportación de este trabajo y lo que se considera como trabajo a futuro que se desprende de la presente investigación.

Esta página se ha dejado intencionalmente en blanco.

Capítulo 2. Marco de referencia

La capacidad de las redes de comunicación de datos están siendo rebasadas, llegando a ser insuficientes para poder atender los grandes volúmenes de información que demandan los nuevos servicios, que pueden llegar a transmitir datos en el orden de los peta bytes. Esta cantidad de información se genera debido al gran número de usuarios, sensores y aplicaciones que utilizan los servicios, como BigData (virtualización de servidores, contenidos multimedia, etc.), que requieren grandes tasas de transmisión que pueden ir de los 10 a los 100 Gbps o inclusive una mayor capacidad.

El concepto SDN surgió como la arquitectura de red, donde el plano lógico de control se desacopla del plano de reenvío de información a diferencia de las redes tradicionales, en lugar de hacer cumplir políticas y protocolos que se ejecutan en dispositivos dispersos, la red se limita a realizar el simple reenvío de información y el controlador de la red toma las decisiones (H.E. Egilmez & Dane, 2012), agilizando los flujos de información en la red.

Las SDN pueden ser una solución a problemas como la demanda de recursos, ya que permitirá que las aplicaciones interactúen con la red en forma automatizada y soliciten las políticas o recursos como tasa de transmisión, calidad de servicio, seguridad, entre otras que requieran para su ejecución. En este capítulo se abordan las principales características de SDN, calidad de servicio y el diseño de capas cruzadas.

2.1 Historia de las redes definidas por software

El término de SDN se generalizó en 2009, éste se fue estructurando desde 2003 a partir de la necesidad de algunos investigadores de separar el plano de control del plano de retransmisión en los dispositivos de red con el objetivo de evaluar nuevos protocolos de red de manera rápida y sencilla. Proyectos sobresalientes como ForCES (Yang, L., Dantu, R., Anderson, T., & Gopal, 2004) y 4D (Greenberg et al., 2005) orquestaban esta idea, además de Ethane (Casado et al., 2007) como predecesor directo de OpenFlow (Mckeown et al., 2008), siendo este último el más relevante.

En 2008 se presentó el documento técnico de OpenFlow (Mckeown et al., 2008), basado en un conmutador (Switch) Ethernet, en el cual se propone una forma en que los investigadores ejecuten protocolos experimentales, además de ser promovido para su despliegue con los proveedores de

productos de redes para agregar OpenFlow a sus equipos. En 2009 como se muestra en la línea del tiempo de la Figura 1, se añade el término de redes definidas por software (SDN) pensado para indicar una distinción entre la configuración "de facto" en las redes tradicionales, contra el proceso de programación de la red (DeCusatis, 2013). Además, en la Figura 1, se observa el rápido interés que se obtuvo en poco tiempo. Las SDN se consideran un paradigma de redes emergentes que da esperanza de cambiar las limitaciones de la infraestructura de las redes actuales. Aunque OpenFlow comenzó como un experimento académico, ganó fuerza de manera significativa en la industria en los últimos años. Hoy en día, la mayoría de los vendedores de equipos comerciales incluyen soporte para OpenFlow en sus equipos.

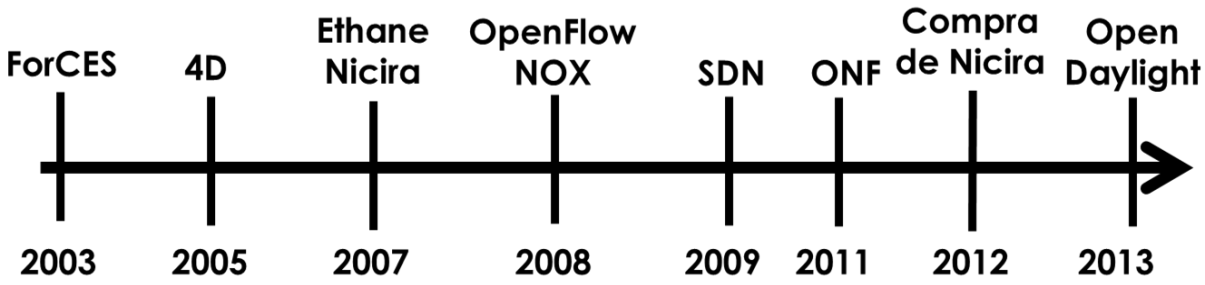

Figura 1. Eventos importantes en la historia de las redes definidas por software.

ForCES (Forwarding and Control Element Separation) (Yang, L., Dantu, R., Anderson, T., & Gopal, 2004), es un trabajo presentado por el grupo de trabajo de la IETF (Internet Engineering Task Force), destaca por ser el primero en proponer la separación del plano de control y el de reenvío. La idea general es la de proveer un dispositivo físico (hardware) simple, como una entidad de solo reenvío y un elemento superior de control basado en programación (software) (Göransson & Black, 2014). 4D (Greenberg et al., 2005) fue el primero en presentar conceptualmente un controlador SDN centralizado, proponiendo una completa reconstrucción de las redes alejada de los dispositivos autónomos, con la idea de concentrar la operación del plano de control en un sistema separado e independiente dedicado solo para ese propósito (Göransson & Black, 2014). Ethane (Casado et al., 2007) es una solución basado en políticas que permiten al administrador de la red, definir políticas a cada nivel de acceso a la red para los usuarios. Proponen 3 principios fundamentales (1) la red debe ser gobernada por políticas de una entidad superior, (2) el ruteo en la red debe conocer estas políticas (3) la red debe reforzar la unión entre los paquetes y su origen (Göransson & Black, 2014).

OpenFlow (Mckeown et al., 2008), basado en Ethane (Casado et al., 2007) en el protocolo de transporte que se utiliza para mantener la comunicación entre un controlador lógicamente centralizado y el

conmutador, además de delinear el comportamiento esperado del conmutador (Göransson & Black, 2014). Cada conmutador mantiene una o más tablas de flujo, que son utilizadas para realizar los paquetes de búsqueda. Los conmutadores Ethane son generalmente iguales a los conmutadores OpenFlow.

El impulso fue lo suficientemente fuerte para hacer que empresas como Google, Facebook, Yahoo, Microsoft, Verizon, y Deutsche Telekom fundaran la *Open Networking Fundation* (ONF)[1], con el objetivo principal de la promoción y adopción de las SDN a través del desarrollo de estándares abiertos (Kreutz et al., 2014). Otra organización con el mismo objetivo es *OpenDayligth*[2], que es promovido por la fundación Linux, que cuenta con el apoyo de empresas como Cisco, IBM, HP, Ericsson y Microsoft, entre otras.

Desde la aparición de las SDN, se han aplicado a una gran variedad de entornos de red, es decir, redes empresariales, redes de centros de datos a gran escala, redes inalámbricas, redes celulares, etc. En poco tiempo han tomado gran importancia no solo en el ámbito académico enfocado a la investigación, sino también en el ámbito comercial, empresarial e industrial. Como resultado del gran interés, está en curso una serie de esfuerzos para la estandarización de las SDN, el cual ya cuenta con el RFC 74260 (E. Haleplidis et al., 2015) por parte de la Fuerza de Tareas de Investigación de Internet (IRTF, por sus siglas en inglés), donde se especifica la terminología, la arquitectura y capas básicas de las SDN.

2.2 Concepto general de las redes definidas por software

Las redes convencionales consisten de dispositivos cerrados, donde cada dispositivo tiene que ser configurado de manera individual, utilizando comandos específicos que tiene cada proveedor de equipos de red. Esto se ha convertido en un obstáculo, al desplegar nuevas versiones de protocolos existentes (por ejemplo, IPv6), actualizaciones o protocolos nuevos en las redes ya instaladas y operando.

El concepto SDN surgió como una arquitectura de red donde a diferencia de las redes convencionales, el plano lógico de control se desacopla del plano de reenvío de información (Figura 2). Las SDN son un nuevo enfoque para la programación de la red, que se refiere a la capacidad de controlar, modificar y manejar el comportamiento de la red dinámicamente mediante software, a través de interfaces abiertas (Hakiri, Gokhale, Berthou, Schmidt, & Gayraud, 2014).

[1] https://www.opennetworking.org/
[2] https://www.opendaylight.org/

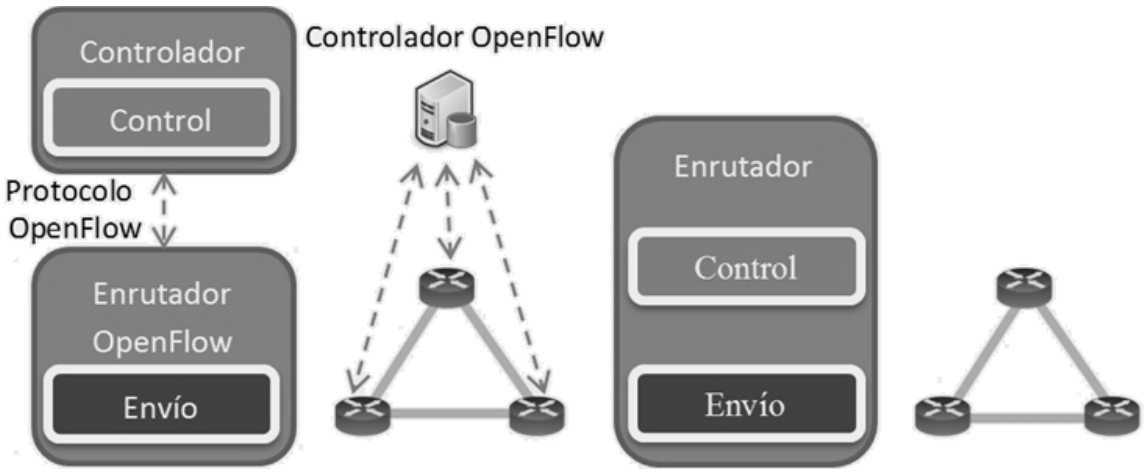

Figura 2. Desacoplamiento del plano de control.

En lugar de hacer cumplir políticas y protocolos que se ejecutan en dispositivos dispersos, la red se limita a realizar el simple reenvío de información y el controlador de la red toma las decisiones sobre el reenvió de los paquetes (H.E. Egilmez & Dane, 2012) agilizando el flujo de información para grandes cantidades de datos y la resolución de problemas de la red. En las redes tradicionales, el plano de reenvío, control, administración y servicios está ampliamente integrado en cada dispositivo de red. Por el contrario, el modelo SDN implementa la mayoría de las decisiones de administración, servicios y selección de ruta como funciones avanzadas dentro del controlador o en aplicaciones (World Wide Technology, 2014). Las diferencias entre estos dos modelos se ilustran en la Figura 3.

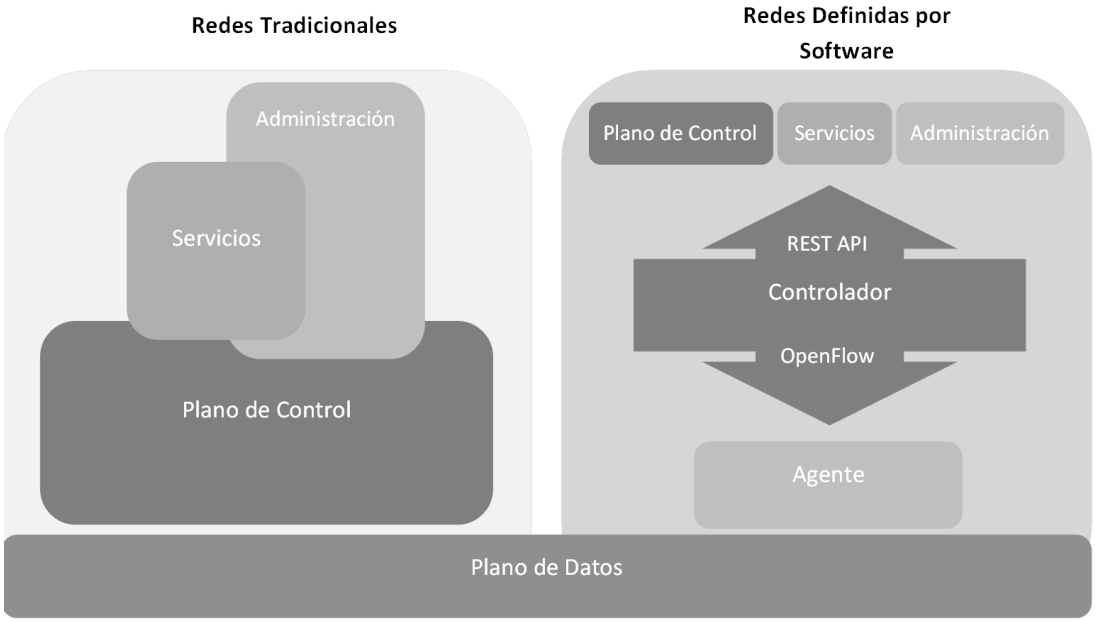

Figura 3. Comparación de la estructura de las redes tradicionales y las redes definidas por software (World Wide Technology, 2014).

Los conmutadores programables siguen las políticas del controlador de la SDN y se envían paquetes en secuencia con el fin de determinar cuál es el camino que los paquetes llevarán a través de la red (Akyildiz, Lee, Wang, Luo, & Chou, 2014).

OpenFlow ofrece una "Interfaz de Programación para Aplicaciones" (API, por sus siglas en ingles), que se encuentra definida y es de código abierto. Proporciona control directo de bajo nivel por tablas de envío y un plano de control que está desacoplado del plano de datos (DeCusatis, 2013).

ForCES (Yang, L., Dantu, R., Anderson, T., & Gopal, 2004), POF (Song, 2013), 4D (Greenberg et al., 2005), OpenFlow (Mckeown et al., 2008) y Ethane (Casado et al., 2007) representan diferentes enfoques para el diseño y despliegue de dispositivos programables del plano de datos. Estas propuestas se basan esencialmente en una modificación de los dispositivos de red para configurar las tablas de reenvió de forma dinámica y remota a través de operaciones simples, tales como añadir, eliminar o actualizar reglas de flujos (Kreutz et al., 2014) (Göransson & Black, 2014).

Además promete simplificar drásticamente la gestión de la red, reducir los costos operativos, y promover la innovación y la evolución de las redes actuales y futuras. Tales características únicas de SDN proporcionan gran incentivo para que las nuevas técnicas de ingeniería de tráfico que explotan la vista global de la red, estados y patrones de flujos, dispongan de las características globales para un mejor control y gestión del tráfico (Akyildiz et al., 2014).

La arquitectura de una SDN ofrece (Akyildiz et al., 2014):

- Visión centralizada de la operación de la red y los requerimientos de calidad de servicio de las diferentes aplicaciones que utilizan la red, incluyendo la información global de la red y la información de la aplicación global (por ejemplo, los requisitos de calidad de servicio).

- Capacidad de programación de los dispositivos de red de forma centralizada, sin tener que manejar los elementos de infraestructura individuales, y un rendimiento mejorado para calidad de servicio.

- Arquitectura Abierta, donde los elementos del plano de datos (es decir los conmutadores), independientemente de los vendedores tienen una interfaz unificada, abierta al controlador para la programación del plano de datos y la recolección del estado de la red.

- Gestión de flujos más flexible y eficiente. Una analogía para SDN podría ser la red de trenes, donde los vagones (flujos) pueden conmutar entre diferentes vías como simples interruptores controlados por una entidad central que monitorea todos los vagones y vías, pudiendo tomar acciones y decisiones rápidamente.

2.2.1 Abstracciones en la arquitectura SDN.

Una SDN tiene un enfoque directo hacia la programación y hacia los programadores, por lo cual se utilizan conceptos, y terminología común en el área de la programación, como la abstracción de información. Las abstracciones, son herramientas esenciales en la investigación en ciencias de la computación y la tecnología de la información, siendo ya una característica ubicua de muchas arquitecturas y sistemas computacionales. El protocolo OpenFlow es una realización de dicha abstracción, que puede ser visto como el equivalente a un "controlador de dispositivo" en un sistema operativo (Kreutz et al., 2014) que tiene el control y conocimiento de la arquitectura de la SDN en diferentes momentos del tiempo.

Una SDN está definida por tres abstracciones fundamentales: El reenvío, la distribución y las especificaciones/configuraciones (Kreutz et al., 2014):

- La abstracción de reenvío, permite que cualquier comportamiento de reenvío deseado por el sistema operativo de la red (controlador), al mismo tiempo oculta detalles del dispositivo físico de reenvío (hardware).

- La abstracción de distribución, provee la vista global de la red, además de proteger las aplicaciones SDN. Su realización requiere una capa de distribución común, que en SDN reside en el sistema operativo de la red (controlador).

- Las especificaciones/configuraciones, es lo que debe permitir a una aplicación de red, definir los requerimientos de desempeño de la red, sin ser responsable de implementar el comportamiento en sí. Esto se logra a través de soluciones de virtualización.

2.2.2 Arquitectura SDN.

Una SDN se define como una arquitectura de red con cuatro pilares (Kreutz et al., 2014):

El plano de control y de datos están desacoplados. La funcionalidad de control se elimina de los dispositivos físicos de red que se convertirán en simples elementos de reenvío de paquetes.

- Las decisiones del reenvío de paquetes se toman en función a flujos, en lugar de basarse en las direcciones destino de los paquetes. Un flujo se define ampliamente por un conjunto de valores en los campos de paquetes que actúan como un criterio apareado (un filtro) y un conjunto de acciones (las instrucciones).
- La lógica de control se mueve a una entidad externa, llamado el controlador de la SDN o sistema operativo de red. Su propósito es similar a la de un sistema operativo tradicional.
- La red es programable a través de aplicaciones en ejecución en la parte superior del controlador que interactúa con los dispositivos en el plano de datos subyacente. Esta es una característica fundamental de la SDN, considerado como su principal propuesta de valor.
- La arquitectura de una SDN contempla tres capas funcionales, el plano de datos, el plano de control y el plano de aplicación. SDN defiende la separación de los planos de control y de datos, los que los equipos de conmutación se controlan mediante un sistema que funciona en un plano de control de forma externa y automatizada. Como se ilustra en la Figura 4.

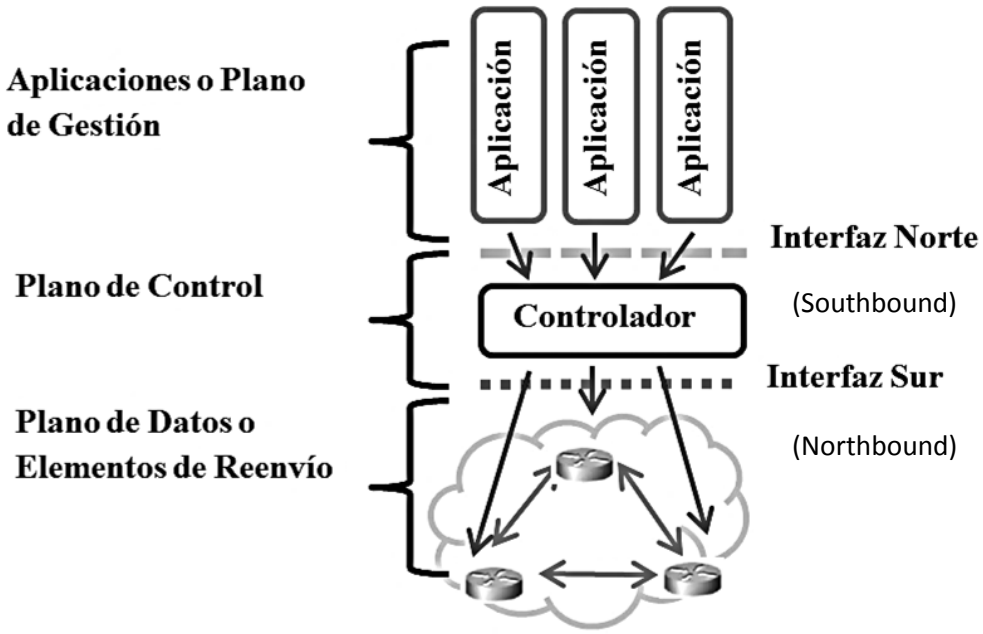

Figura 4. Arquitectura de las redes definidas por software (DeCusatis, 2013).

2.2.3 Interfaces.

Toda la lógica de control de la red se implementa como una aplicación en la parte superior del sistema operativo de red que interactúa con las API de red que ofrece el controlador. Las aplicaciones toman decisiones de control, que manipulan un mapa de la red que se les presenta. El controlador es responsable de mantener la consistencia de este mapa, además, traduce las manipulaciones del mapa dentro del plano de datos, a través de la modificación de las tablas de reenvío por medio de la API, lo que resulta en un plano de datos donde los flujos (no paquetes) son la unidad fundamental del control (Shenker & Casado, 2011).

El controlador cuenta con dos interfaces principales, una que interactúa con el plano de gestión llamada *Northbound* o hacia el norte (NBI, por sus siglas en inglés), y otra que interactúa con el plano de datos, llamada *Southbound* o hacia el sur (SBI, por sus siglas en inglés) (Figura 4). Así mismo, y con la finalidad de establecer comunicación con otros controladores en la red, se cuenta con las interfaces *Eastbound* o hacia el este (EBI, por sus siglas en inglés*)* y *Westbound* o hacia el oeste (WBI, por sus siglas en inglés) (DeCusatis, 2013).

Interfaz Southbound: Este conjunto de instrucciones son las API expuestas a capas inferiores que permiten la externalización del plano de control de la SDN al plano de datos (Hakiri et al., 2014). Formaliza la manera en que los elementos de control y plano de datos interactúan (Kreutz et al., 2014).

Interfaz Northbound: Permite el intercambio de datos entre el controlador y la aplicación. El tipo de información que se intercambia, su forma y su frecuencia depende de cada aplicación. No hay estandarización para esta interfaz (Hakiri et al., 2014). Esta API abstrae los conjuntos de instrucciones de bajo nivel utilizados por las SBI para programar dispositivos de reenvío (Kreutz et al., 2014).

Interfaces Eastbound y Westbound: Permite la interconexión de redes convencionales con las SDN, además sirven como conducto de información entre varios planos de control de la SDN de diferentes dominios de SDN. Ayudan a lograr una visión de red global e influyen en las decisiones de enrutamiento de cada controlador (Hakiri et al., 2014).

El objetivo de estas interfaces es proporcionar dos cosas (DeCusatis, 2013):

- Una apropiada abstracción de estados y eventos relacionados con la red subyacente.

- Un canal de comunicaciones con características apropiadas, para apoyar la necesaria funcionalidad requerida entre los clientes en cualquier extremo del protocolo.

2.2.4 Plano de Administración o Aplicaciones.

Las aplicaciones definen el comportamiento de la red. Las aplicaciones son proveídas con una vista de la topología completa de la red, generalmente incluyendo alguna representación de los equipos conectados y sus puntos de conexión a la red. Las aplicaciones pueden consultar información acerca de las propiedades, tales como las cargas de tráfico (DeCusatis, 2013).

La API de red que proporciona el plano de control expone la topología y la conectividad de la red, así como un conjunto de servicios que permiten respuestas programáticas a eventos en la red. Las aplicaciones son los consumidores de los servicios prestados por la plataforma del controlador (DeCusatis, 2013).

El plano de administración también se conoce como plano de gestión, dado que es el conjunto de aplicaciones que aprovechan las funciones ofrecidas por la interfaz norte para implementar el control de la red y la lógica de funcionamiento. Esto incluye aplicaciones tales como enrutamiento, cortafuegos, balanceadores de carga, seguimientos y entre otros. En esencia, una aplicación de gestión define las políticas, que se traducen en última instancia en instrucciones específicas para la SBI que programan el comportamiento de los dispositivos de reenvío como se muestra en la Figura 5 (Kreutz et al., 2014).

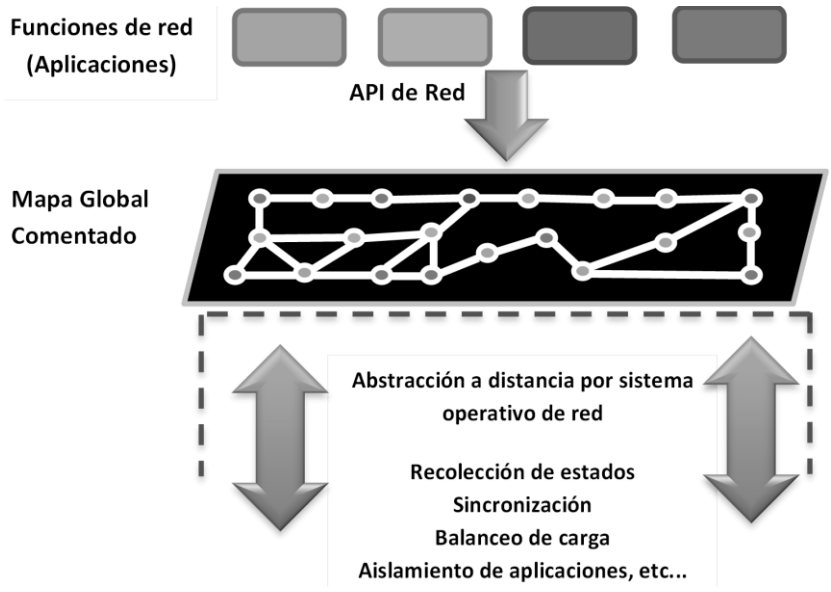

Figura 5. Abstracción de red presentada a las aplicaciones (Kreutz et al., 2014).

2.2.5 Plano de Control.

El plano de control proporciona una capa agregada de control en el envío del estado de la red. Aunque el modelo del plano de control es centralizado, la implementación puede ser distribuida para la escalabilidad y redundancia. El plano de control presenta una abstracción de la red hacia las aplicaciones. La aplicación no tiene que preocuparse acerca de cómo se está creando el mapa y la forma en que se mantiene sincronizada y actualizada. Todo lo que sabe es que tiene el mapa como entrada, lleva a cabo su función de control y entrega una decisión (DeCusatis, 2013).

El controlador es un elemento crítico en una arquitectura SDN, ya que es la pieza clave para el apoyo a la lógica de control para generar la configuración de la red con base en las políticas definidas por el operador de red. Al igual que en un sistema operativo tradicional, la plataforma de control abstrae los detalles de bajo nivel de la conexión y la interacción con dispositivos de reenvío, es decir materializa las políticas de red como se muestra en la Figura 6 (Kreutz et al., 2014).

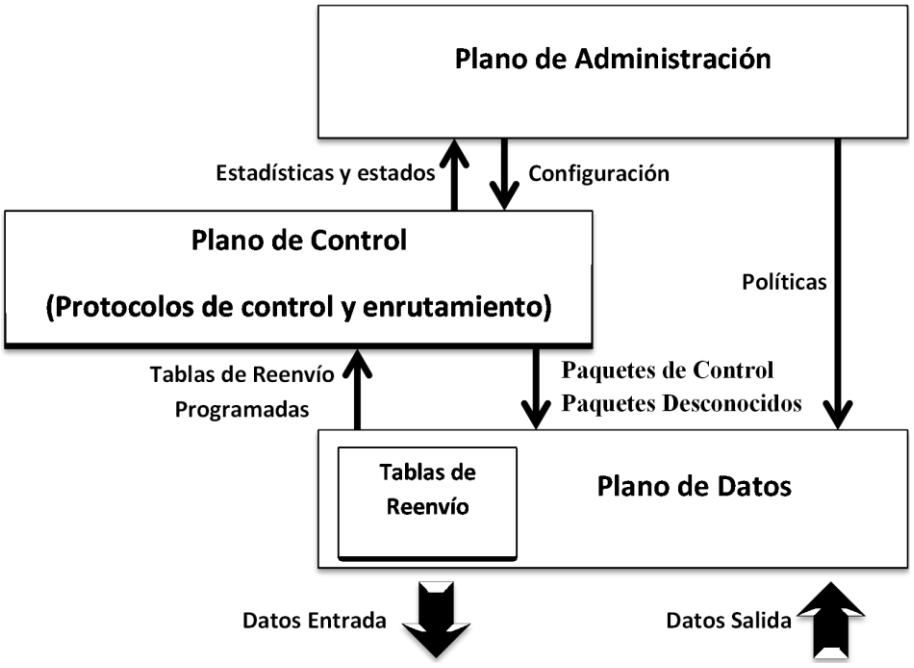

Figura 6. Roles de los planos de control, datos y administración (Göransson & Black, 2014).

2.2.6 Plano de Datos.

El plano de datos proporciona el servicio fundamental de reenvío de datos, está optimizado para mover datos entre los puertos y se compone de uno o más elementos de reenvío interconectados por enlaces. Cada conmutador presenta un modelo de reenvío local para el plano de control, este modelo de reenvío

proporciona una cierta capacidad de programación que se puede apreciar en la Figura 7 (DeCusatis, 2013).

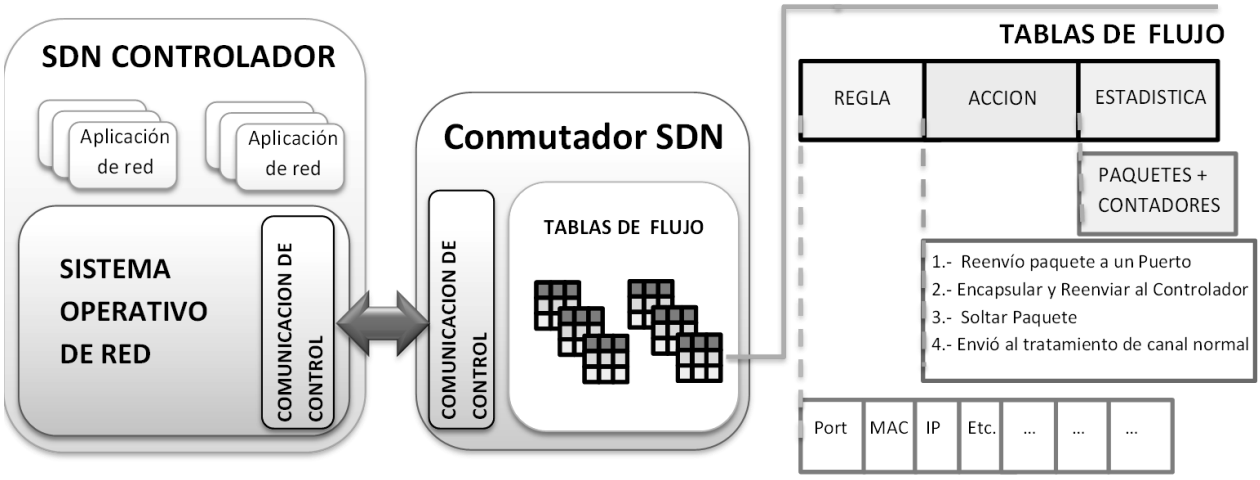

Figura 7. Comunicación entre un controlador y un conmutador SDN (DeCusatis, 2013).

La abstracción de reenvío común elimina la necesidad de múltiples planos de control distribuido independientes, dando a un controlador externo la capacidad de definir flujos de paquetes independientemente de la capa tradicional de red. El mayor beneficio es que en lugar de que las redes exhiban un comportamiento fijo definido por los protocolos, su comportamiento puede ser ahora programable, definida por software (DeCusatis, 2013).

Los dispositivos de reenvío son los equipos o dispositivos físicos (hardware) en el plano de datos basados en programación (software) que realizan una serie de operaciones elementales. Los dispositivos de reenvío han definido conjuntos de instrucciones (por ejemplo, reglas de flujo) que se utilizan para tomar acciones sobre los paquetes de entrada (por ejemplo, hacia adelante a un puerto específico). Estas instrucciones son definidas por la API Southbound y se instalan en los dispositivos de reenvío por el controlador SDN (Kreutz et al., 2014).

Se utiliza el término conmutador SDN (SDN switch) para referirse al elemento de reenvío en el plano de datos. El término conmutador SDN no debe considerarse como una limitación de las funciones del elemento de red (DeCusatis, 2013), si el dispositivo es un enrutador (Router), conmutador (Switch), o cortafuegos (Firewall).

En el conmutador SDN una o más instancias gestionan las conexiones con el plano de control. La gestión de puertos y el motor de reenvío, proporcionan las funciones reales de manejo de los paquetes. Puede haber interfaces de configuración o de vigilancia adicionales soportados por el conmutador SDN (DeCusatis, 2013). De la Figura 8, además de un mecanismo de reenvío para el plano de datos, un conmutador SDN debe tener un operador que comunica al controlador SDN y expone la programabilidad del motor de reenvío.

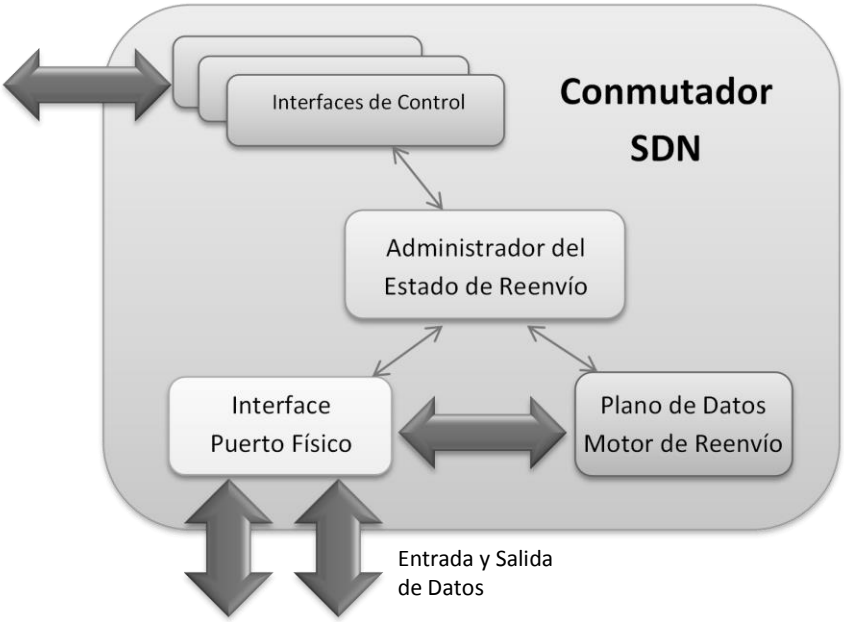

Figura 8. Arquitectura del conmutador SDN (DeCusatis, 2013).

2.3 Calidad de servicio

La Recomendación ITU-T E.800 (International Telecommunication Union; coordinación de Telecomunicaciones), facilita una serie de términos más comúnmente utilizados en el estudio y la gestión de la calidad de servicio (QoS). Esos términos y definiciones se aplican a todos los servicios de telecomunicación y a todas las topologías de red utilizadas para prestar dichos servicios (UIT-T, 2008).

Desde el punto de vista del proveedor, la calidad de funcionamiento de la red es un concepto con respecto al cual se definen, miden y controlan las características de la red para lograr un nivel satisfactorio de QoS. Los intereses y los puntos de vista de usuarios y proveedores difieren, y normalmente hay que llegar a un compromiso entre calidad y economía (UIT-T, 2008).

En la utilización de un servicio, se pueden identificar dos secciones (UIT-T, 2008):

- La organización, es decir la Administración de telecomunicaciones, la empresa de explotación, etc., que proporciona los medios y facilidades, para acceder y utilizar los al servicios.
- La red, es decir los medios necesarios (terminales, líneas, equipos de conmutación, etc.) realmente utilizados.

Para especificar la calidad de servicio (QoS) de extremo a extremo de la Figura 9, es necesario determinar las condiciones operativas específicas en que un servicio se proporciona a través de la conexión. La calidad de servicio también puede verse alterada, dadas unas condiciones operativas específicas, por condiciones del entorno, como el tráfico y el enrutamiento (UIT-T, 2008).

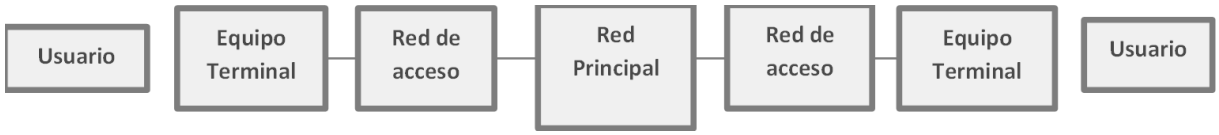

Figura 9. Factores que influyen en la calidad de servicio extremo a extremo (UIT-T, 2008).

2.3.1 Definición.

La calidad de servicio son las características de una entidad que determinan su capacidad para satisfacer las necesidades explicitas e implícitas. Las características deben ser observables y/o mensurables. Cuando las características se definen, se convierten en parámetros y se expresan en unidades de medida (UIT-T, 2008).

Por definición de la ITU, Calidad de servicio de telecomunicaciones es:

“La totalidad de las características de un servicio de telecomunicaciones que determinan su capacidad para satisfacer las necesidades explícitas e implícitas del usuario del servicio “ (UIT-T, 2008).

La calidad de servicio se caracteriza por el efecto combinado de la logística, la facilidad de utilización, las aptitudes, la calidad de funcionamiento en la seguridad y otros factores específicos de cada servicio (UIT-T, 2008).

2.3.2 Calidad de servicio en redes

Los administradores de red utilizan la calidad de servicio (QoS) para garantizar el rendimiento a aplicaciones de misión crítica o las que requieran un trato preferencial, de manera que sus transacciones

se puedan procesar con la garantía de las condiciones requeridas. Se utiliza QoS para gestionar la prioridad y recursos de la red, de modo que tengan los recursos de red necesarios. No todas las aplicaciones requieren las mismas condiciones, por ejemplo, los cajeros electrónicos requieren más interactividad mientras que las aplicaciones de streaming pueden soportar retardos mayores.

La calidad de servicio para las redes incluye un conjunto de normas y mecanismos para asegurar un rendimiento alto. Mediante el uso de mecanismos de QoS, los administradores de red pueden utilizar los recursos existentes de manera eficiente y garantizar un nivel de servicio requerido, sin expansión reactiva o el exceso de aprovisionamiento de sus redes.

Tradicionalmente, el tráfico de la red es tratado por igual. El resultado fue que todo el tráfico de red recibe el mejor esfuerzo de la red, sin garantías de fiabilidad, retardo, variación de retardo, u otras características de rendimiento (Hilmi Enes Egilmez, 2012). El concepto de calidad de servicio es aquel en la que las exigencias o requerimientos de algunas aplicaciones o los usuarios son más críticos que otros, lo que significa que una parte del tráfico necesita tratamiento preferencial.

El objetivo de la calidad de servicio es proporcionar el servicio de entrega preferencial para las aplicaciones que lo requieran, garantizando suficientes recursos, el control de la latencia y fluctuaciones, además de la reducción en la pérdida de datos.

2.3.3 Parámetros.

Para medir la calidad de servicio en un entorno de red, se toman en cuenta los siguientes parámetros tradicionales como métricas formales para su medición.

1. **Disponibilidad del servicio (Availability):** La fiabilidad de la conexión de los usuarios. Número de intentos exitosos entre el número total de intentos. Medido en porcentaje.
2. **Retardo (Delay):** El tiempo que tarda un paquete para viajar a través de la red desde un extremo a otro. Medido en tiempo.
3. **Fluctuación del retardo (Jitter):** La variabilidad del retardo en ejecución entre los paquetes a través de la red desde un extremo a otro. Medido en tiempo.
4. **Caudal eficaz (Throughput):** La tasa de transmisión real de transporte de datos a través de la red (sin encabezados, sin tiempos de guarda etc.) y siempre es inferior a la tasa de transmisión. Medido en bits por segundo.

5. **Tasa de trasmisión (Bit rate):** La velocidad máxima de la transmisión total este término también se utiliza para nombrar a los recursos de velocidad máxima del canal o el servicio. Medido en bits por segundo.
6. **Tasa de pérdida paquetes (Packet loss):** La cantidad de paquetes que son descartados, perdidos o dañados mientras atraviesan la red y no llegan al otro extremo. Este parámetro es medido en porcentaje.
7. **RTT (Round- trip delay time):** Se define como el tiempo total transcurrido entre el envío de un paquete y la recepción de la respuesta (ACK) por parte del receptor. Medido en tiempo.
8. **PSNR (Peak signal-to-noise ratio):** Parámetro utilizado como medida de calidad en la reconstrucción de la compresión de imágenes en la transmisión de video sobre redes. Es un concepto similar a SNR (Relación señal a ruido), es una relación entre la señal útil y la señal de ruido introducida, la diferencia que tiene PSNR es que compara la diferencia el nivel constante de gris en las imágenes con el nivel de distorsión en la imagen resultante. Es medido en escala logarítmica.
9. **Latencia (Latency):** Se define como el tiempo que tarda un paquete en ser procesado dentro de un dispositivo, es decir, el tiempo entre el arribo de un paquete, su procesamiento dentro del dispositivo y una vez que sale del dispositivo. Medido en tiempo.

Los diseños de redes, por lo general tratan de intentar maximizar la disponibilidad del servicio y la tasa de transmisión efectiva, reducir el retraso, y tratar de eliminar las fluctuaciones y la pérdida de paquetes (UIT-T, 2008).

2.4 Requerimientos de aplicaciones de alto desempeño.

La calidad de servicio se utiliza en cómputo de alto desempeño (HPC, High Performance Computing) con el fin de reducir el impacto entre aplicaciones para los códigos científicos y el uso de la red de comunicaciones. Los administradores de sistemas de HPC pueden aplicar políticas de calidad de servicio específicos en función de cada puesto de trabajo por medio de herramientas de gestión de los recursos del sistema. Básicamente las aplicaciones de HPC se clasifican en dos tipos de tráficos generales, aquellas que son sensibles al retardo y aquellas sensibles a la tasa de transmisión, asignando alta prioridad a las aplicaciones sensibles al retardo y baja prioridad a las aplicaciones sensibles a la tasa de transmisión (Jokanovic, A., Sancho, J. C., Labarta, J., Rodriguez, G., & Minkenberg, 2012).

En el caso de cómputo en la nube, generalmente todo el tráfico se trata de la misma manera utilizando el mejor esfuerzo, por lo cual no se da ninguna garantía que la información llegue a su destino en el menor

tiempo posible, ni que cumpla con los requerimientos de los parámetros de calidad que la aplicación pueda requerir.

Las aplicaciones sensibles al retardo requieren una interacción en tiempo real y los paquetes deben llegar en el menor tiempo posible, esto quiere decir que este parámetro tiene que mantenerse bajo y la fluctuación debe mantenerse estable. Al alejarse de estos requerimientos, la pérdida de paquetes se podría incrementar de manera importante.

Las aplicaciones sensibles a la tasa de transmisión generan grandes cantidades de información que necesitan ser transferidas entre servidores, generalmente este parámetro tiende a ser alto, soporta retardos altos, sin embargo se tienen que mantener estables, además de asegurar que la información llegue completa, y una pérdida de paquetes baja o nula para evitar retransmisiones.

Para poder proporcionar calidad de servicio a una aplicación de HPC, los parámetros de los cuales son (Jokanovic, A., Sancho, J. C., Labarta, J., Rodriguez, G., & Minkenberg, 2012): **(1) Retraso (2) Fluctuación (3) Tasa de trasmisión (4) Tasa de pérdida paquetes.**

Estos cuatro parámetros son básicos para que se elija una ruta específica hacia el destino (Figura 10) con base a los requerimientos de la aplicación entre las posibles rutas existentes en la nube, para mantener el rendimiento deseado de la red y las aplicaciones.

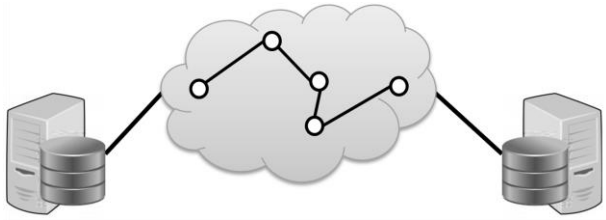

Figura 10. Representación de una comunicación entre centros de datos a través de la nube computacional por una ruta específica con base a los requerimientos de la aplicación.

2.5 Definición de parámetros de QoS en la red

Uno de los objetivos de la presente investigación es la definición de un esquema de calidad de servicio para aplicaciones de cómputo de alto desempeño, por lo tanto, los cuatro parámetros de red que se utilizarán como métricas para medir el rendimiento de este tipo de aplicaciones son: retardo, fluctuación del retardo, tasa de trasmisión y tasa de pérdida paquetes. A continuación se describe cada uno de ellos.

2.5.1 Tasa de transmisión

Para definir la máxima tasa de trasmisión consumida por un flujo de información ($\boldsymbol{X_e^{m,k}}$), se considera que el flujo transita el enlace "***e***" hacia el destino "***k***", donde el flujo de información tiene que ser menor o igual a ($\boldsymbol{\mathcal{U}_e}$). La capacidad del enlace "***e***" se representa por la ecuación (1) y se ilustra en la Figura 11 (Raayatpanah, Fathabadi, Khalaj, Khodayifar, & Pardalos, 2014).

$$max\ (X_e^{m,k}) \leq \mathcal{U}_e \qquad (1)$$

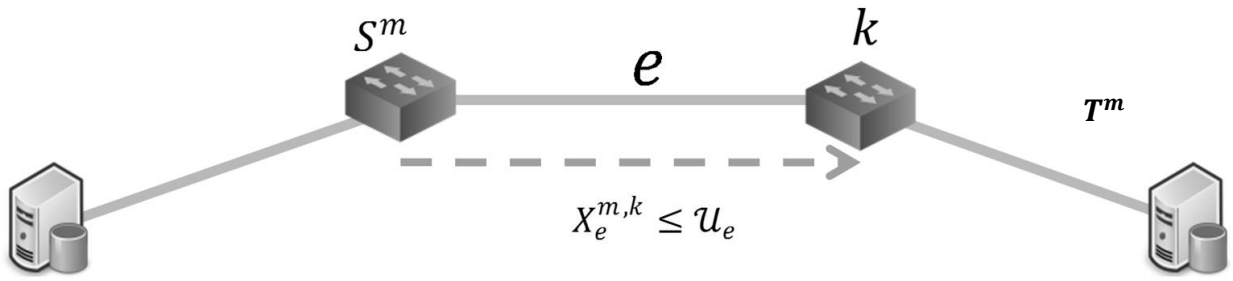

Figura 11. Máxima tasa de transmisión para un enlace extremo a extremo.

Dónde:

$e = enlace.$

$\mathcal{U}_e = capacidad\ del\ enlace.$

$S^m = nodo\ fuente.$

$T^m = destino\ en\ sesión$ m.

$k = nodo\ destino \quad k \in T^m$.

$m = sesión\ entre\ nodo\ fuente - destino.$

$M = Multiples\ sesiones.$

$X_e^{m,k} = flujo\ de\ información\ sobre\ el\ enlace$ "e" $hacia\ el\ destino$ "k" $que\ pertenece\ a\ la\ sesion$ "m".

En caso de múltiples sesiones en el mismo enlace, la suma de las velocidades de transmisión consumidas en el enlace "***e***" por las "***m***" sesiones hacia cada destino "***k***", tiene que ser menor o igual a la capacidad total del enlace "***e***" (ver ecuación (2)) y se ilustra en la Figura 12 (Raayatpanah et al., 2014).

$$\sum_{m \in M} {}^{k \in T^m} (X_e^{m,k}) \leq \mathcal{U}_e \qquad (2)$$

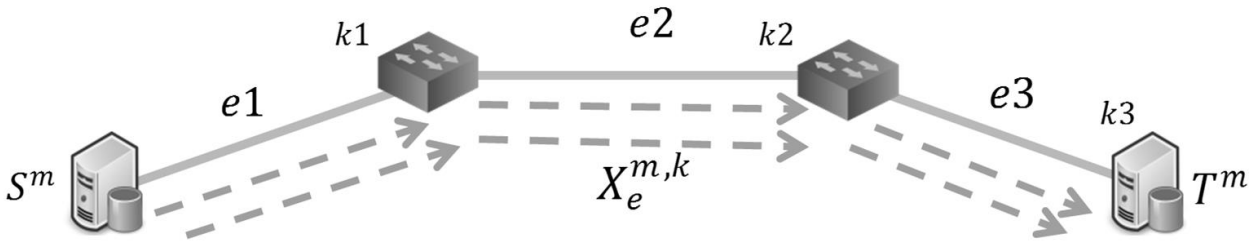

Figura 12. Máxima tasa de trasmisión para múltiples sesiones.

En la comunicación extremo a extremo, donde una trayectoria ya se ha definido, la velocidad máxima de transmisión ($\boldsymbol{X_e^{m,k}}$), desde el nodo fuente ($\boldsymbol{S^m}$) hasta el nodo destino ($\boldsymbol{T^m}$) a través de los $\boldsymbol{k}$ nodos, estará dada por la capacidad máxima disponible de los enlace en la trayectoria con el menor valor y se representa en la ecuación (3).

$$max\ (X_e^{m,k})\ \leq min(\mathcal{U}_e) \tag{3}$$

Por ejemplo si los enlaces de la Figura 12 tuvieran las siguientes capacidades disponibles, $\mathcal{U}_{e1} = 10Mbps$, $\mathcal{U}_{e2} = 5Mbps\ y\ \mathcal{U}_{e3} = 8Mbps$, por lo tanto la máxima tasa de transmisión es el mínimo valor entre las capacidades disponibles de los enlaces, por lo tanto la máxima velocidad que alcanzará la transmisión de este ejemplo será 5Mbps al utilizar la ecuación (3).

2.5.2 Retardo

Si se considera una trayectoria establecida para la comunicación extremo a extremo, el retardo total ($\boldsymbol{D^{m,k}(p)}$) de una trayectoria "$\boldsymbol{p}$" hacia un destino "$\boldsymbol{k}$", se define como la suma de los retardos ($\boldsymbol{d_e}$) de cada enlace, más la suma de las latencias ($\boldsymbol{l_k}$) de cada uno de los dispositivos pertenecientes a la trayectoria "$\boldsymbol{p}$", (ver ecuación (4)) se ilustra en la Figura 13. De manera general se puede representar como una suma de sumatorias entre los retardos y las latencias como en la ecuación (5) (Raayatpanah et al., 2014).

$$D^{m,k}(p) = d_{e1} + d_{e2} + d_{e3} + l_{k1} + l_{k2} \tag{4}$$

$$D^{m,k}(p) = \sum_{e \in p} d_e + \sum_{k \in p} l_k \tag{5}$$

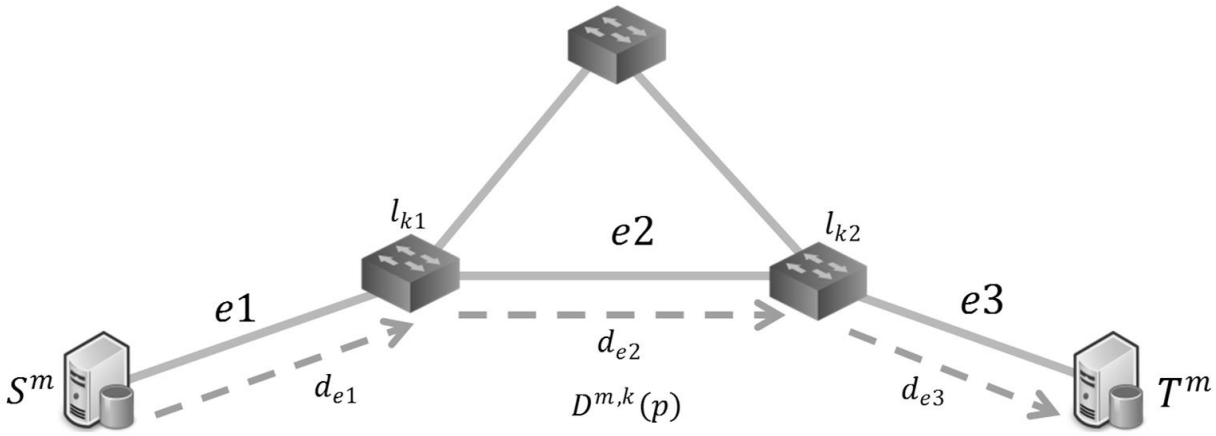

Figura 13. Retardo en una enlace extremo a extremo.

Dónde:

$d_e = retardo\ de\ los\ paquetes\ del\ enlace$ "e".

$P^{m,k} = conjunto\ de\ todos\ los\ caminos\ desde\ la\ fuente\ S^m\ hacia\ el\ nodo\ T^m\ en\ la\ sesion$ "m".

$p = camino\ o\ trayectoria\ extremo\ a\ extremo,\ p \in P^{m,K}$.

$D_{max}^{m,k} = máxima\ tolerancia\ de\ retardo.$

$desde\ la\ fuente\ S^m\ hacia\ el\ nodo$ "k" $en\ la\ sesion$ "m".

Además, la probabilidad de que el retardo en la trayectoria, sea igual o menor a la máxima tolerancia de retardo, es igual a 1 menos la probabilidad de violación de la restricción de retardo, desde el nodo fuente hacia el nodo destino, como se representa en la ecuación (6) (Raayatpanah et al., 2014).

$$\Pr\left(D^{m,k}(p) \leq D_{max}^{m,k}\right) = 1 - \beta^{m,k} \tag{6}$$

Dónde:

$\beta^{m,k} = probabilidad\ de\ violación\ de\ las\ restricciones\ de\ retardo.$

Dado que la variable aleatoria $\boldsymbol{D^{m,k}(p)}$ solo toma valores no negativos con una media finita, el siguiente acotamiento superior sobre la probabilidad de violación de retardo extremo a extremo puede ser obtenido usando las desigualdades de Markov's en la ecuación (6), como resultado se obtiene la ecuación (7) (Raayatpanah et al., 2014).

$$\Pr\left(D^{m,k}(p) \geq D_{max}^{m,k}\right) \leq \frac{E(D^{m,k}(p))}{D_{max}^{m,k}} \tag{7}$$

Dónde:

$E(D^{m,k}(p)) = es\ el\ promedio\ o\ valor\ esperado\ de\ retardo\ extremo\ a\ extremo$

La siguiente restricción puede ser derivada por cada camino $\boldsymbol{p} \in \boldsymbol{P^{m,K}}$ en orden, para garantizar la probabilidad de violación de retardo extremo a extremo, representado en la ecuación (8) (Raayatpanah et al., 2014).

$$\frac{E(D^{m,k}(p))}{D_{max}^{m,k}} \leq \beta^{m,k} \tag{8}$$

El promedio de retardo extremo a extremo se puede representar, como la suma de los retardos promedio en cada enlace "***e***", más la suma de las latencias promedio en cada nodo del camino "***p***", se representa en la ecuación (9) (Raayatpanah et al., 2014).

$$E(D^{m,k}(p)) = \sum_{e \in p} \overline{d_e} + \sum_{k \in p} \overline{l_k} \tag{9}$$

Si la trayectoria "***p***" es usada para transmitir un flujo hacia el nodo "***k***", entonces la ecuación (10) debería satisfacer la desigualdad (Raayatpanah et al., 2014).

$$\frac{\sum_{e \in p} \overline{d_e} + \sum_{k \in p} \overline{l_k}}{D_{max}^{m,k}} \leq \beta^{m,k} \tag{10}$$

Donde el retardo en la trayectoria "***p***" se define con la siguiente restricción de la ecuación (11):

$$D(p) = \begin{cases} \dfrac{\sum_{e \in p} \overline{d_e} + \sum_{k \in p} \overline{l_k}}{D_{max}^{m,k}}, & F(p) > 0 \\ 0, & otros\ casos \end{cases} \tag{11}$$

Dónde:

$F(p) = flujo\ de\ información\ sobre\ la\ trayectoria$ "p".

2.5.3 Fluctuación del retardo

Tomando como base un tráfico de congestión ligero-moderado y retardos independientes para cada uno de los enlaces, la probabilidad que la trayectoria "***p***" satisfaga la restricción de la aplicación en las

fluctuaciones de retardo, es menor o igual que la varianza del retardo entre la restricción al cuadrado. Se representa en la ecuación (12) (Raayatpanah et al., 2014).

$$\Pr\left(|D^{m,k}(p) - E(D^{m,k}(p))| \geq J^{m,k}\right) \leq \frac{V(D^{m,k}(p))}{(J^{m,k})^2} \tag{12}$$

Dónde:

$J^{m,k} = \ máxima\ tolerancia\ de\ jitter.$

$V(D^{m,k}(p)) = varianza\ del\ retardo\ extremo\ a\ extremo.$

De la ecuación anterior, el retardo extremo a extremo, está delimitado por la ecuación (13) y se puede apreciar en la Figura 14:

$$E(D^{m,k}(p)) + J^{m,k} \tag{13}$$
$$E(D^{m,k}(p)) - J^{m,k}$$

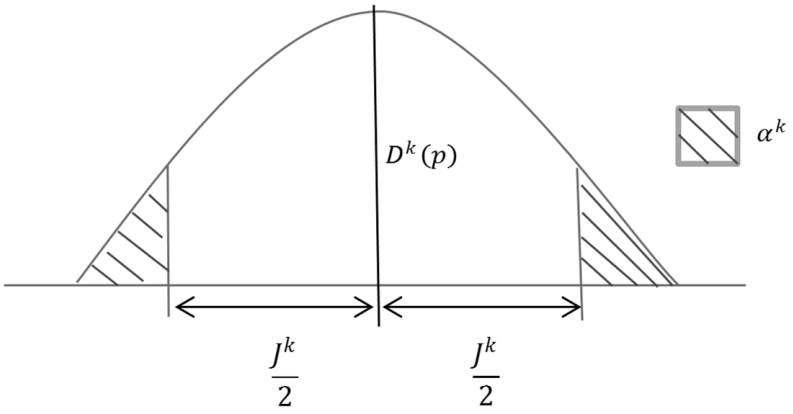

Figura 14. Representación gráfica de variación de las fluctuaciones de retardo permitida.

La distribución de la varianza del retardo extremo a extremo de la trayectoria "***p***", es la suma de las varianzas en cada enlace usado en la trayectoria, como se muestra en la ecuación (14) (Raayatpanah et al., 2014).

$$V(D^{m,k}(p)) = \sum_{e \in p} \sigma_e^2 \tag{14}$$

Dónde:

$\sigma_e^2 = varianza\ de\ la\ variable\ aleatoria\ d_e$

Se puede considerar la siguiente restricción para garantizar la probabilidad de violación de las fluctuaciones para satisfacer la trayectoria en la ecuación (15) (Raayatpanah et al., 2014).

$$\frac{\sum_{e\in p}\sigma_e^2}{(J^{m,k})^2} \leq \alpha^{m,k} \tag{15}$$

Dónde:

$\alpha^{m,k} = probabilidad\ de\ violación\ del\ jitter\ desde\ la\ fuente\ S^m$
$hacia\ el\ nodo$ "k" $en\ la\ sesion$ "m".

Por lo tanto la restricción de las fluctuaciones ***J(p)*** para la trayectoria ***p***, está definido en la ecuación (16) como:

$$J(p) = \begin{cases} \dfrac{\sum_{e\in p}\sigma_e^2}{(J^{m,k})^2}, & F(p) > 0 \\ 0, & otros\ casos \end{cases} \tag{16}$$

Por lo tanto la probabilidad de violación de fluctuación está garantizada con un máximo y un mínimo.

$$J_T \in \left[J^{min}, J^{max}\right] \tag{17}$$

2.5.4 Tasa de pérdida de paquetes

La tasa de pérdida de paquetes es la suma de los paquetes perdidos en **N** intervalos de tiempo, divido entre la entrada total de paquetes medidos en el mismo intervalo de tiempo, (ecuación (18)). Tomando los valores esperados, la probabilidad de pérdida de paquetes en un sistema de búffer finito, se muestra en la ecuación (19) (Raayatpanah et al., 2014).

$$PLR = \lim_{N\to\infty} \frac{\sum_{t=1}^{N} L(t)}{\sum_{t=1}^{N} Q(t)} \tag{18}$$

$$Pls = \frac{\mathbb{E}[L(t)]}{\mathbb{E}[Q(t)]} \tag{19}$$

Dónde:

$PLR = probabilidad\ de\ pérdida\ de\ paquetes\ en\ un\ sistema\ de\ buffer\ finito.$
$L(t) = pérdida\ de\ paquetes\ en\ un\ intervalo\ de\ tiempo$ "t".

$Q(t) = entrada\ total\ de\ paquetes\ medidos\ en\ el\ mismo\ intervalo\ de\ tiempo$ "t".
$q(t) = número\ de\ paquetes\ paquetes\ que\ ocupan\ el\ buffer\ el\ mismo\ intervalo\ de\ tiempo$.
$Pls = probabilidad\ de\ pérdidad\ de\ paquetes.$
$\mathbb{E} = valor\ esperado.$

La pérdida de paquetes $\boldsymbol{L(t)}$ se puede representar como una suma de sumatorias de las funciones de identificación particular de cada caso donde se pierden paquetes como se muestra en la ecuación (20), por ejemplo que el tiempo de vida del paquete sea igual a cero, que el paquete exceda el tiempo máximo de retardo permitido, que el búffer se desborde, entre otros, sin embargo solo se tomará la suma general de todo los casos de pérdida de paquetes en una trayectoria ***p*** (Raayatpanah et al., 2014).

$$L(t) = \sum_{t=1}^{N} \mathbb{I}(q(t) > b) + \sum_{t=1}^{N} \mathbb{I}(x) + \sum_{t=1}^{N} \mathbb{I}(y) + \cdots + \sum_{t=1}^{N} \mathbb{I}(n) \tag{20}$$

Dónde:
$b = capacidad\ del\ buffer.$
$\mathbb{I}(x) = función\ de\ identificación\ particular.$

2.6 Diseño de capas cruzadas

El diseño de arquitecturas de red se basa en el principio de capas que proporciona una herramienta atractiva para el diseño de sistemas interoperables para el despliegue rápido y de implementación eficiente (Zorba, N. , Skianis C. and Verikoukis, 2011).

Una arquitectura en capas, prohíbe la comunicación directa entre capas no adyacentes, esto significa que capa superior sólo haga uso de los servicios en las capas inferiores y viceversa. Alternativamente, las capas pueden diseñarse violando la arquitectura de referencia, por ejemplo, que permitan la comunicación directa entre capas no adyacentes o compartir variables entre capas. Tal violación de una arquitectura es un diseño de capas cruzadas con respecto a la arquitectura de referencia (Srivastava & Motani, 2005).

En este trabajo de tesis se utiliza el concepto de diseño de capas cruzadas (del inglés Cross-Layer Design, CLD) para la formalización en las propuestas de mejora a la arquitectura de la SDN.

2.6.1 Modelo OSI.

El modelo OSI (del inglés, Open System Interconnection) fue desarrollado para apoyar la estandarización de arquitecturas de red utilizando el modelo de capas. Los principales conceptos que motivan la utilización de capas son las siguientes (Zorba, N. , Skianis C. and Verikoukis, 2011):

- Cada capa realiza un subconjunto de las funciones de comunicación requeridas.
- Cada capa se apoya en la siguiente capa inferior para realizar funciones más sencillas.
- Cada capa proporciona servicios a la siguiente capa más alta.
- Los cambios en una capa no deben requerir cambios en otras capas.

Tales conceptos fueron utilizados para definir una pila de protocolos de referencia de siete capas, pasando de la capa física hasta la capa de aplicación, la arquitectura del modelo OSI se muestra en la Figura 15 (Zorba, N. , Skianis C. and Verikoukis, 2011).

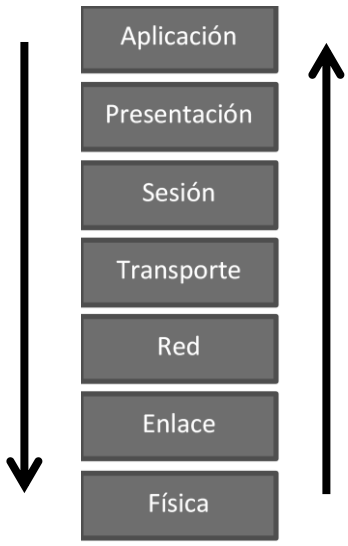

Figura 15. Modelo OSI (Wetteroth, 2001).

Una arquitectura en capas, como el modelo OSI, divide la tarea global de redes y define una jerarquía de los servicios que debe proporcionar cada capa de manera individual. Los servicios en las capas se realizan mediante el diseño de protocolos independientes para las diferentes capas (Srivastava & Motani, 2005).

De manera general se describe la tarea global para cada una de las capas de la Figura 15 (Wetteroth, 2001): (1) la capa de física es la que se encarga del uso del medio para la transmisión, (2) la capa de enlace se encarga de las conexión entre equipos adyacentes con el direccionamiento físico, (3) la capa de

red se encarga de la interconexión diferentes redes con el direccionamiento lógico, (4) la capa de transporte se encarga de mantener la conexión extremo a extremo y la fiabilidad de los datos, (5) la capa de sesión se encarga de la comunicación entre dispositivos en la red, (6) la capa de presentación se encarga de darle el formato y representación a los datos, (7) la capa de aplicación se encarga de realizar los procesos del usuario y ofrecer los servicios de red.

Un protocolo en una capa o parte de una capa, se implementa en una entidad que se comunica con otras entidades (en otros sistemas en red). Las unidades de datos del protocolo (PDU) son construidas por la carga útil (generada por una entidad en una capa adyacente superior) y cabecera (que contiene información del protocolo). El formato del PDU, así como la definición del servicio se especifica por el protocolo en un determinado nivel de la pila (Zorba, N. , Skianis C. and Verikoukis, 2011).

2.6.2 Diseño de capas cruzadas (Cross-Layer Design)

La arquitectura en capas del modelo OSI prohíbe la comunicación directa entre capas no adyacentes; la comunicación entre capas adyacentes se limita a llamadas a procedimientos y respuestas, denominadas primitivas (Srivastava & Motani, 2005).

Para superar estas limitaciones se ha propuesto una modificación del paradigma de capas, denominado diseño de capas cruzadas. La idea central es la de mantener las funcionalidades asociadas a las capas originales, adicionalmente que permita la coordinación, la interacción y la optimización de protocolos que cruzan diferentes capas (Zorba, N. , Skianis C. and Verikoukis, 2011).

En términos generales, el diseño de capas cruzadas se refiere al desarrollo de protocolos realizando la explotación, de manera activa, de la dependencia entre las capas de protocolo para obtener mejoras en el rendimiento, donde los protocolos de las diferentes capas están diseñados de forma independiente (Srivastava & Motani, 2005).

En un diseño de capas cruzadas, los protocolos pueden ser diseñados para contravenir la arquitectura de capas del modelo OSI, por ejemplo, al permitir la comunicación directa entre los protocolos de capas no adyacentes o compartiendo las variables entre las capas (Srivastava & Motani, 2005).

La arquitectura de capas puede ser contravenida de cuatro formas, las cuales se describe a continuación (Srivastava & Motani, 2005):

- **Creación de nuevas interfaces**: Varios diseños de capas cruzadas requieren creación de nuevas interfaces entre las capas, como se ilustran en la Figura 16a. Las nuevas interfaces se utilizan para el intercambio de información entre las capas. La creación de una interfaz nueva no está considerada en la arquitectura en capas del modelo OSI.

- **Fusión de capas adyacentes**: El diseñar dos o más capas adyacentes entre sí de tal manera que el servicio prestado por la nueva súper capa, sea la unión de los servicios prestados por las capas que la constituyen. Esto no requiere que se creen nuevas interfaces. La súper capa puede interactuar con el resto de la pila utilizando las interfaces que ya existen en la arquitectura original, como se ilustran en la Figura 16b.

- **Acoplamiento diseño sin nuevas interfaces**: El diseño consiste en acoplar dos o más capas, sin crear las interfaces adicionales para compartir la información en el tiempo de la ejecución. Mientras que no se crean nuevas interfaces, no es posible reemplazar una capa sin realizar cambios correspondientes a otra capa, como se ilustran en la Figura 16c.

- **Calibración vertical a través de capas**: Se refiere al ajuste de los parámetros que se extienden a través de las capas. El rendimiento visto a nivel de la aplicación es una función de los parámetros de todas las capas por debajo de ella, como se ilustran en la Figura 16d.

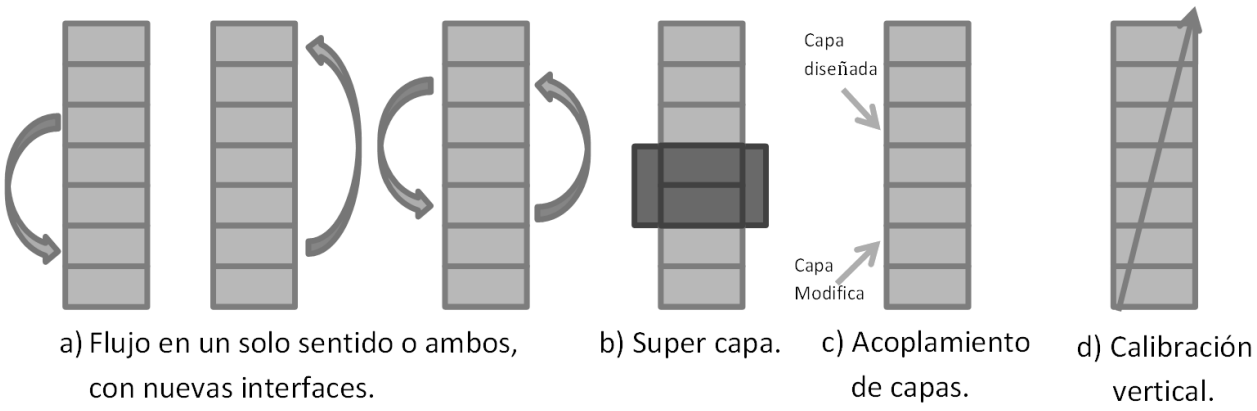

Figura 16. Diferentes formas de violar la arquitectura en capas (Srivastava & Motani, 2005).

Capítulo 3. Estado del Arte SDN

En los últimos 20 años, los enrutadores han evolucionado para incluir más servicios, como parte de la función básica para determinar la ruta y la conmutación de paquetes, agregando funciones de capas superiores (capas de transporte, sesión, presentación y aplicación), cuando normalmente trabajan en las capas inferiores (capas de red, enlace y física) (World Wide Technology, 2014).

En las redes tradicionales, el plano de reenvío, el control, la gestión y los servicios están integrados en gran medida en cada dispositivo de red. Por otro lado, el modelo de SDN despliega la mayor parte de la gestión, los servicios y las decisiones de selección de ruta, como una función avanzada en un controlador o en un dispositivo independiente (World Wide Technology, 2014).

En los casos de uso de las SDN, es importante entender cuáles funciones proporcionan el mayor valor agregado. En esta sección se mencionarán algunos casos de uso de SDN, diferentes alternativas de implementación, soluciones y protocolos para entornos de SDN, además de algunos proveedores.

En la guía de SDN del 2015 (Metzler & Metzler, 2015) se realizaron varias encuestas, en relación al uso e implementación de SDN. De acuerdo a los resultados de dichas encuestas, la Tabla 1 muestra el porcentaje de uso de SDN, y en donde se ha implementado. Se puede apreciar que la mayor atención la reciben los centros de datos, además de la implementación de SDN en las redes de sucursales y campus, al igual que en las redes de cobertura amplia (WAN, por sus siglas en inglés) (Metzler & Metzler, 2015).

Tabla 1. Enfoque de implementación SDN (Metzler & Metzler, 2015).

Enfoque de implementación SDN	Porcentaje
Centros de Datos.	64 %
WAN.	26 %
Sucursales y/o Campus.	25 %
Es poco probable que implementen SDN en los próximos dos años.	12 %
No saben /NA.	10 %
Es muy probable implementar un servicio de un proveedor de servicios WAN con base en SDN.	8 %
Otros.	6 %

A continuación se mencionan algunos de los casos de uso SDN que se han implementado en los centros de datos, WAN y en campus (Metzler & Metzler, 2015):

- Centros de datos, (SDC, SDDB).
- Migración a máquinas virtuales
- Encadenamiento de servicios
- Servicios de seguridad
- Servicios balanceo de carga
- Nuevo modelo de la nube definida por software
- Nuevos modelos de la nube distribuida para hosting
- WAN
- Acceso basado en roles
- Enlaces troncales de Google G-Scale WAN
- Campus
- QoS dinámico e Ingeniería de Tráfico
- Redes cableadas e inalámbricas unificadas
- Gestión de calidad de servicio para Microsoft Lync a través de redes cableadas e inalámbricas
- Aplicaciones basadas en SDN (como Bonjour de Apple)

En general, en la Figura 17 se ilustran los entornos en donde se han implementado las SDN con mayor fuerza (Metzler & Metzler, 2015):

Figura 17. Ramificación de las aplicaciones y enfoques para las redes definidas por software (Fujitsu, 2014).

Debido al interés que surgió por las SDN, como la implementación, la simulación, etc. Se han realizado una serie de propuestas como alternativas para el plano de datos y el plano de control, así como para las interfaces *northbound* y *southbound*. En esta sección se presentan las distintas opciones que existen para las SDN, como en los planos, las interfaces y los simuladores.

3.1 Plano de datos.

El plano de datos, se podría dividir en dos clasificaciones: los dispositivos con interfaces físicas, y los conmutadores virtuales. En los dispositivos con interfaces físicas (del inglés hardware), la mayor implementación es hacia los equipos que cuentan con soporte habilitado para OpenFlow. Aunque no es la única *southbound* que existe, es la más conocida e implementada. Existen otros dispositivos físicos que utilicen alguna otra Interface Southbound, libres o propietaria de cada proveedor.

En el mercado existe una gran cantidad de dispositivos certificados que soportan OpenFlow en la página oficial de la ONF, la lista se puede consultar en la página oficial[3]. Existe una gran variedad de productos para diferentes soluciones e implementaciones.

Las compañías que desarrollan y manufacturan estos equipos son los principales proveedores de soluciones de red y software, sumándose algunas empresas emergentes, en la Tabla 2 se mencionan algunas de ellas.

Tabla 2. Fabricantes y proveedores de soluciones SDN. (DeCusatis, 2013)

Chips	Conmutadores	Servicios de Red	Controladores SDN	Redes Ópticas	Virtualización	Otros
Broadcom Marvell Mellanox Intel Xilinx Ezchip Tech.	Dell/Force 10 NEC Cisco Pica8 HP Extreme Alcatel-Lucent	Citrix A10 Radware Embrane HP Cisco	Big Switch OpenDayligth VMware HP Cisco NEC	Infinera ZTE Cyan Ciena ADVA Optical	Citrix Microsoft VMware	IBM Ericsson Juniper Huawei Brocade Arista Plexxi Centec Networks

Un conmutador virtual es la virtualización de un conmutador dentro de un sistema físico, tiene el mismo funcionamiento de un conmutador físico dentro del sistema, el cual reenvía el tráfico entre diferentes máquinas virtuales (VM) en el mismo sistema físico, así como también reenvía tráfico entre las máquinas virtuales y la red física (Pfaff, B. Davie, & VMware, 2013). Estos conmutadores virtuales se utilizan en diferentes plataformas para realizar emulaciones y simulaciones.

La diferencia entre simulación y emulación, es que la simulación solo trata de reproducir o fingir el comportamiento de la red con un bajo nivel de realismo. La emulación permite imitar o modelar el comportamiento de la red con un alto nivel de realismo en una plataforma virtual, de manera que funcione como si se estuviera utilizando una red real.

[3] https://www.opennetworking.org/

Para los conmutadores virtuales existe una gran tendencia, OpenvSwitch, rebautizado por VMware como NSX vSwitch, el cual es un conmutador virtual de código abierto diseñado para ser utilizado como un conmutador virtual OpenFlow en entornos de servidores virtualizados.

GENI[4] (Berman et al., 2014) (del inglés Global Enviroment for Network Innovations), es una plataforma que proporciona un laboratorio virtual desde una conexión remota, con una gran escala y equipo real distribuido por todo Estados Unidos, se utiliza para la investigación y para la educación, además permite evaluar diferentes entornos de red y probar nuevos protocolos en la misma.

OFELIA[5] , es un proyecto europeo que proporciona instalaciones experimentales basadas en OpenFlow. Una de las opciones más utilizada es Mininet[6] (Handigol, Heller, Jeyakumar, Lantz, & Mckeown, 2012), es un emulador SDN. Con esta herramienta se crea una red virtual realista, como conmutadores y códigos de aplicación.

Finalmente EstiNet[7], que al igual que Mininet es un emulador y además un simulador para redes SDN, es capaz de simular conmutadores OpenFlow y ejecutar los controladores del mundo real como NOX, POX, Floodlight[8], y Ryu[9].

3.2 Plano de Control.

Existen controladores que soportan más de una Interface Southbound, como los promovidos por el proyecto OpenDayligth y Cisco, así mismo otros proveedores, buscan darle competencia y ofrecer alternativas diferentes a OpenFlow. Aunque la gran mayoría de los controladores tienen soporte para OpenFlow, el cual consideran como un estándar para utilizar las SDN.

En la Tabla 3 y 4 se mencionan algunos de los controladores propuestos y sus características más relevantes que se encuentran disponibles en el mercado, además algunos de ellos continúan en desarrollo para su mejora.

---

[4] https://www.geni.net
[5] http://www.fp7-ofelia.eu/
[6] http://mininet.org/
[7] http://www.estinet.com/
[8] http://www.projectfloodlight.org/floodlight/
[9] https://osrg.github.io/ryu/

Tabla 3. Características de algunas alternativas de controladores SDN.

Nombre	Arquitectura		Lenguaje Prog.	Southbound	Grupo de trabajo
NOX (Gude et al., 2008)	Centralizado	Multihilo	C++, Python	OpenFlow, Ethane	Niciria
POX (Prete, Shinoda, Schweitzer, & De Oliveira, 2014)	Centralizado	Centralizado	Python	OpenFlow	Niciria
HP VAN SDN (Hewlett-Packard, 2013)	Distribuido	Multihilo	Java	OpenFlow, y otros SB.	HP
Beacon (Erickson, 2013)	Centralizado	Multihilo	Java	OpenFlow	Standford
Floodligth [10]	Centralizado	Multihilo	Java	OpenFlow, y otras redes.	Big Switch Network
HyperFlow (Tootoonchian & Ganjali, 2010)	Distribuido	Multihilo	Python, C++	OpenFlow	University of Toronto
Kandoo [11] (Yeganeh & Ganjali, 2012)	Distribuido	Multihilo	Go	OpenFlow	University of Toronto
ONIX (Koponen et al., 2010)	Distribuido	Centralizado	C++, Python, Java	OpenFlow	Niciria
Maestro[12]	Centralizado	Multihilo	Java	OpenFlow	Rice University
OpenDaylight (Hydrogen, Helium, Lithium)	Distribuido	Centralizado	Java, C	OpenFlow, y otros SB.	Linux Foundation, NEC
Ryu [13]	Centralizado	Multihilo	Java	OpenFlow, OVSDB	NTT
Trema (Dietz, 2012)	Centralizado	Multihilo	Ruby , C	OpenFlow	Sin info
Hedera (Al-Fares, Radhakrishnan, & Raghavan, 2010)	Distribuido	Centralizado	Python	OpenFlow	University of California San Diego
Cisco One (Cisco, 2015)	Distribuido	Centralizado	Java, Python	OpenFlow, y otros SB.	Cisco

Tabla 4. Alternativas de controladores SDN.

4D (Greenberg et al., 2005)	DevoFlow (Mogul et al., 2010)	NSX Controller (VMware, 2015)	Junos NorthStar Controller [14] [15]	Oracle SDN (ORACLE, 2014)
Ericsson SDN controller (Ab, 2014)	SOX (Luo et al., 2012)	ETHANOL (Moura, Bessa, Vieira, & Macedo, 2015)	OpenContrail (Singla, A., & Rijsman, 2013)	BalanceFlow (Hu, Wang, Gong, Que, & Cheng, 2013)
NOX-MT (Tootoonchian, Gorbunov, Ganjali, Casado, & Sherwood, 2012)	Big Network Controller	DIFANE (M. Yu, Rexford, Freedman, & Wang, 2010)	Rosemary (Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., ... & Kang, 2014)	Mahout (Curtis, Kim, & Yalagandula, 2011)
NSX controller	Fortnox	SmarthLigth	Ryu	ONOS
PANE	NVP Controller	NOX-MT	Oracle SDN	RouteFlow
Sflow	PolicyCop	Picos8	OMNI	MUL
Hyfi	OFNIC [16]	Fleet	ClosedFlow	Arista
DISCO	Meridian	BalanceFlow	DevoFlow	Programable Flow
OpenNass	Brocade	Helios	MobileFlow	Etc.

[10] http://www.projectfloodlight.org/floodlight/
[11] http://www.kandoo.org/
[12] https://code.google.com/p/maestro-platform/
[13] https://osrg.github.io/ryu-book/en/Ryubook.pdf
[14] https://www.opennetworking.org/products-listing/juniper-northstar-controller
[15] http://www.juniper.net/us/en/products-services/sdn/northstar-network-controller/
[16] http://fiware-ofnic.readthedocs.io/en/latest/Installation_and_administration_manual.html

3.3 Interfaces del norte.

Las SDN permiten el control de la red para ser aprovechada por aplicaciones de terceros a través de la API de la NBI. Algunas de las APIS utilizadas en las NBI, son REST API, Java API, y Python API. De las API anteriores, la REST proporciona una integración simple y permite una interacción con mínima sobrecarga entre clientes y servidores. Esa es la razón por la que muchas grandes empresas como Facebook y Google utilizan API REST para ofrecer sus servicios. La misma razón también se aplica en SDN; la mayoría de los controladores de SDN hoy utilizan REST API como NBI para proporcionar información de la red a las instancias de terceros (Oktian, Lee, Lee, & Lam, 2015).

El papel de la NBI en la SDN es proporcionar una API de alto nivel entre el controlador y las aplicaciones para facilitar el cálculo de algunas operaciones de la red según los eventos en el circuito de control (Zhou, Li, Luo, & Chou, 2014).

REST es un estilo de arquitectura para el diseño de aplicaciones en red. Ha ganado popularidad en la implementación de sistemas débilmente acoplados. Los servicios REST se están convirtiendo en el estilo de elección para la API de las NBI y ganando cada vez más importancia en la arquitectura de las SDN. La adopción de REST para SDN en la NBI dentro de esta arquitectura de control tiene las siguientes ventajas (Zhou et al., 2014):

1) REST no utiliza ningún tipo de registro de recursos centralizado. Permite que los elementos de red, como enrutadores, conmutadores, dispositivos intermedios (por ejemplo, dispositivos NAT y DPI), se desplieguen y se cambian dinámicamente de manera distribuida.

2) REST separa las representaciones de recursos, la identificación e interacción, puede ajustar las representaciones de recursos y protocolos de red basado en las capacidades del cliente SDN y condiciones de la red para optimizar el rendimiento de la API.

3) La tendencia actual en SDN, es el uso de la composición de programación para lograr la flexibilidad funcional. REST puede proporcionar composiciones orientadas a servicios que son independientes de los lenguajes de programación y se pueden ejecutar en diferentes plataformas.

4) REST API admite la migración de servicios compatibles con versiones anteriores a través de la migración localizada por el cual un recurso que acaba de agregar sólo afecta a los recursos que se

conectan a ella. Combinado con la interfaz uniforme, puede aliviar la tensión entre los nuevos despliegues de servicio y compatibilidad con versiones anteriores.

5) REST logra la escalabilidad del servidor manteniendo el servidor sin estado y mejora el rendimiento del servidor a través de cachés en capas. Esta característica es útil, cuando un controlador de una SDN tiene que soportar un gran número de aplicaciones basadas en host simultáneas y utilizar recursos de la red de una manera eficiente. Para hacer realidad estos beneficios y ventajas de REST, necesita ser mantenido en el diseño de una API REST escalable y orientada al servicio un conjunto de restricciones REST.

3.4 Interfaces sur.

Las interfaces Southbound básicamente son un protocolo de transporte, y son el puente que conecta el controlador con los dispositivos de reenvío en el plano de datos en una arquitectura de SDN. Además del protocolo OpenFlow, se están utilizando otros protocolos existentes de red, como OSPF, MPLS, BGP, e IS-IS con el propósito de encontrar formas de ejecutar un entorno SDN en un modo híbrido.

Existen diferentes propuestas de interfaces Southbound, algunas de ellas son: JumpFlow (Z. Guo et al., 2015), SSH, SNMP, Cisco onePK (Kiran, S., & Kinghorn, 2015), ROFL[17] (Revised OpenFlow Library set), HAL (Belter, Parniewicz, et al., 2014), PAD (Belter, Binczewski, et al., 2014), Brocade OpenScript (Delivery, 2012), XMPP (Singla, A., & Rijsman, 2013) entre otras más. Algunas de estas propuestas se describen en la Tabla 5.

3.5 Simuladores SDN

Existen varias plataformas de software para realizar simulaciones o emulaciones de una SDN utilizando el soporte para el protocolo OpenFlow. Tomando un extracto de (Al-somaidai & Yahya, 2014) y de la página de Internet de herramientas para simulaciones de redes[18], los simuladores como Trema, Opnet, Qualnet, Onesim, Peersim y Psim, en sus páginas oficiales y documentación no se menciona que cuenten con el soporte a SDN como se muestra en la Tabla 6.

[17] http://roflibs.org/
[18] https://networksimulationtools.com/

Tabla 5. Características de alternativas de interfaces Southbound.

Southbound Interface	Año	Promotores Principales	Características
ForCES (RFC5812) (Yang, L., Dantu, R., Anderson, T., & Gopal, 2004)	2003	IETF.	-Primero en proponer la separación del plano de control y de reenvió. -Se usa como conceptualización ya que no existe una implementación real, continua activo y en desarrollo.
NETCONF (RFC4741) (R. Enns & Juniper, 2006)	2006	IETF, CISCO, OpenDayligth.	-Comunicaciones entre dispositivo para realizar cambios en la configuración de red. -Lenguaje XML. -Lenguaje de modelado de datos YANG. -Continua activo y en desarrollo.
LISP (RFC6830) (Farinacci, Fuller, Meyer, Lewis, & Cisco Systems, 2013)	2006	IETF, CISCO, OpenDayligth.	-Especifica la separación entre los localizadores y las identidades. -Apoyo en la cartografía de flujos. -No existe una implementación real, pero continua activo y en desarrollo.
Ethane (Casado et al., 2007)	2007	ONF, OpenDayligth.	-3 principios fundamentales: (1) La red debe ser gobernada por políticas de una entidad superior. (2) El ruteo en la red debe conocer estas políticas. (3) La red debe reforzar la unión entre los paquetes y su origen. -Predecesor de OpenFlow.
OpenFlow (Mckeown et al., 2008)	2008	ONF, OpenDayligth.	-Primero en recibir el termino SDN. -Permite experimentar e innovar nuevos protocolos en la redes. -Este protocolo define la comunicación entre el controlador y el conmutador. -Los conmutadores OpenFlow están basados en los conmutadores Ethane. -Distribuido comercialmente por proveedores de equipos de red y organizaciones que promueven el desarrollo de nuevos protocolos. -Continúa activo y en desarrollo.
OVSDB (Pfaff et al., 2013)	2012	OpenDayligth, VMware.	-Versión Virtualizada de OpenFlow, con mayores características. -Gestión avanzada para OpenVSwitch. Capacidades: -Crear varios conmutadores virtuales. -Establecer las políticas de calidad de los servicios en las interfaces. -Adjuntar las interfaces a los conmutadores. -Configurar interfaces de túnel en rutas de datos OpenFlow, gestionar las colas. -Recoger las estadísticas.
POF (Song, 2013)	2013	POForwarding Org., Huawei.	-Uno de los primeros competidores directos de OpenFlow. -Propone un conjunto de instrucciones genéricas de flujos. -El elemento de reenvío POF no que entender el formato del paquete, como en OpenFlow. -El dispositivo de reenvío es visto como una caja blanca, solo con capacidades de procesamiento y reenvío.
OpFlex (Cisco, 2014)	2014	Cisco, Microsoft, Red Hat, Citrix IETF.	-Protocolo de transporte para transferir las políticas, eventos, estadísticas e información de fallo, de un controlador hacia a un conjunto de dispositivos inteligentes de red. -Centraliza el control de las políticas. -Basado en los protocolos de control de red tradicional y distribuido.
OpenState (Bianchi, Bonola, Capone, & Cascone, 2014)	2014	OpenState Org.	-Basado en OpenFlow. -Otorgo lógica de control a los conmutadores. -Descentraliza un poco el control. -Basado en una máquina de estados infinitos.
I2RS[19]			-Inyecta o recupera información. -Control y analizar las operaciones de BGP. -Optimizar y elegir los puntos de salida de la red. -Apoyar una reacción rápida distribuida a los ataques basados en la red. -Redirigir el tráfico para el destino bajo ataque.

Tabla 6. Características de alternativas de controladores SDN.

Plataforma	Sistema Operativo	Openflow version	Licencia	Lenguaje	Soporte SDN confirmado
Mininet	Linux	1.0, 1.1,1.2,1.3	Codigo abierto	Python	SDN
Estinet	Fedora	1.0,1.3	Propietaria	Sin info	SDN
Ns3	Linux	Sin info	Codigo abierto	Python	SDN
Ns2	Linux	Sin info	Codigo abierto	TCL	SDN
Omnet++	Windows,Mac,Linux	1.0,1.3	Codigo abierto	C++	SDN
Trema	Linux	Sin info	Codigo abierto	Sin info	X
Opnet	Windows	Sin info	Propietaria	C++	X
Qualnet	Windows,Mac	Sin info	Propietaria	C++	X
Onesim	Windows,Mac,Linux	Sin info	Codigo abierto	Java	X
Peersim	Windows,Linux	Sin info	Codigo abierto	Java	X
Psim	Windows,Mac,Linux	Sin info	Codigo abierto	Java	X

[19] http://searchsdn.techtarget.com/tip/IETF-SDN-I2RS-uses-traditional-routing-protocols-in-software-networks

Por otro lado, Omnet++, en su paquete de distribución, no incluye la funcionalidad para simular SDN, sin embargo existen extensiones que permiten simular el protocolo OpenFlow, tanto en la versión 1.0 como 1.3. Sin embargo, no fue posible descargar la extensión de la versión 1.3 (Salih, Cosmas, & Zhang, 2015) para evaluarla y la instalación de la versión 1.0 puede llevar mucho tiempo para realizar una instalación correcta.

El simulador de redes NS3 es una versión reescrita de NS2, permite el desarrollo de modelos de simulación de alto desempeño, lo que habilita el uso de la emulación, se usan principalmente en ambientes educativos y de investigación. En NS3 se tiene soporte para OpenFlow y NS2 tiene una funcionalidad más sencilla, sin embargo no tiene soporte para OpenFlow, por lo cual es necesario escribir todo el script de código de la topología, flujos de información, monitoreo, protocolos y el comportamiento de la red para trabajar en un esquema SDN.

Estinet es una plataforma virtual de redes, cuenta con el modo de simulación y emulación, se requiere licencia aunque cuenta con un periodo de prueba con recursos limitados. Aunque tiene detalles de funcionamiento con algunos controladores como Floodlight, cuenta con herramientas de monitoreo y configuración de la red.

Mininet es la plataforma de emulación de código abierto y es una de las plataformas más utilizadas para trabajar con OpenFlow. Dada su naturaleza de código abierto, el soporte técnico y asesoramiento es proporcionado por los mismos usuarios y la comunidad de desarrollo. Es una plataforma muy limitada en el entorno gráfico y configuración, soporta una cantidad moderada de nodos sin problemas (Handigol et al., 2012), sin embargo se requiere un monitoreo externo ya que no cuenta con esta función.

3.6 Calidad de servicio en redes definidas por software.

A continuación se mencionan algunas de las propuestas más relevantes, donde se modela la arquitectura de una SDN con un enfoque de calidad de servicio (Qos), que servirán como medio de comparación y apoyo para sustentar una nueva propuesta.

OpenQos (H.E. Egilmez & Dane, 2012)**:** En esta propuesta se diseña un controlador que permite la calidad de servicio (Qos) para la entrega de servicios multimedia a través de redes con arquitectura OpenFlow donde los flujos multimedia se colocan dinámicamente en rutas donde se garantiza la calidad

de servicio, efectos adversos minimos sobre otros tipos de tráfico en la red. Además es comparado con otros modelos de QoS de las redes convencionales como IntServ y DiffServ.

OpenADN (Paul & Jain, 2012)**:** El enfoque de este diseño permite a los proveedores de servicios para aplicación (ASP, por sus siglas en inglés) hacer cumplir las políticas de gestión del tráfico de aplicaciones y limitaciones de entrega de aplicaciones en el nivel requerido. Está diseñado en el marco de una SDN que permite a cada ASP adaptarse a los requisitos específicos de la aplicación. Las entidades del plano de datos pueden pertenecer a la misma ASP o pueden ser delegadas a los proveedores de terceros, como proveedores de servicios de Internet (ISP, por sus siglas en inglés) o proveedores de servicios de la nube (CSP, por sus siglas en inglés). Además utiliza el diseño de capas cruzadas que permite que la información del flujo de tráfico sea enviada como etiqueta donde solo los dispositivos conscientes de OpenADN pueden interpretarla.

OpenSession (Arefin, Rivas, Tabassum, & Nahrstedt, 2013)**:** Propone un protocolo de control de sesión de la red implementando el diseño de capas cruzadas, donde se aprovecha el apoyo de las SDN, para múltiples trasmisiones, múltiples sitios, teleimmersion 3D (3DTI), entre otros. Además, mejora la interactividad, la utilización y la escalabilidad de los recursos, agregando características que colaboran con las aplicaciones que ocurren en tiempo real.

FlowQos (Seddiki et al., 2014)**:** Esquema de calidad de servicio SDN que toma los parámetros tradicionales para medir el rendimiento (caudal eficaz, tasa de transmisión, retardo, fluctuación del retardo), pero solo para la red de acceso de banda ancha, como lo es ADSL.

ARVS (Tsung-Feng Yu, Kuochen Wang, & Yi-Huai Hsu, 2015)**:** Propone un enfoque de enrutamiento adaptativo para transmisiones de video con calidad de servicio (QoS) a través de redes definidas por software (SDN). Se realiza una comparación concreta contra OpenQos mejorando la tasa de pérdida de paquetes, además de la carga en la red evitando la congestión de la ruta más corta. En la Tabla 7 se muestran las características generales de cada propuesta descrita anteriormente.

Algunos de los puntos importantes detectados que deben tomarse en cuenta son los siguientes:

- Los controladores SDN no consideran la calidad de servicio, las propuestas existentes implementan un módulo en el controlador adicional para proporcionar calidad de servicio. OpenFlow tiene un soporte limitado de calidad de servicio por colas simples para reservación de la tasa de transmisión, así como el

monitoreo de este mismo y la pérdida de paquetes. Generalmente es utilizado un sistema de monitoreo externo al controlador.

- El enfoque de gran parte de los trabajos va dirigido al enrutamiento de los flujos para mejorar la transmisión multimedia (video streaming). Además no se consideran flujos de información de otro tipo de aplicaciones, ni la prioridad de las aplicaciones.
- La tasa de transmisión es el parámetro que presenta mayor relevancia, por lo que gran parte del objetivo de QoS se enfoca en él.

Tabla 7. Algunas soluciones propuestas para calidad de servicio.

Modelo	Enfoque	Southbound api	Controlador	Año	Lenguaje	Características
OpenQos	Controlador Qos routing	OpenFlow	Floodligth	2012	Java	Aplicado en conmutadores físicos en ambiente de prueba
OpenADN	Aplicación	OpenFlow	propio	2012	Sin info	Usa cross-layer design usando etiqueta ente capa 3 y 4.
OpenSession	Controlador Tablas de reenvió	OpenFlow	Floodligth	2013	Java	Implementado en conmutadores físicos.
FlowQos	Red de Acceso	OpenvSwitch	Pox	2014	Python	Utiliza Openwrt
ARVS	Routing	OpenFlow	Floodligth	2015	Sin info	Emulación mininet

Esta página se ha dejado intencionalmente en blanco.

Capítulo 4. Propuesta de mejora del controlador SDN en el proceso de selección de la ruta con mejores condiciones.

La arquitectura actual de las SDN no considera un mecanismo para ofrecer niveles de calidad de servicio. Debido a que OpenFlow proporciona un soporte muy limitado, se requieren elementos adicionales para ofrecer QoS en una arquitectura de SDN.

Las funcionalidades del controlador no consideran, los parámetros de QoS para enrutar o clasificar los flujos de información. En la actualidad, existen diferentes arquitecturas y herramientas para proporcionar QoS, sin embargo, no han tenido un éxito real para su implementación.

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 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 (Quintana, 2001). Además de la tasa de transmisión, los enlaces disponen de una serie de propiedades que sirven como métricas del desempeño (Quintana, 2001).

El enrutamiento con calidad de servicio presenta algunas desventajas, la mayor de las cuales es la exigencia de conocer el estado completo de la red en cada nodo para poder determinar un camino válido (Quintana, 2001).

En este trabajo se presenta como propuesta un algoritmo para realizar el cálculo de estimación de retardo, latencia y fluctuaciones del retardo en el proceso de selección de la trayectoria entre nodos. Así, el controlador tendrá una visión global, completa y actualizada de la red mediante técnicas de diseño de capas cruzadas, donde se abstraen las métricas de la red y los requerimientos de las aplicaciones, con base en la tasa de transmisión, la pérdida de paquetes, el retardo y la fluctuación del retardo, que son definidas como sensibles para las aplicaciones de alto desempeño en la sección 2.4.

4.1 Consideraciones en SDN

Las SDN así como las redes estándar, utiliza como método de enrutamiento el algoritmo de la ruta más corta o también conocido como de costos mínimos (Algoritmo Dijkstra). Este algoritmo explora los caminos adyacentes partiendo del nodo origen, donde busca el de menor valor. Se utilizan como métrica la distancia o costo del enlace, así como los saltos para llegar al destino.

En un controlador SDN, la trayectoria seleccionada entre un nodo origen y un nodo destino, siempre se elige la ruta la más corta, se utiliza esta ruta para enviar la información sin revisar si los enlaces se encuentras congestionados. Esto lleva a los enlaces al punto de la saturación, por lo tanto a la degradación del servicio que proporciona la red.

El mayor enfoque que existe para resolver esta problemática es aumentar, monitorear o controlar solo las tasas de transmisión y para conocer las condiciones de la red es necesario utilizar más de una sola métrica. Existen diferentes soluciones de monitoreo o enrutamiento con calidad de servicio, sin embargo ninguna utilizada con gran éxito, por la complejidad y procesamiento que estas requieren.

4.1.1 Enrutamiento en SDN

Los algoritmos de enrutamiento utilizados en los controladores se basan en el algoritmo de Dijkstra, este algoritmo permite obtener la ruta más corta hacia el destino, la cual usa como métrica la distancia, el costo o saltos. Este tipo de algoritmos no consideran los parámetros de calidad de servicio para generar sus rutas, por lo tanto es necesario un trabajo fuerte en esta área para agregar estas variables en la toma de decisión de la ruta. Aunque existen trabajos en el enrutamiento con calidad de servicio, la realidad es que no han sido implementados y solo se quedan en planteamientos.

El algoritmo de la ruta más corta se utiliza comúnmente por su rápida respuesta en la selección de la trayectoria, sin embargo sigue el esquema del mejor esfuerzo, donde no se tiene ninguna garantía de que la información llegue, ni que la ruta sea seleccionada con calidad de servicio. La gran desventaja este algoritmo es que no conoce realmente las condiciones en la red y las decisiones que toma no están enfocadas a garantizar el buen desempeño de la red ni el rendimiento de las aplicaciones.

El controlador es un elemento crítico en una arquitectura SDN, es la pieza clave para el apoyo de la lógica de control que genera la configuración de la red con base a las políticas definidas por el operador de red.

La plataforma de control abstrae los detalles de bajo nivel de la conexión y la interacción con dispositivos de reenvío como se ilustra en la Figura 19 (Kreutz et al., 2014).

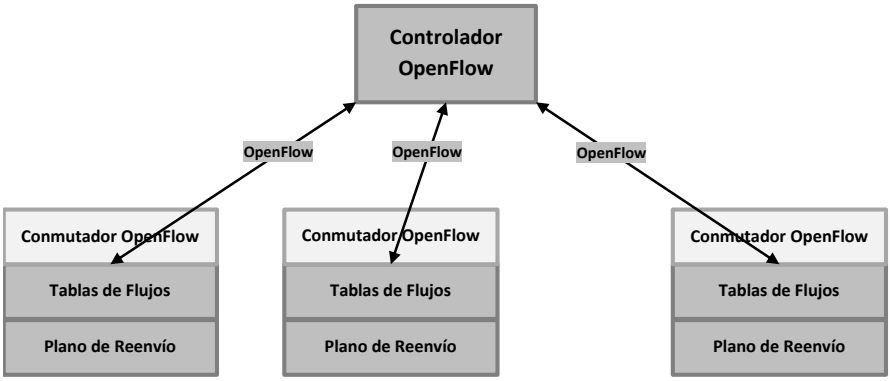

Figura 18. Comunicación entre conmutadores y un controlador OpenFlow (Göransson & Black, 2014).

Floodlight es un controlador utilizado comúnmente como referencia en la literatura, por su estabilidad y amplia documentación. Este controlador utiliza un algoritmo de enrutamiento Dijkstra donde se busca el camino más corto hacia el destino. Por lo tanto los parámetros de calidad de servicio no son considerados en la toma de decisiones al generar la ruta en las tablas de flujo. Un ejemplo simple se muestra en la Figura 20 donde la trayectoria elegida para el nuevo flujo es la ruta más corta.

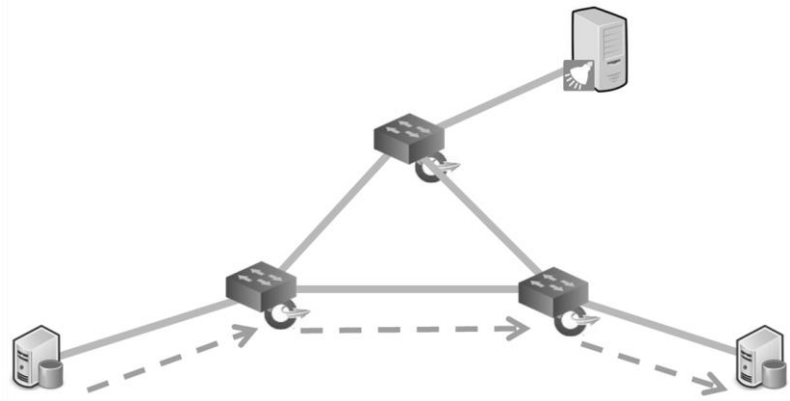

Figura 19. Ruta más corta por algoritmo Dijkstra.

4.1.2 Monitoreo de red en SDN

En su arquitectura original, el monitoreo de la red se realiza a través del plano de aplicación como una entidad externa a la red. Los parámetros obtenidos de este monitoreo pasan por toda la pila de capas o protocolos, por lo que el retardo que este procesamiento genera no permite que el monitoreo sea realmente en tiempo real, y en consecuencia, la aplicación que implementa las políticas de calidad de

servicio en la red tomará decisiones con información precisa pero atrasada. Para hacer más eficiente la toma de decisiones en el ofrecimiento de niveles de calidad de servicio, es necesario un monitoreo de bajo nivel el cual se podría llevar a cabo por medio de OpenFlow.

La aplicación sonda procesa, provee e implementa las políticas que posteriormente se utilizan en el plano de control, creando tablas de enrutamiento que notifican al plano de datos para su ejecución. En la Figura 18 se ilustra una aproximación de la arquitectura SDN en esquema de capas del modelo OSI, esto permite apreciar la interacción entre capas de cualquier aplicación que quiera acceder con la información de la red.

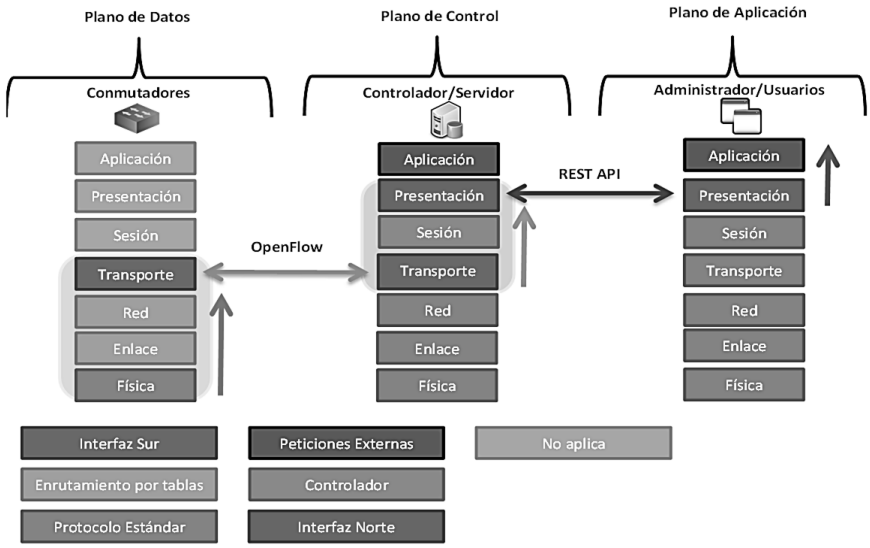

Figura 20. Representación del modelo OSI de la arquitectura SDN y el flujo entre capas.

En el Anexo A se describe los procesos de mensajería de OpenFlow y las diferentes versiones de este. OpenFlow tiene un soporte de calidad de servicio muy limitado a través de un mecanismo simple de colas, que proporciona una tasa de trasmisión mínima garantizada y limitando la tasa de transmisión máxima. Además de un monitoreo nativo por el cual proporciona la información de las tasas de transmisión y la pérdida de paquetes.

4.1.2.1 OpenFlow

A través de OpenFlow el controlador puede obtener los parámetros de la tasa de trasmisión y la pérdida de paquetes en forma directa. Además OpenFlow tiene un soporte limitado de calidad de servicio por medio de colas simples para el control de velocidades de transmisión, así como la información de los flujos.

OpenFlow entrega al controlador métricas como la pérdida de paquetes y tasa de trasmisión (ONF, 2012), sin embargo estas métricas no son utilizadas por el controlador para generar la trayectoria. OpenFlow es un protocolo de transporte que utiliza distintos mensajes descrito en el Anexo A, Cuenta con una gran variedad de tipos de mensajes, pero dos de los mensajes que presentan una especial utilidad para esta tesis, son el de solicitud de eco y el de respuesta de eco (Echo Request/Reply). Los cuales son similares a un ping pero de manera nativa en el protocolo OpenFlow.

Los mensajes de eco (solicitud y respuesta) en una SDN, incluyen marcas de reloj que permiten determinar los retardos de ida y vuelta, entre el controlador (C0) y el conjunto de conmutadores (S1, S2, S3). Así mismo, estos mensajes pueden ser utilizados para verificar conectividad y calcular las tasas de transmisión de la topología. (ONF, 2012). La suma de los retardos de cada enlace (T_x) y las latencias de cada conmutador de la trayectoria (Lat_{sx}) hacia un conmutador destino ($Lat(Sx)$) dan como resultado el retardo total que retorna esta función (RTT), como se muestra en la Figura 21 y la ecuación (21).

$$RTT(Sx) = \sum T_x + 2\sum Lat_{sx} + Lat(Sx) \tag{21}$$

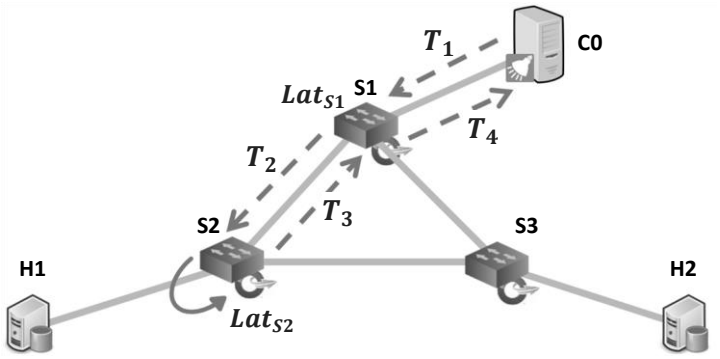

Figura 21. Echo Request/Reply desde el controlador hacia un conmutador.

4.1.2.2 Estimación de retardos en la trayectoria de SDN

Como se describe en (C. Yu et al., 2015), un paquete que atraviesa una trayectoria en la red experimenta retardos por propagación y por conmutación. El retardo de propagación es el tiempo en que el paquete pasa por el medio entre conmutadores y depende de las propiedades físicas del medio de comunicación.

El retardo de conmutación es el tiempo que el paquete pasa dentro de un conmutador y depende de las diversas funciones aplicadas al paquete. Este retardo de conmutación es conocido como latencia. En general, la latencia de un conmutador tiene tres componentes: Búsqueda (Lookup), Reenvío (Forwarding) y Control (C. Yu et al., 2015).

A continuación se describen cada uno de ellos:

Búsqueda: Cuando un conmutador recibe un paquete en uno de sus puertos de entrada, el conmutador busca la coincidencia en su tabla de flujos para determinar a donde reenviar el paquete(C. Yu et al., 2015).

Reenvío: Un paquete es transferido a través del sistema interno del conmutador desde el puerto de entrada a un puerto de salida. Si el puerto de salida está transmitiendo otro paquete, el nuevo paquete se coloca en la cola de salida. El tiempo que un paquete pasa en la cola depende de la cantidad de tráfico que atraviesa el mismo puerto de salida (C. Yu et al., 2015).

Control: Si no existe coincidencia del paquete en las tablas de flujos, el paquete se encapsula en un mensaje de control de OpenFlow (PacketIn) y se envía al controlador. El controlador regresa las tablas de flujos para los nuevos paquetes (C. Yu et al., 2015).

Dado que el retardo de propagación de un enlace depende de las propiedades físicas del medio y la distancia a recorrer, y si se considera que si la distancia y el medio no cambian en los enlaces, se puede suponer que el retardo por propagación es constante. En cambio, la latencia del conmutador estará en función del tipo y cantidad de tráfico que procese. A mayor cantidad de tráfico, es de esperar que la latencia se incremente considerablemente.

Por lo general, las soluciones de monitoreo de red propuestas para las SDN se implementan como agentes externos, así como en (Maugendre, 2015) entre otros, se propone realizar la medición del retardo. Para entender el mecanismo de medición propuesto, considere la red mostrada en la Figura 22. Para medir la latencia (T3) entre los dos conmutadores S1 y S2, el controlador (C0), utilizando mensajes de OpenFlow, envía un paquete a S1 (*PacketOut*) indicando que debe reenviarlo por un determinado puerto hacía el siguiente conmutador (S2). Cuando S2 recibe el paquete, envía un paquete (*PacketIn*) al controlador para indicarle el estado del paquete. Cuando el controlador recibe el paquete *PacketIn*, el controlador puede determinar el retardo total, T_{Total}, el cual será la suma de los retardos *T1, T2* y *T3*, es decir (ecuación 22).

$$T_{total} = T1 + T2 + T3 \quad (22)$$

Para determinar *T1 o T2,* se procede de manera similar, se considera el tiempo (*Ta*) en que el controlador envía un mensaje de solicitud de estado de puerto del conmutador S1 y el tiempo (*Tb*) en que recibe el paquete de respuesta del estado del puerto desde S1, es decir, *T1 = 0.5 (Tb – Ta).* Para *T2 = 0.5 (Ta – Tb).* Para obtener el retardo entre los dos conmutadores, despejamos *T3*: $T3 = T_{Total} - T2 - T1$. Como se puede observar en la Figura 22, en esta propuesta de monitoreo utilizan un esquema tipo simulador, donde cada nodo se encuentra conectado directamente al controlador.

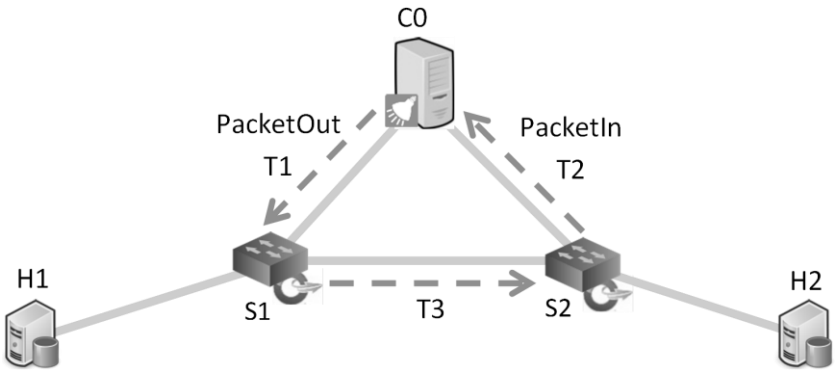

Figura 22. Proceso de estimación del método propuesto en (Maugendre, 2015).

El monitoreo que se realiza en (Maugendre, 2015) es muy cercano al real, sin embargo se realiza para un solo enlace entre dos nodos. Al escalar este método para ***N*** rutas posibles entre ***X*** nodos en una topología en un ambiente distribuido, se incrementa en gran medida el procesamiento y el retardo. Al igual que las propuestas de monitoreo de SDN se requiere un agente externo para manejar este método.

4.2 Estimación de retardos propuesta en la trayectoria de SDN

La propuesta que se hace en el presente trabajo de tesis para la estimación de retardos y latencias, se realiza en el mismo controlador sin la necesidad de un agente externo. La propuesta de monitoreo consiste en calcular los retardos de propagación de los enlaces, los cuales se consideran fijos, así como medir la latencia, mediante el uso de paquetes de eco con OpenFlow (Echo Request/Reply) que el controlador ya utiliza para verificar la conectividad.

El método propuesto es una aproximación de las condiciones que se tienen en la red. Conociendo los retardos en los enlaces y nodos, es posible calcular el retardo total para las ***N*** rutas posibles y con esto determinar la trayectoria que cuente con mejores condiciones.

Algunos artículos como (Rumble, Ongaro, Stutsman, Rosenblum, & Ousterhout, 2011), (Q. Guo, Feng, Wu, & Yu, 2016) y (Cummings, 2006) calculan la latencia de los conmutadores de red y el retardo de propagación, donde la latencia de los conmutadores es de 10-30 µs en (Rumble et al., 2011) y (Cummings, 2006), así como 44.48 µs para un paquete de 512 bytes en (Q. Guo et al., 2016). En general, en un conmutador se puede considerar una latencia estándar muy baja menor a 1 ms por cada nodo, por lo tanto los cálculos realizados para la estimación de la latencia reflejan el comportamiento del tráfico con base en el retardo. En (Rumble et al., 2011) de igual manera calculan la velocidad en la fibra óptica de 5ns/m.

El proceso de estimación del retardo para cada enlace no es exacto ya que influyen factores adicionales, sin embargo se puede considerar que el cálculo de estimación que se propone en este documento es una aproximación al comportamiento que se tiene en la red. Es posible considerar esto con base en la sección 4.1.4, el tráfico es reflejado sobre la latencia de cada conmutador y es una aproximación por que los paquetes de eco, no utilizan los puertos por donde está fluyendo este tráfico, por lo tanto es una estimación de las condiciones en la red.

De manera general, en la Figura 23 se muestra el proceso de estimación, en el cual se tiene un sistema SDN inicializado. Mediante OpenFlow se extrae la información necesaria, como la topología, los valores de RTT hacia cada nodo, entre otros y la distancia de cada enlace se captura adicionalmente. Con esta información se realiza el cálculo y se envía la información al mecanismo que requiera la información, donde se podrá tomar una decisión con respecto a las métricas.

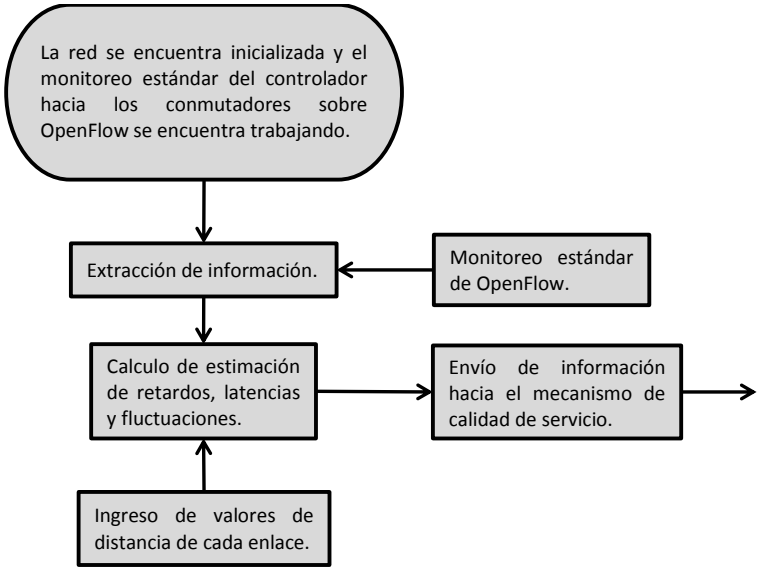

Figura 23. Diagrama de flujos para la estimación de parámetros.

4.2.1 Calculo de retardo por propagación

Para realizar los cálculos del método propuesto se toma como base que los enlaces que se implementan en la nube de Internet generalmente son enlaces de fibra óptica, el cálculo de retardo de propagación para cada enlace requiere el valor de velocidad de la luz a través de la fibra óptica como medio de propagación, el cálculo se realiza con la ecuación (23).

$$C = \frac{c}{i} \tag{23}$$

Dónde:

C = velocidad de la luz en un medio (m/s).

c = velocidad de la luz en espacio libre (299 792 458 m/s).

i = índice de refracción de la fibra óptica (rango típico entre 1.4 a 1.6).

El primer paso es calcular el retardo de propagación para cada en enlace con la ecuación (24), se calcula tomando como valor de entrada la distancia de cada enlace y la velocidad de la luz en el medio de propagación previamente calculado.

$$d_e = \frac{Long(e)}{C} \tag{24}$$

Dónde:

$Long(e) = longitud\ del\ enlace.$

$c = velocidad\ de\ la\ luz.$

4.2.2 Calculo de latencia en cada conmutador

El cálculo de la latencia se puede obtener a partir del cálculo del retardo total de la ecuación (5) y el valor de RTT que se obtiene por mensaje de solicitud de eco. Ya que el RTT incluye las latencias de todos los nodos de la ruta así como los retardos por propagación, podemos expresar e RTT como se indica en la ecuación (25).

$$RTT(Sx) = \sum \overleftarrow{d_{ex}}(p) + \sum \overrightarrow{d_{ex}}(p) + 2\sum Lat(p) + Lat(Sx) \tag{25}$$

La latencia hacia un nodo ***x*** se puede despejar de la ecuación (25) para obtener la ecuación (26), se tiene que al RTT obtenido hacia un nodo ***x*** se le restan los retardos por cada enlace ida y vuelta por donde pasa, además se le resta 2 veces las latencias de los nodos por los que pase en su camino al nodo destino (ida y vuelta), como resultado se tiene una aproximación de la latencia de ese nodo ***x***, este proceso se ilustra en la Figura 21.

$$Lat\ (Sx) = RTT(Sx) - \sum \overleftarrow{d_{ex}}(p) - \sum \overrightarrow{d_{ex}}(p) - 2\sum Lat(p) \tag{26}$$

4.2.3 Calculo de retardo

Al conocer los valores de latencias (l_k) de cada nodo (S_x) y retardo por propagación (d_e) para cada enlace (e_x), se puede estimar el retardo total (D(p)) para diferentes trayectorias como se ilustra en la Figura 24, y así elegir el camino con menor retardo total para un nuevo flujo de información que requiera baja latencia, tomando la suma de los valores de retardo en cada enlace y latencia en cada nodo sobre la ruta representada en la ecuación (27).

$$D(p) = \sum_{e \in p} d_e + \sum_{k \in p} l_k \tag{27}$$

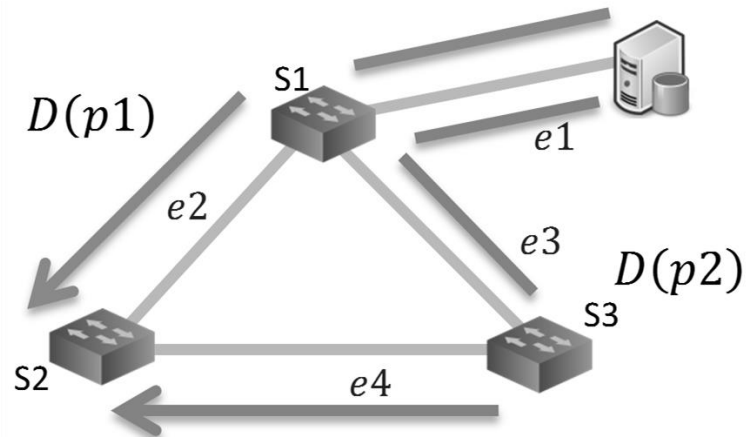

Figura 24. Retardos de diferente trayectoria hacia un mismo destino.

4.2.4 Calculo de las fluctuaciones del retardo

Así como el retardo, es posible calcular una aproximación de sus fluctuaciones. Para obtener el cálculo de la fluctuación del retardo en cada ruta, se toman los valores de retardo previamente calculados. Como se ilustra en la Figura 25, el valor absoluto de la diferencia entre dos valores de retardo en

instantes de tiempo consecutivos, es la fluctuación del retardo en la ruta que se representa en la ecuación (28).

$$J\big(D(p)\big) = |D\ (p(t)) - D\ (p(t-1))| \tag{28}$$

Las fluctuaciones del retardo debe satisfacer una restricción, por ejemplo un valor de J < $\pm\ 30ms$ para la ecuación (17), garantizando un máximo permitido para la ecuación (29).

$$|D1(p) - D2(p)| \geq J \tag{29}$$

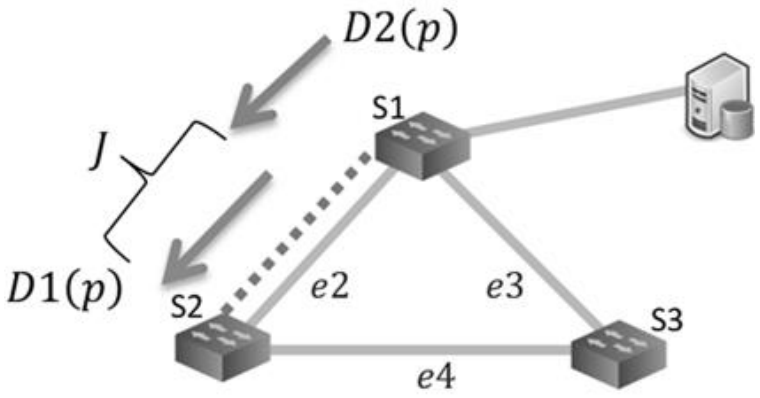

Figura 25. Fluctuaciones de una trayectoria.

4.3 Modificaciones al controlador SDN utilizando técnicas de diseño de capas cruzadas.

La estimación de parámetros ofrece una visión más completa, al agregar las métricas de retardo y fluctuación al conocimiento del controlador, además de la tasa de transmisión y pérdidas de paquetes con las que cuenta. Ya que en el esquema actual de una SDN estas métricas no son consideradas, es necesario implementar un mecanismo en el controlador que permita la consideración de las métricas de QoS para la toma de decisiones.

La propuesta es utilizar técnicas del diseño de capas cruzadas, que añada las funciones de abstracción de información de las aplicaciones, como sus requerimientos de desempeño, así como la abstracción de las condiciones y estado de la red. Esta información será utilizada para la toma de decisiones de enrutamiento y elegir la trayectoria con base a las métricas. En la Figura 26 se muestra la arquitectura de

una SDN donde se agrega un módulo de capas cruzadas. Parte del diseño planteado como propuesta del optimizar se basa en (Olivares, 2008).

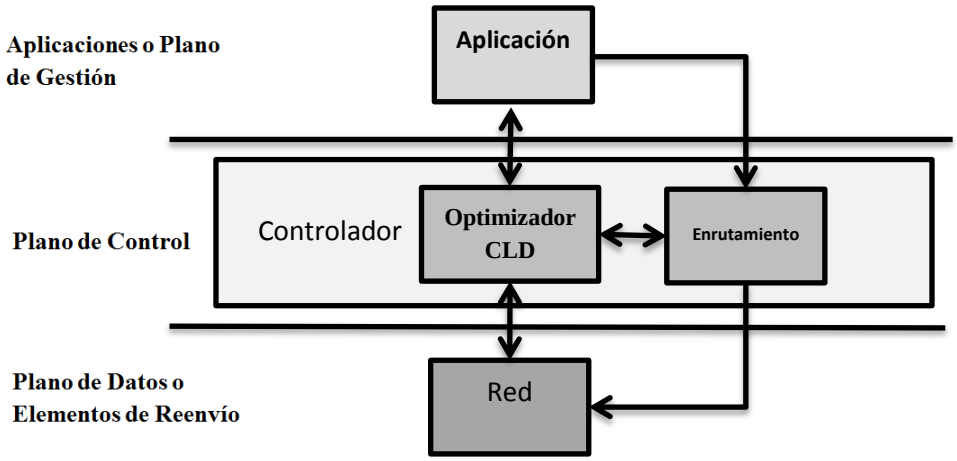

Figura 26. Optimizador CLD en SDN.

En una topología dada existen ***n*** cantidad de trayectorias para un flujo (enlace extremo a extremo), representadas en la ecuación (30).

$$P^{m,k} = \{p_1, p_2, \dots \dots, p_n\} \tag{30}$$

Dónde:

$P^{m,k} = conjunto\ de\ trayectorias\ desde\ la\ fuente\ S^m\ hacia\ el\ nodo\ \mathrm{k}\ en\ la\ sesion$ "m".

$p = camino\ o\ trayectoria\ extremo\ a\ extremo,\ \ p \in\ P^{m,K}$.

Cada trayectoria tiene diversas características, que sirven como métricas o parámetros para conocer las condiciones de la trayectoria a través de su rendimiento. Algunas de las métricas fueron descritas en el capítulo I, como el retardo ***(D(p))***, las fluctuaciones del retardo ***(J(p))***, la pérdida de paquete ***(PLR(p))*** y la tasa de transmisión (por conveniencia de nomenclatura se realiza el cambio de variable donde $\mathrm{B(p)} = X_e^{m,k}$), además se considera la métrica de costo o distancia del enlace ***(Cost(p))***, estas características de cada trayectoria se representan en la ecuación (31).

$$p_n = [Cost(p_n), D(p_n), J(p_n), PLR(p_n), B(p_n)] \tag{31}$$

El algoritmo de enrutamiento estándar basa la elección de la trayectoria en el costo o distancia, que es la métrica utilizada por el controlador para elegir la ruta más corta ***($\tilde{p}$)***, se representa en la ecuación (32) como el mínimo valor de costo entre las ***n*** posibles trayectorias.

$$\tilde{p} = \left[min(Cost\left(P^{m,k}\right))\right] \tag{32}$$

Las métricas que proporcionan las condiciones de rendimiento con las que cuenta la trayectoria no son tomadas en cuenta, el conjunto de algunos de estos parámetros debería considerarse para elegir la trayectoria con mejor rendimiento que podría ser una diferente a la trayectoria más corta. La trayectoria con mejores condiciones ***($\widehat{p_n}$)*** es el camino con los valores de las métricas con mejor valor entre todas las trayectorias, como se representa en la ecuación (33).

$$\widehat{p_n} = \left[\hat{D}(p_n), \hat{J}(p_n), \widehat{PLR}(p_n), \hat{B}(p_n)\right] \tag{33}$$

La ruta más corta ($\tilde{p}$) no siempre es la ruta con mejores condiciones ($\hat{p}$) en una o varias métricas. Para que una trayectoria tenga el rendimiento con mejores condiciones de alguna métrica identificada con un circunflejo ***(^)***, debe tener el menor valor de esa métrica entre las ***n*** trayectorias, representadas en las ecuaciones (34), (35) y (36), en excepción de la tasa de transmisión donde se desea la máxima tasa de transmisión entre las ***n*** rutas, representada en la ecuación (37).

$$\hat{D}(p_n) = \left[min(D\left(P^{m,k}\right))\right] \tag{34}$$

$$\hat{J}(p_n) = \left[min(J\left(P^{m,k}\right))\right] \tag{35}$$

$$\widehat{PLR}(p_n) = \left[min(PLR\left(P^{m,k}\right))\right] \tag{36}$$

$$\hat{B}(p_n) = \left[max(B\left(P^{m,k}\right))\right] \tag{37}$$

Dónde:

$D(p_n) = Retardo\ de\ la\ trayectoria.$

$PLR(p_n) = Pérdida\ de\ paquetes\ de\ la\ trayectoria.$

$B(p_n) = Velocidad\ de\ transmisión\ disponible\ en\ la\ trayectoria.$

$J(p_n) = Fluctuaciones\ en\ la\ trayectoria.$

La red presta el servicio de conexión al conjunto de aplicaciones de los clientes, donde cada aplicación tiene sus propios requerimientos de la red para asegurar un desempeño óptimo, cada aplicación se

caracteriza por su utilidad que define los requerimientos necesarios de la red. Este conjunto de aplicaciones se puede representar con la ecuación (38) y los requerimientos con la ecuación (39).

$$A = \{\widehat{a_1}, \widehat{a_2}, \dots \dots, \widehat{a_n}\} \tag{38}$$

$$\widehat{a_n} = \left[D_{\widehat{a_n}}, J_{\widehat{a_n}}, PLR_{\widehat{a_n}}, B_{\widehat{a_n}}\right] \tag{39}$$

Dónde:

$A = conjunto\ de\ aplicaciones.$

$\hat{a} = Requerimientos\ para\ un\ desempeño\ óptimo\ de\ la\ aplicaciones.$

$D_{\widehat{a_n}} = Retardo\ requerido\ por\ la\ aplicación\ .$

$PLR_{\widehat{a_n}} = Pérdida\ de\ paquetes\ requerido\ por\ la\ aplicación.$

$B_{\widehat{a_n}} = Velocidad\ de\ transmisión\ requerida\ por\ la\ aplicación\ .$

$J_{\widehat{a_n}} = Fluctuaciones\ en\ la\ trayectoria.$

En ocasiones la ruta más corta no cumple los requerimientos de QoS de las aplicaciones, por lo cual es necesario elegir la ruta con base a los requerimientos de la aplicación, donde cada parámetro de la red tendría que cumplir las restricciones de la aplicación para elegir una ruta. Como se representa en las ecuaciones (40), (41) y (42), cada métrica tiene un límite máximo tolerable y para la tasa de transmisión un mínimo requerido por la aplicación, representado en la ecuación (43). Cada parámetro de la trayectoria en la red debe cumplir con la restricción impuesta por la aplicación.

$$D(p_n) \leq D_{\widehat{a_n}} \tag{40}$$

$$J(p_n) \leq J_{\widehat{a_n}} \tag{41}$$

$$PLR(p_n) \leq PLR_{\widehat{a_n}} \tag{42}$$

$$B_{\widehat{a_n}} \leq B(p_n) \tag{43}$$

El conjunto de parámetros de la aplicación y la red conforman las entradas del optimizador de CLD. Se realizan conjuntos de pares válidos, donde la aplicación utiliza la red de manera eficiente, representado en las ecuaciones (44) y (45).

$$Y = A \times P^{m,k} = \{\gamma_1, \gamma_2, \dots, \dots, \gamma_n\} \tag{44}$$

$$\gamma_n = (\widehat{a_n}, p_n) = (\left[D_{\widehat{a_n}}, J_{\widehat{a_n}}, PLR_{\widehat{a_n}}, B_{\widehat{a_n}}\right], [D(p_n), J(p_n), PLR(p_n), B(p_n)]) \tag{45}$$

4.4 Propuesta para la selección de la ruta con mejores condiciones desde el punto de vista de la aplicación.

Para este caso se consideran tres métricas: el retardo, la pérdida de paquetes y la tasa de transmisión. Utilizando las ecuaciones de la sección 4.3, así como las consideraciones mencionadas en 4.1 y 4.2 se obtiene la ecuación (46).

$$\gamma_n = (a_n, p_n)$$
$$a_n = [D_{\widehat{a_n}}, PLR_{\widehat{a_n}}, B_{\widehat{a_n}}] \ , \quad p_n = [D(p_n), PLR(p_n), B(p_n)]$$
$$\gamma_n = \left([D_{\widehat{a_n}}, PLR_{\widehat{a_n}}, B_{\widehat{a_n}}], [D(p_n), PLR(p_n) , B(p_n)]\right) \tag{46}$$

Existe un subconjunto de trayectorias ***(Q)*** dentro del total de las ***n*** trayectorias posibles, que cumplen las restricciones de la aplicación. Como primera instancia se elige el parámetro de la tasa de transmisión, para verificar que existe la capacidad disponible y evitar la saturación de las trayectorias entre los pares $\boldsymbol{\gamma_n}$ reduciendo el vector a $\dot{\boldsymbol{\gamma}}_{\boldsymbol{n}}$. Como segundo parámetro se tiene la restricción de retardo con esto se obtiene el subconjunto ***(Q)*** de ***m*** rutas que cumplen ambas restricciones, como se representa en la ecuación (47) y (48), así como los pares válidos se reducen de $\boldsymbol{\gamma_n}$ a $\boldsymbol{\gamma_m}$. En la Figura 27 se ilustra el proceso de selección para obtener el subconjunto ***(Q)*** con base en las restricciones de pérdida de tasa de transmisión disponible y el retardo.

$$\exists \{Q \ \in \ P^{m,k} \ | \ (\ v[D(p_n)] \leq D_{\widehat{a_n}} \ ) \cap \ (\ v[B(p_n)] \leq B_{\widehat{a_n}} \ )\} \tag{47}$$
$$Q = \{\overline{p_1}, \ \overline{p_2}, \ \ldots\ldots, \overline{p_m}\} \tag{48}$$
$$\gamma_m = \left([D_{\widehat{a_n}}, PLR_{\widehat{a_n}}, B_{\widehat{a_n}}], [D(\overline{p_m}), PLR(\overline{p_m}) , B(\overline{p_m})]\right)$$

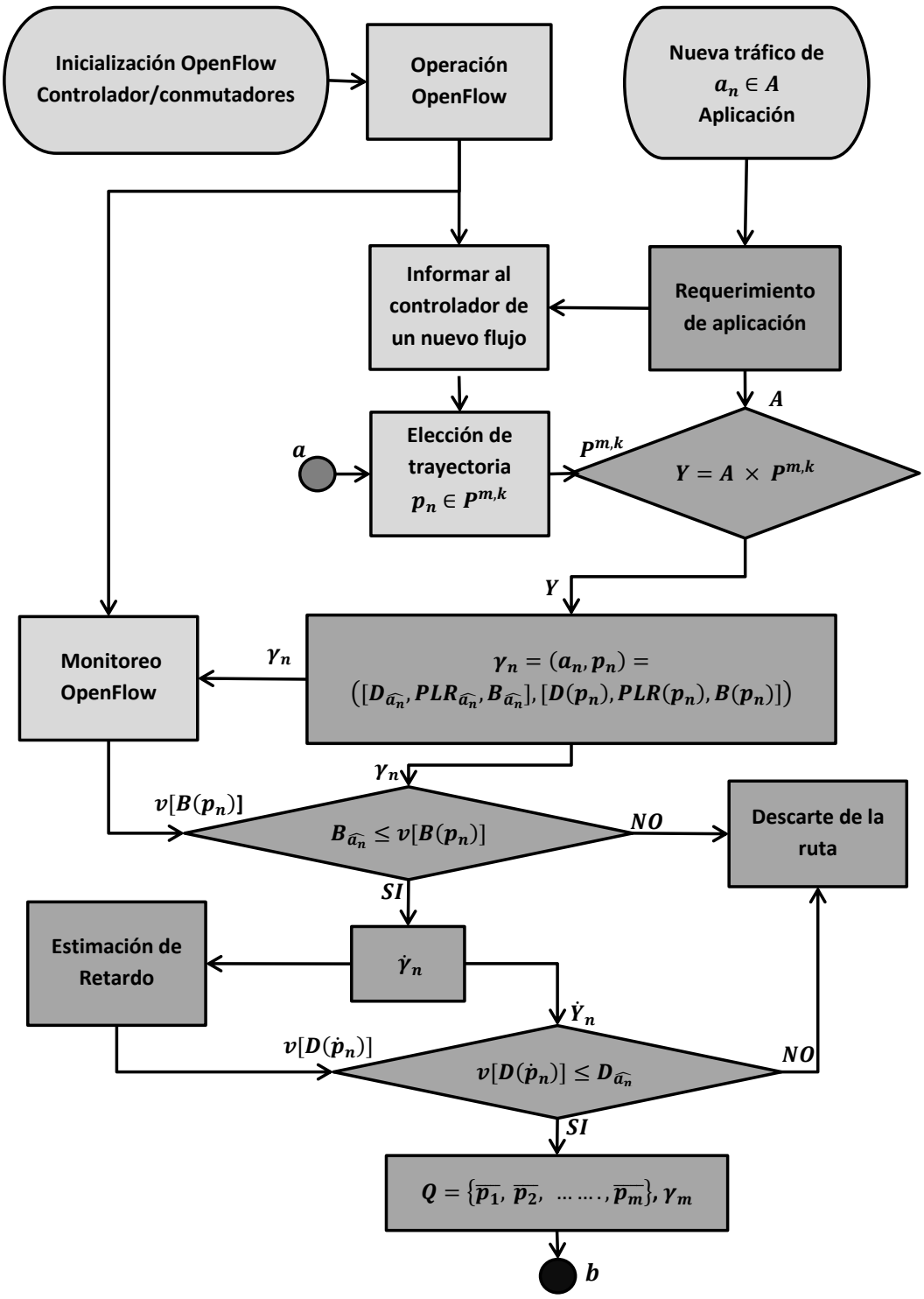

Figura 27. Diagrama de flujo para la selección de trayectoria parte 1.

Una vez evaluado el subconjunto ***(Q)***, se elige la ruta de menor retardo entre el subconjunto con base a la estimación calculada, como se representa en la ecuación (49). Esta ruta es la que cuenta con mejores condiciones en ese instante de tiempo, si se toma la decisión en otro instante de tiempo, la ruta con mejores condiciones con base al retardo podría ser diferente.

$$min(v[D(Q)]) \quad (49)$$

Por otro lado, la métrica de la pérdida de paquetes se utiliza como condicionante de la ruta, representada en la ecuación (50). Si la trayectoria elegida con el menor retardo del conjunto excede la restricción de pérdida de paquetes, la trayectoria se descarta y se elige otra trayectoria con el mismo proceso entre las rutas restantes del subconjunto ***(Q)***. Si todas las trayectorias del conjunto ***(Q)*** se descartan se reinicia por completo el proceso de elección de la ruta o en su defecto se deniega el acceso a la aplicación ya que no se cuentan con los recursos suficientes en la red para dar el desempeño que se solicita. En la Figura 28 se describe el diagrama de flujo para la selección de la trayectoria con base en el retardo y con restricción de la pérdida de paquetes.

$$\hat{p} \quad | \quad (PLR(\overline{p_m}) \leq PLR_{\widehat{a_n}}) \quad (50)$$

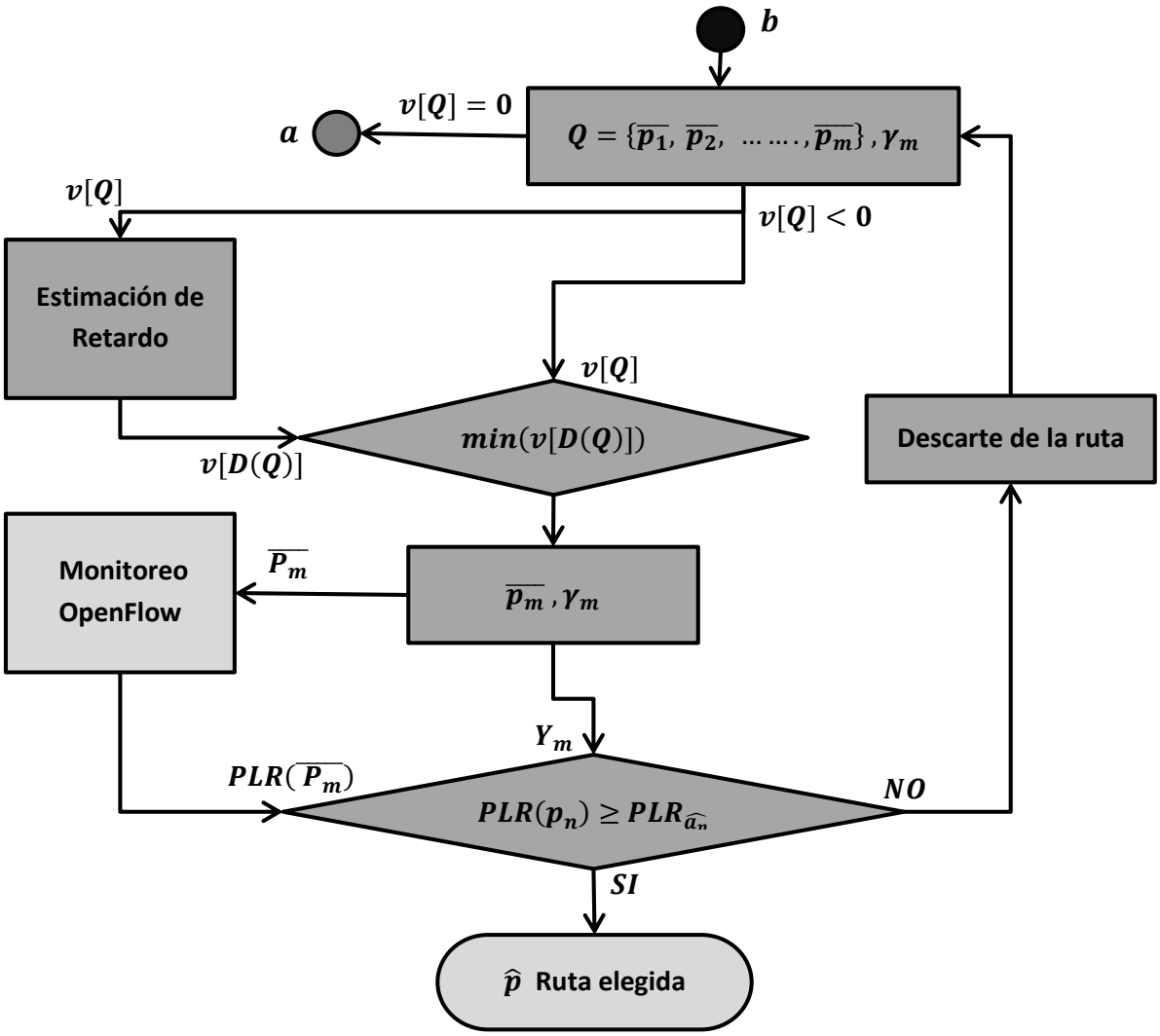

Figura 28. Diagrama de flujo para la selección de trayectoria parte 2.

La selección de la trayectoria es un proceso de Poisson y se puede modelar con un sistema M/M/1/k, donde la entrada es exponencial con media $1/\lambda$, la salida es exponencial con media $1/\mu$, se cuenta con un solo recurso para atender las peticiones, la capacidad del sistema está limitada por un número máximo de usuarios (o flujos) en el sistema, las variables aleatorias exponenciales no tienen memoria y tienen una alta correlación entre ellas. La función de probabilidad exponencial para las variables aleatorias está dada por la ecuación (51) y la función de probabilidad de Poisson del proceso está dada por la ecuación (52) (Serrano & Hern, 2016).

$$f(x) = \lambda e^{-\lambda x} \tag{51}$$

$$f(n) = \frac{\lambda^n e^{-\lambda}}{n!} \tag{52}$$

La capacidad del sistema está limitada por la métrica de tasa de transmisión de la trayectoria entre los requisitos de las aplicaciones, con redondeo hacia abajo, donde la restricción ***k*** en el sistema está dada por la ecuación (53).

$$k = \left\lfloor \frac{B(p_n)}{B_{\widehat{a_n}}} \right\rfloor \tag{53}$$

La probabilidad de bloqueo en cada estado estacionario (Þ) es equiprobable, está dada por la ecuación (54), está en función de ***k*** va a depender de la capacidad disponible y el requerimiento.

$$\text{Þ} = \frac{1}{k+1} \tag{54}$$

Cada parámetro considerado tiene un objetivo particular, la tasa de transmisión para limitar los recursos disponibles y evitar la saturación de los enlaces. La pérdida de paquetes nos ayuda a mantener el mayor caudal eficaz posible, además la selección de ruta con base en el retardo permite mantener un flujo de información rápido y ágil.

La correlación entre las variables aleatorias exponenciales permite predecir el comportamiento entre ellas, por ejemplo al mantener el retardo menor a 100ms en una trayectoria se estima que la trayectoria puede tener un valor pérdida de paquetes por debajo de 10^{-3}, como se muestra en la Figura 29 que representa la ecuación (51) de la función de probabilidad de las variables exponenciales para la tasa de perdida de paquetes y el retardo.

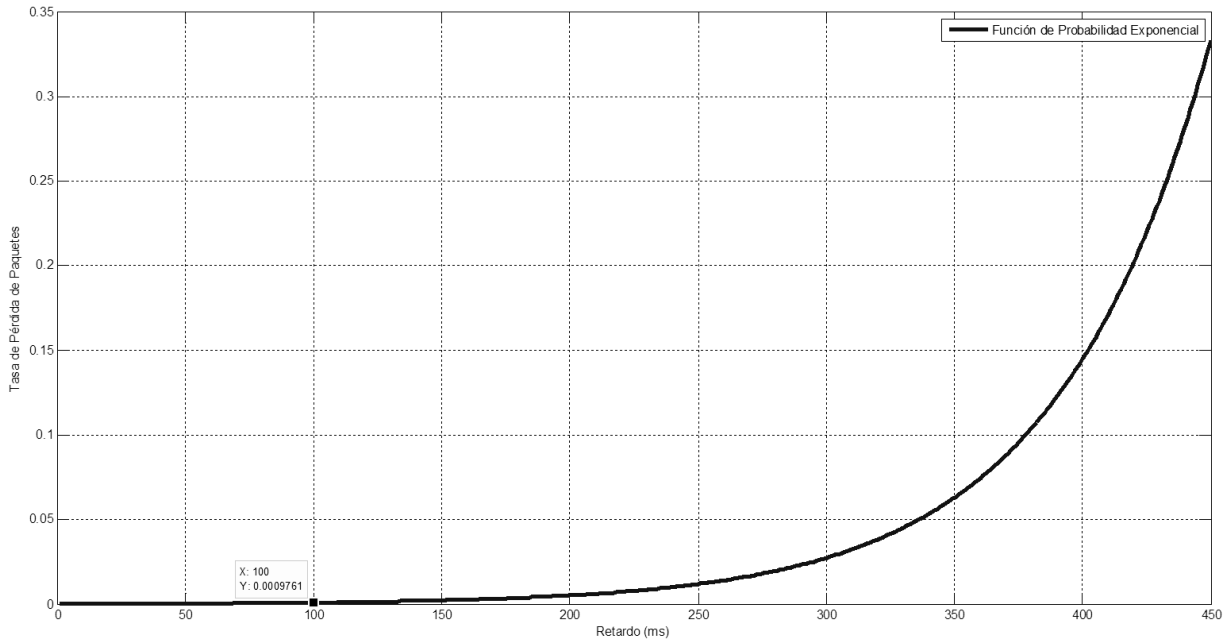

Figura 29. Desempeño predecible con la función de probabilidad exponencial para retardos y pérdidas de paquetes.

Además al mantener un valor pérdida de paquetes por debajo de 10^{-3}, se maximiza el rendimiento de la aplicación al mantener el caudal eficaz (Thoughput) en sus mejores condiciones, a mayores pérdidas el caudal eficaz disminuye drásticamente. Como se muestra en la Figura 30 que representa la ecuación (51) de la función de probabilidad de las variables exponenciales para la tasa de perdida de paquetes y el caudal eficaz en escala logarítmica.

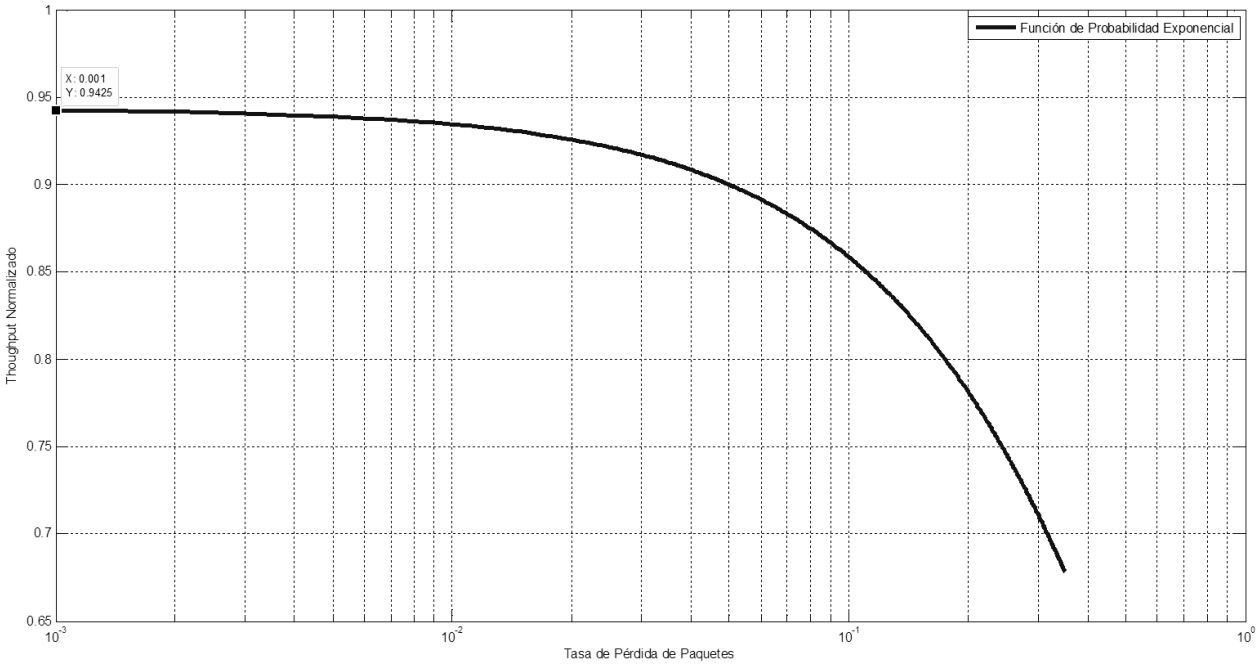

Figura 30. Desempeño predecible con la función de probabilidad exponencial para pérdidas y caudal eficaz.

4.5 Emulación

De las plataformas de red para realizar emulación de una SDN como se aborda en el capítulo 2. Mininet es la plataforma más común y utilizada como referencia en diversos artículos para trabajar SDN con OpenFlow por la estabilidad que ofrece.

De los controladores disponibles en su gran mayoría cuentan con soporte OpenFlow, entre los más utilizados son NOX y Floodlight; la ventaja de Floodlight sobre NOX es la gran documentación que existe y el apoyo entre la comunidad que lo utiliza. Es comúnmente utilizado por la estabilidad que ofrece y se usa como un controlador de referencia.

Para la emulación se utilizó una máquina virtual sobre vSphere de VMware, con sistema operativo Ubuntu 14.04 LTS, 11.9GB de memoria RAM y Procesador Intel Xeon, donde se encuentra instalado Mininet (simulador SDN) y Floodlight (Controlador SDN).

Las versiones de software utilizado son:

- Ubuntu 14.04.3 LTS Trusty kernel 3.19.0-25-generic
- Mininet 2.2.1 (Plano de Datos)
- Floodlight 1.2 (Plano de Control, utiliza REST API como interfaz del Norte)
- Openvswitch 2.0.2
- OpenFlow 1.3 (Interfaz del Sur)
- Matlab 2014
- Iperf (Plano de Aplicación)

Se definió la estructura del sistema de emulación, las entradas y salidas esperadas. El emulador es un ambiente controlado pero introduce variables, de las cuales no se tienen control y agregan un margen de error, por lo tanto son consideradas como se ilustra en la Figura 31. Estas variables incontrolables se deben a la estructura del mismo emulador y la interacción entre Mininet, el sistema operativo y el equipo físico donde se encuentran (hardware).

4.5.1 Topología

La topología nos permite considerar diferentes posibilidades de rutas y condiciones de la red, por esto, al igual que OpenQos (H.E. Egilmez & Dane, 2012) se utiliza una topología de bucle piramidal como se muestra en la Figura 32, que permite estresar el algoritmo de enrutamiento para diferentes casos.

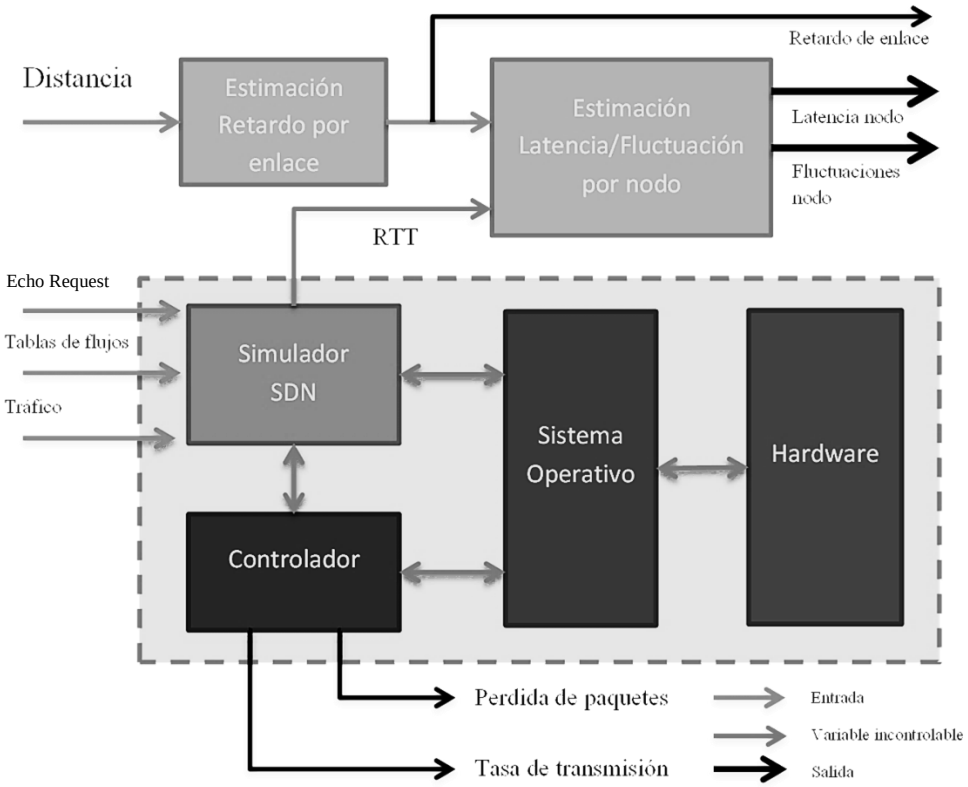

Figura 31. Estructura del sistema de emulación.

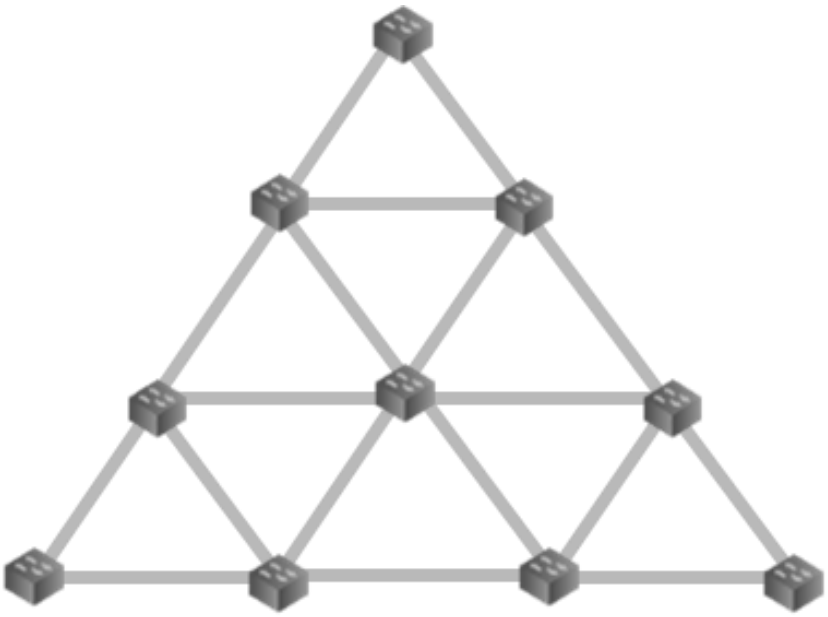

Figura 32. Topologías bucles en forma piramidal.

La topología de red se considera como un ambiente distribuido, y forma parte de la nube. La comunicación se realiza a través de la red de extremo a extremo entre dos dispositivos por una aplicación en común, como se ilustra en la Figura 33.

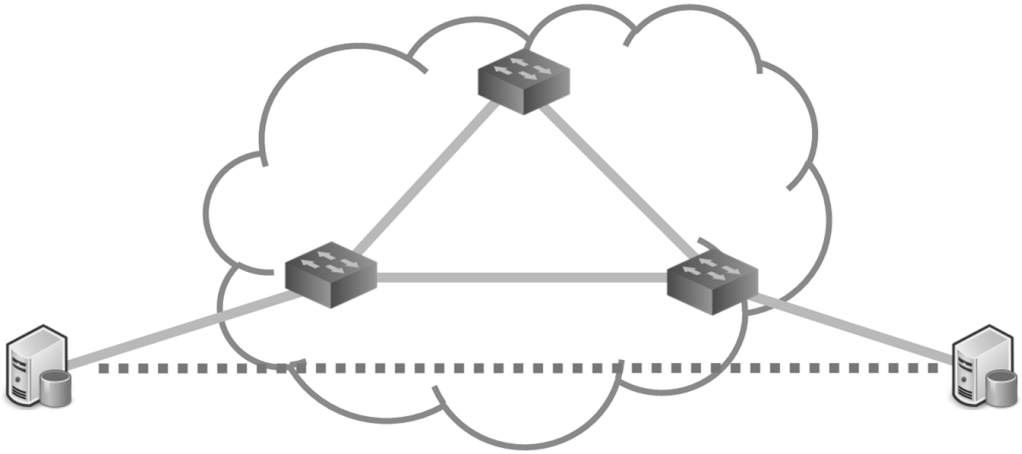

Figura 33. Enlace extremo a extremo a través de la nube.

El controlador elige las trayectorias de los flujos con base en su algoritmo de enrutamiento de la ruta más corta. Las tablas de enrutamiento se ingresan manualmente para definir una ruta fija hacia cada nodo. Las tablas fijas se utilizan para definir los caminos para llegar a los conmutadores, en la Figura 34 se describe los caminos para alcanzar cada nodo.

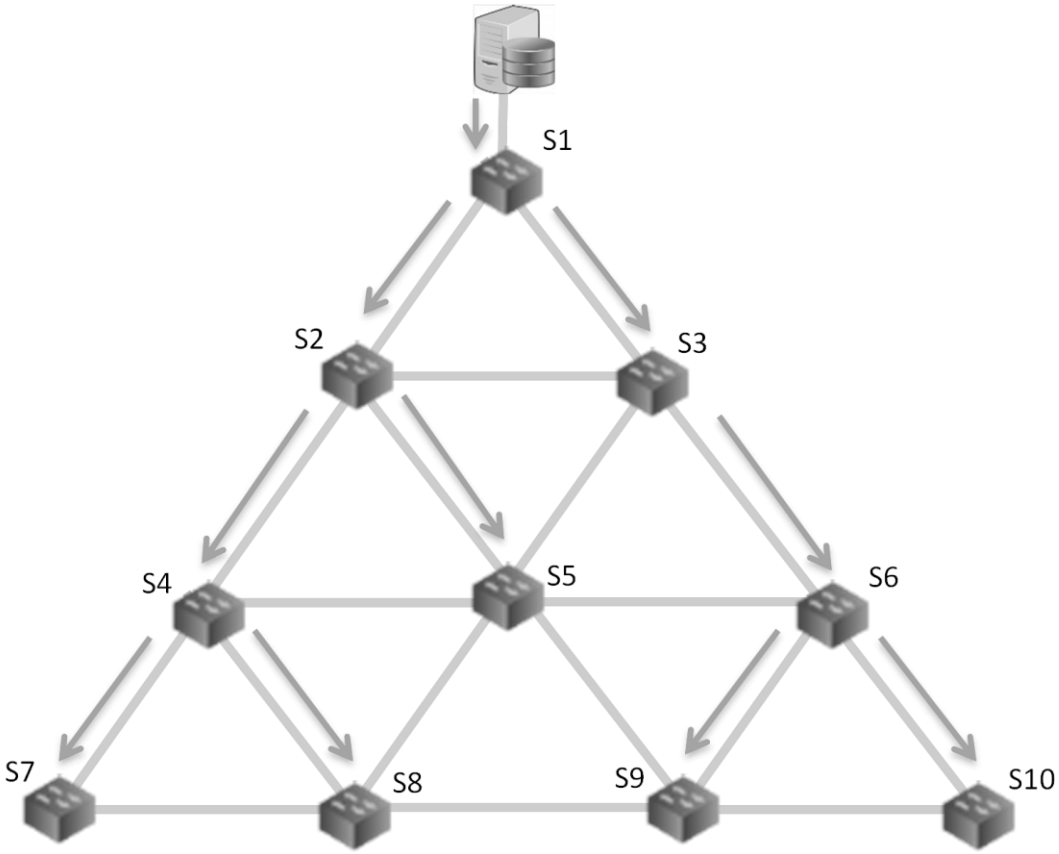

Figura 34. Trayectorias definidas para alcanzar cada conmutador.

La configuración de los parámetros en Mininet, se utiliza para establecer las condiciones iniciales de tráfico base en la topología, por ejemplo OpenQos (H.E. Egilmez & Dane, 2012) utiliza la configuración de cada enlace suponiendo un tráfico moderado-bajo, estos parámetros definen el comportamiento de cada enlace en la red de las capacidades disponibles en ese momento. Los parámetros configurados en cada enlace son retardo, pérdida de paquetes, fluctuación del retardo y tasa de transmisión.

Mininet permite asignar valores predeterminados de retardos, fluctuaciones y capacidad de transmisión por enlace, además de pérdida de paquetes por nodo, se utiliza la configuración de la topología en la Figura 35. Se considera el mismo valor de distancia para cada enlace de 10 km (10000 metros), esto permite mantener una estabilidad y homogeneidad en la topología, así como reducir los factores adicionales que afecten el cálculo de estimación.

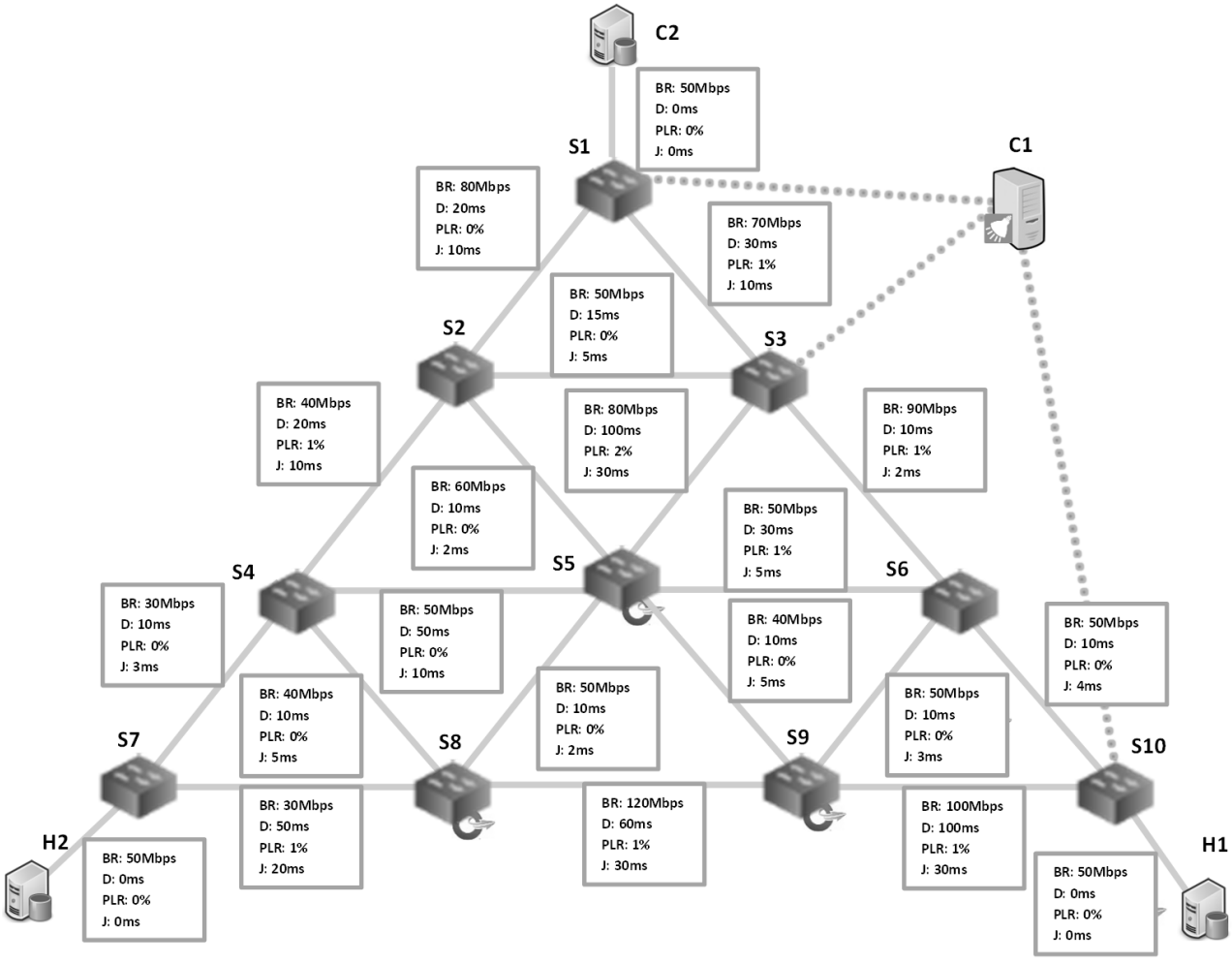

Figura 35. Configuración de la topología.

La comunicación entre el controlador y el conmutador se realiza mediante un canal seguro por cifrado asimétrico basado en TLS (por sus siglas en inglés, Transport Layer Security) (Göransson & Black, 2014). En un escenario real, el controlador se encuentra conectado directamente a algún nodo y a través de este nodo se comunica hacia los demás nodos como se muestra en la Figura 36 inciso A. En un esquema de simulación, el controlador se conecta directamente a cada nodo por medio de enlaces, en muestra en la Figura 36b, al igual que en la Figura 22.

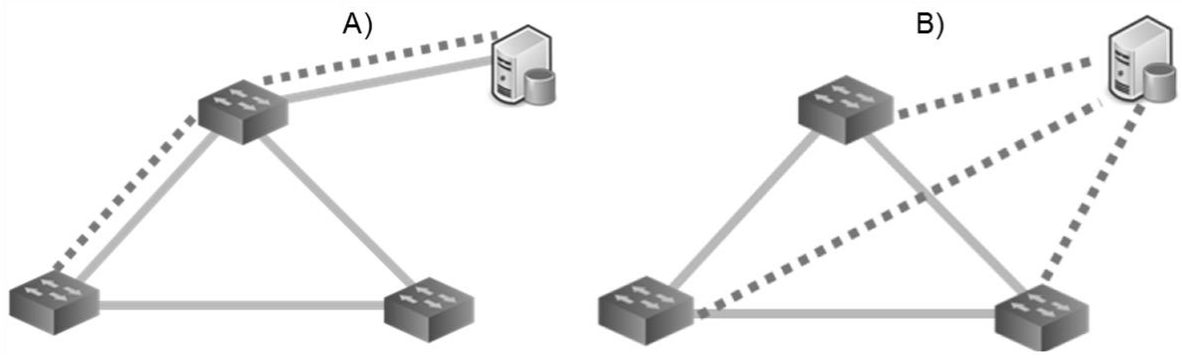

Figura 36. Comunicación Controlador-conmutador A) Topología en un escenario real B) Topología emulada.

4.5.2 Aplicación

El tráfico de la nube en (Buh, T., Trobec, R., & Ciglič, 2014) es caracterizado por las longitudes de paquetes, donde las de mayor uso se encuentran entre los 1,280 y 1,500 bytes y entre 64 y 83 bytes de longitud de información, como se muestra en la Tabla 8.

Tabla 8. Porcentajes del tráfico sobre la nube (Buh, T., Trobec, R., & Ciglič, 2014).

Distribución de tipos de protocolos en el backbone de Internet	
Tipo de Protocolo	Porcentaje de tráfico
TCP	75
UDP	10
ICMP	6
TCP6	6
UDP6	0.5
ICMP6	0.2
OTROS	2.3

Distribución de tamaño de paquetes en el backbone de Internet	
Longitud de paquete (Bytes)	Porcentaje de tráfico
64-83	36
84-159	7
160-319	2
320-639	3
640-1279	3
1278-1500	49

El tráfico de una trasmisión de datos en grandes volúmenes de información como las aplicaciones de alto desempeño (HPC) y Big Data, por lo general, utilizan longitudes máximas de paquetes. Una trama

Ethernet (Figura 37) de tamaño máximo consta de 1518 bytes (1500 de datos y 18 de campos) sin incluir el preámbulo. Un Echo Request/Reply estándar se realiza con una longitud de 32 bytes, sin embargo puede ser modificado. Para obtener un valor de RTT similar a una transmisión de datos de HPC, las solicitudes de eco se realizan con tramas de 1518 bytes.

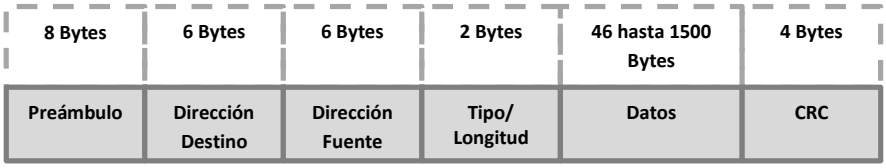

Figura 37. Trama Ethernet (Spurgeon, 2000).

Para generar tráfico similar de una aplicación de alto desempeño y medir el rendimiento de la aplicación por las diferentes rutas posibles, se considera una aplicación HPC sensible al retardo y a las fluctuaciones del retardo. El tráfico en la nube se caracteriza por ser en su mayoría tráfico TCP, aunque el uso del protocolo UDP es menor, es el protocolo usado para aplicaciones interactivas o de procesamiento en tiempo real, que tienen mayor sensibilidad al retardo. Con estas consideraciones para la configuración del tráfico de aplicaciones de alto desempeño se utiliza el protocolo UDP, con una trama Ethernet de 1518 bytes sin incluir el preámbulo.

En la literatura para aplicaciones de alto desempeño solo se menciona los requerimientos generales como bajo retardo, alta disponibilidad de la tasa de transmisión, bajas pérdidas de paquetes, etc., sin embargo no se mencionan los requerimientos específicos para cada parámetro. Los requerimientos de una aplicación de alto desempeño interactiva, sensible al retardo y a las fluctuaciones del retardo, se tomaron a partir de las Clases de calidad de servicio que se muestran en la Tabla 9 (Marchese, 2007), en la cual se observa que las características de las aplicaciones HPC a simular coinciden en la Clase cero o uno de QoS, las cuales se definen como un retardo menor a 100 ms, fluctuaciones ±50 ms y una pérdida de paquetes menor a 1x 10-3.

Tabla 9. Clases de calidad de servicio (Marchese, 2007).

Clase QoS	Características	Retardo	Fluctuaciones	Perdidas de paquetes
0	Tiempo real, sensibles a fluctuaciones, altamente interactivo	100ms	50 ms	1 x 10-3
1	Tiempo real, sensibles a fluctuaciones, interactivo	400 ms	50 ms	1 x 10-3
2	Transacciones de datos, altamente interactivo	100 ms	N/A	1 x 10-3
3	Transacciones de datos, interactivo	400 ms	N/A	1 x 10-3
4	Baja perdida (Transacciones cortas, lotes de datos, transmisión de video)	1 s	N/A	1 x 10-3
5	Aplicaciones tradicional de redes IP por defecto	N/A	N/A	N/A

A la par de las aplicaciones de HPC existen aplicaciones de tele-inmersión que también requieren características de calidad de servicio similares, como son altas tasas de transmisión (Gbps), retardos y fluctuaciones del retardo bajos, así como la garantía de entrega. Para dichas aplicaciones se definen parámetros específicos, por ejemplo, el retardo para cada tipo de tráfico, las cuales se muestran en la Tabla 10. Por ejemplo la aplicación de renderización[20] de imágenes en HPC, se caracteriza por utilizar un flujo constante de información, o pueden ser multi-flujos hacia diferentes destinos para dividir las tareas.

Tabla 10. Requerimientos de retardo para aplicaciones interactivas (Berliner, Clark, & Hartono, 2011).

Tipo de Aplicación	Retardo
Control	< 30 ms
Texto	< 100 ms
Audio	< 30 ms
Video	< 100 ms
Rastreo	< 10 ms
Bases de datos	< 100 ms
Simulación	< 30 ms
Háptico	< 10 ms
Renderizado	< 30 ms

Con las consideraciones de las aplicaciones sensibles al retardo como lo es el renderizado en tele-inmersión podemos definir que una aplicación de renderizado en HPC tiene la misma exigencia, requiere que el retardo sea menor a 30 ms, así como fluctuaciones del retardo de ±50 ms y una pérdida de paquetes menor a $1x\ 10^{-3}$, además de la máxima velocidad de transmisión disponible.

[20] Término latino-español para referirse al término en inglés rendering, aunque no existe una traducción como tal para este término.

Capítulo 5. Emulación y Resultados

En el capítulo 4 se describieron de manera detallada, el diagrama del algoritmo propuesto (Figura 27 y 28), la topología (Figura 32) y la aplicación (Tabla 10) que se utiliza para el experimento, además de algunas consideraciones que se realizaron para este caso. En este capítulo se muestran los resultados experimentales del algoritmo de estimación, la técnica para el diseño de capas cruzadas propuesta y el mecanismo simple de enrutamiento que utiliza las métricas de QoS.

5.1 Consideraciones de emulación

Para establecer las condiciones iniciales del comportamiento de la red, se considera un caso con 10 nodos, a los cuales se realizó un procedimiento de solicitud de eco hacia cada nodo. Esto conforma una corrida que se considera como un instante de tiempo de las condiciones de la red. Cada solicitud de eco se realizó con el tráfico base configurado en la topología de la Figura 35.

Es posible calcular la media o proporción de una muestra, y conocer así cuál es la probabilidad de que el universo bajo análisis tenga ese mismo valor o comportamiento similar. El valor a calcular en la muestra será el más probable para el universo o caso estudiado. En total se realizaron alrededor de 1500 corridas como muestra inicial, con un intervalo de respuesta del 92.12% de Echo Request/Reply realizados y un 7.88% de Echo Request/Reply sin respuesta correcta. El 5.85% pertenecen a Echo Request/Reply que no tienen respuesta debido a las pérdidas de paquetes configurados en la topología y el 2.02% de picos de latencia inducidos por las variables que agrega el sistema. Como se observa en la Figura 31 el esquema cuenta con variables incontrolables por parte del mismo sistema que generan latencias adicionales. Esto se considera como parte del margen de error con el que se cuenta para el cálculo de estimación del retardo emulación.

Para cada corrida de Echo Request/Reply se realizaron los cálculos de estimación de latencias, con esto se obtiene un instante de tiempo de las condiciones de la red. Con base en estos datos se pueden calcular los valores de retardo para las ***m*** rutas posibles entre el destino y la fuente de una transmisión extremo a extremo.

El objetivo principal es asignar la trayectoria que cumpla con los requerimientos de la aplicación HPC, por ejemplo, el renderizado de imágenes requiere ciertas características de la red para enviar su información

de un cliente al servidor donde se procesara la información, que como ya se mencionó en la sección 4.5.2 están definidas por: un retardo menor a 30 ms, fluctuaciones del retardo menor a 50 ms y una pérdida de paquetes menor a $1x10^{-3}$, además de una tasa máxima de transmisión de 30 Mbps de limite en la topología para todas las ***n*** trayectorias entre el cliente (H2) y el servidor (H1) de la Figura 35. A la aplicación se asigna como requisito utilizar múltiplos de 10Mbps para escalonar el tráfico y simular tráfico bajo (1/3 de la capacidad del enlace), moderado (2/3 de la capacidad del enlace), alto (en el límite, 3/3 de la capacidad del enlace) y saturado (4/3 de la capacidad del enlace).

Las fluctuaciones del retardo al igual que los demás métricas tienen una gran importancia en el comportamiento de la red y el rendimiento de las aplicaciones. Al ser un ambiente controlado, durante la emulación este parámetro no presenta variaciones que violen la restricción que requiere la aplicación, aun y cuando en la topología se configuran valores mayores, como se ilustra en la Figura 35. Aunque se realiza la estimación de las fluctuaciones del retardo, este parámetro no se consideró como parte de la toma de decisiones como se observa en el diagrama de la Figura 27 y 28.

Como parte de la emulación se obtiene el subconjunto ***(Q)*** de ***m*** trayectorias, que cumplan la restricción de retardo y tasa de transmisión disponible, como se muestra en de la ecuación (55). Se sigue el proceso descrito en la sección 4.4 y la secuencia de la Figura 27 para obtener el subconjunto ***(Q)***, donde el número de trayectorias disponibles es ***m*** igual a 8 como se muestra en la ecuación (56), por lo tanto se cuenta con 8 trayectorias que cumplen ambas restricciones de retardo y tasa de transmisión.

$$\{Q \in P^{m,k} \mid \left(D(p_n) \leq D_{\widehat{a_n}} \right) \cap \left(B(p_n) \leq B_{\widehat{a_n}} \right)\} \quad (55)$$

$$Q = \{\overline{p_1}, \overline{p_2}, \overline{p_3}, \overline{p_4}, \overline{p_5}, \overline{p_6}, \overline{p_7}, \overline{p_8}\} \quad (56)$$

5.2 Resultados

El siguiente paso es elegir la ruta entre el conjunto ***(Q)*** de ***m*** posibilidades; de las 8 trayectorias se selecciona la de valor de retardo menor, como se muestra en la ecuación (57) y la Figura 28.

$$min(v[D(Q)]) \quad (57)$$

La probabilidad de cada ruta de ser la que tenga menor valor de retardo de las ***m*** posibles trayectorias, corresponde a un proceso de Poisson donde la distribución de las probabilidades es dada por la ecuación (52). Con base en la muestra inicial de 1500 corridas se toma un primer caso, donde se realiza con el

tráfico base y bajo las mismas condiciones iniciales en la red para cada ciclo, se elige la ruta con menor retardo. Como se esperaba, la ocurrencia en la selección de trayectorias con base en el retardo tiene su media en la primera ruta (la ruta más corta) y el comportamiento de la ocurrencia sigue una distribución de la función de probabilidad de Poisson. La probabilidad de ocurrencia para cada trayectoria en la selección de ruta se muestra en la Figura 38 en forma de barras, esta se compara con el comportamiento esperado de la función de probabilidad de Poisson representado en una línea continua.

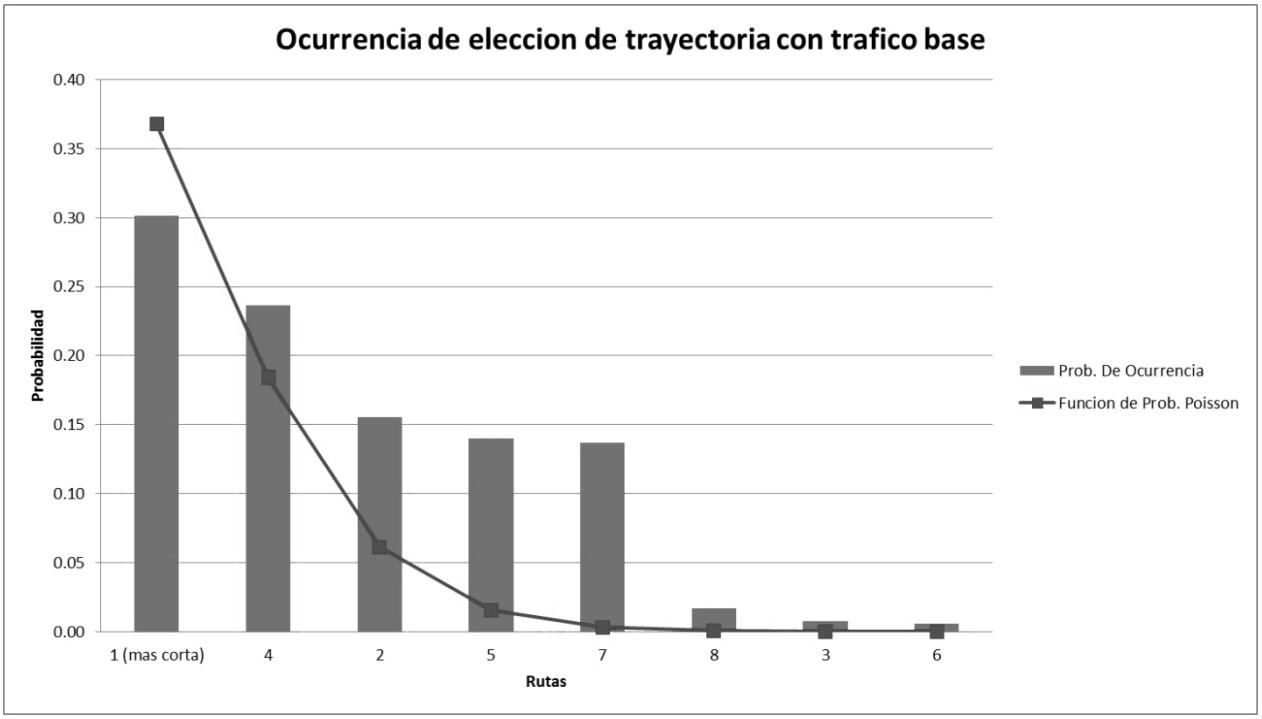

Figura 38. Ocurrencia de elección de trayectoria con tráfico base.

El siguiente paso fue añadir tráfico de una aplicación como HPC, para saturar la ruta más corta de manera escalonada y observar el cambio en las decisiones que se toman para cada caso de tráfico. Esta prueba se realizó con un muestreo de alrededor de 250 corridas de solicitudes de eco.

La ocurrencia en la selección de la trayectoria con menor retardo con tráfico bajo y moderado, como se muestra en las Figuras 39 y 40 respectivamente, sigue la distribución de la función de probabilidad de Poisson. Al igual que en la Figura 38 la distribución tiene la media en una primera ruta. No se encuentra una diferencia significativa en la distribución de la ocurrencia conforme se aumenta el tráfico en estas condiciones, la distribución de probabilidad para los casos de tráfico bajo y moderado se mantiene sin cambios considerables. Lo anterior obedece a que la ruta más corta sigue sin presentar estado de congestión y el retardo requerido por la aplicación se mantiene en condiciones aceptables. En la Figura

40 se presentan los resultados para el caso en donde el tráfico cae en la clasificación de moderado, representando una gran utilización de la ruta sin llegar a la congestión.

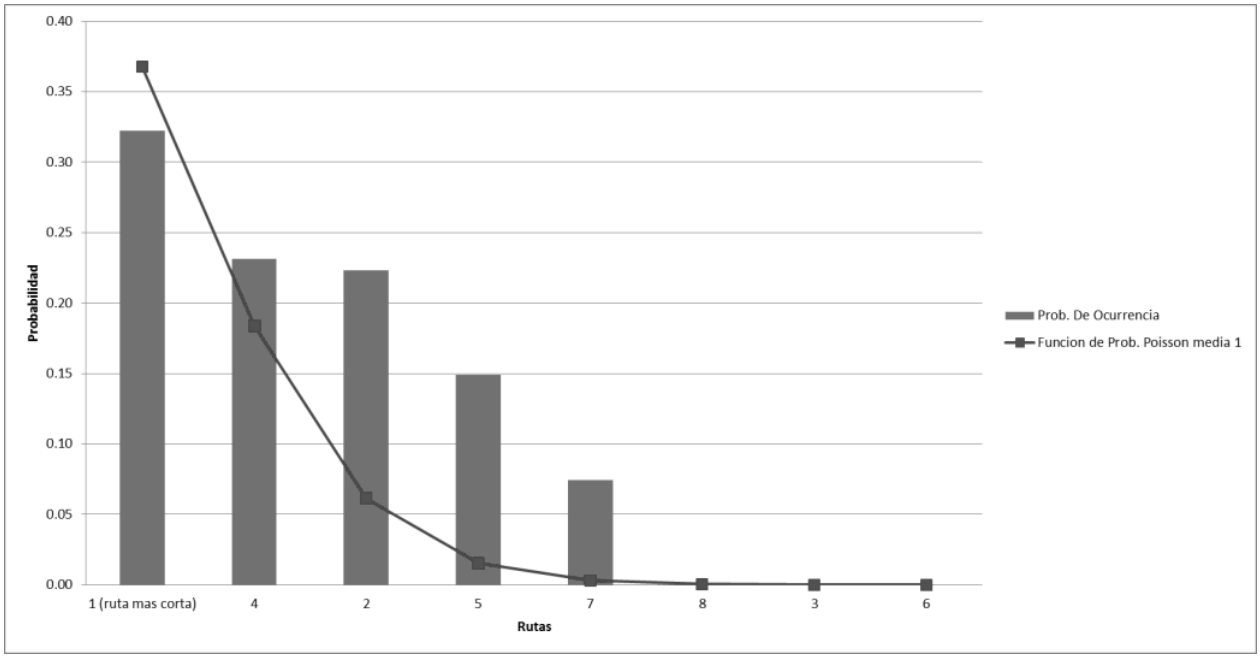

Figura 39. Ocurrencia de elección de trayectoria con tráfico bajo en la ruta más corta.

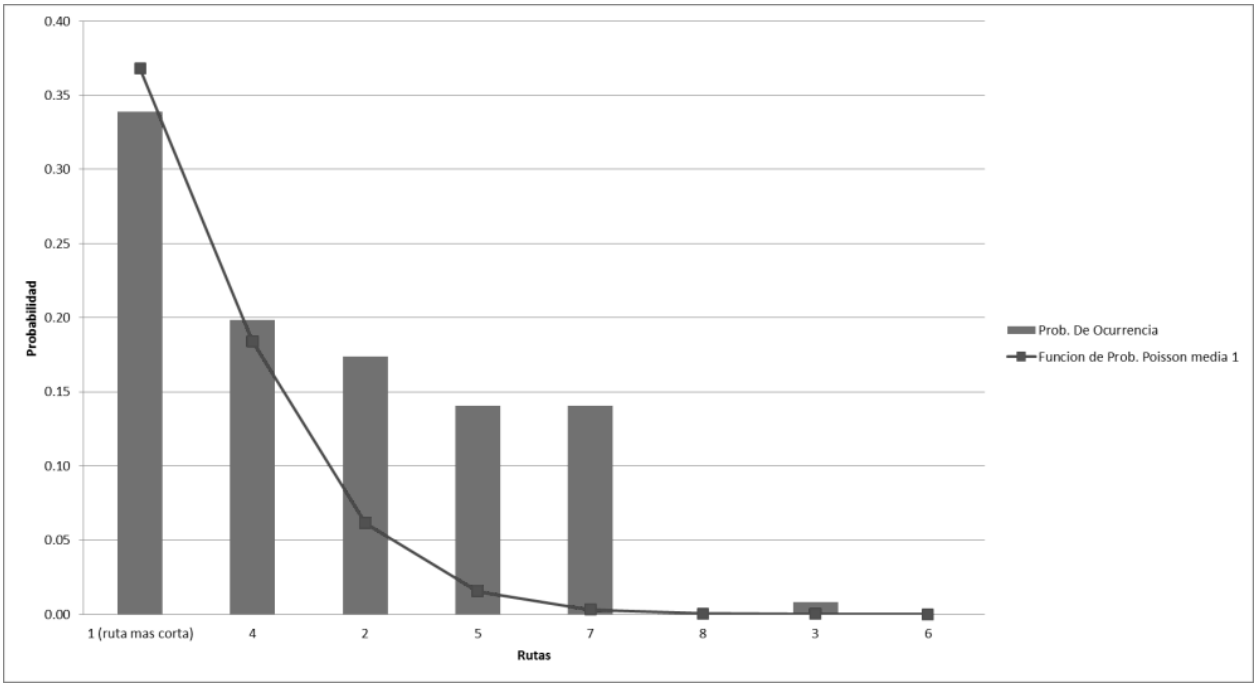

Figura 40. Ocurrencia de elección de trayectoria con tráfico moderado en la ruta más corta.

Hasta aquí, se han realizado pruebas de cómo el algoritmo de decisión de la ruta más corta sigue siendo la mejor opción para el intercambio de tráfico, siempre y cuando la ruta cumpla con las condiciones de los requerimientos de retado.

En el algoritmo de una SDN, la probabilidad de ocurrencia de la ruta más corta es de “uno” para los casos de tráfico bajo, moderado, alto y saturado. Sin importar las condiciones de la red o si no hay recursos suficientes para atender un nuevo flujo, siempre se elige la ruta más corta por omisión y como resultado este algoritmo tiende a saturar las trayectorias. Por otro lado, comparado con la selección de la ruta en base al retardo, redujo la probabilidad de ocurrencia de la trayectoria más corta en la selección de la ruta; con condiciones de tráfico bajo se reduce a un 0.32, con tráfico moderado cambia a un 0.34, con tráfico alto a un 0.21 y con tráfico saturado a un 0.27 de probabilidad de ocurrencia de selección, ya que no siempre ésta trayectoria va a cumplir las restricciones de retardo.

Debido a que el comportamiento tiene una distribución de Poisson, en condiciones de tráfico alto y saturado sobre la ruta más corta, la media se desplaza hacia una segunda ruta. Por lo tanto, disminuye la probabilidad de ocurrencia de la ruta más corta a alrededor de un 0.25 y la probabilidad de bloqueo del sistema al redistribuir el tráfico entre las demás rutas. La ocurrencia de selección de trayectoria con menor retardo en condiciones de tráfico alto (límite del enlace) se muestra en la Figura 41 y en saturación del enlace en las Figuras 42.

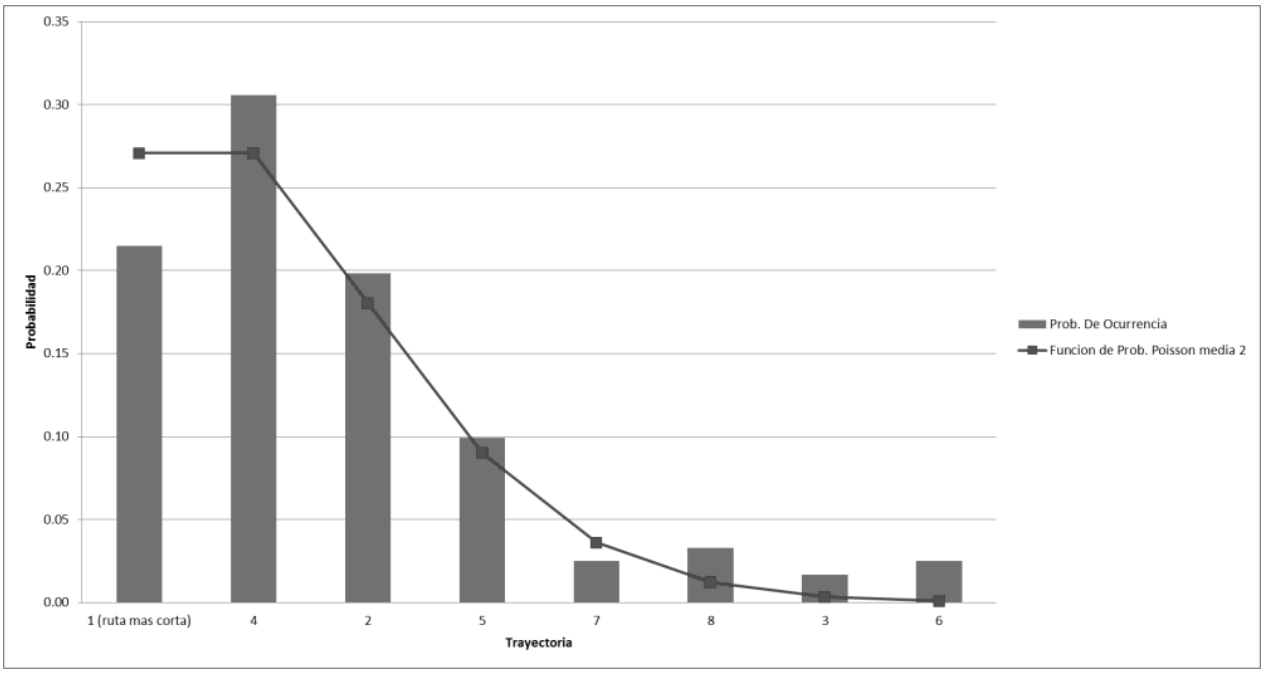

Figura 41. Ocurrencia de elección de trayectoria con tráfico alto (límite del enlace) en la ruta más corta.

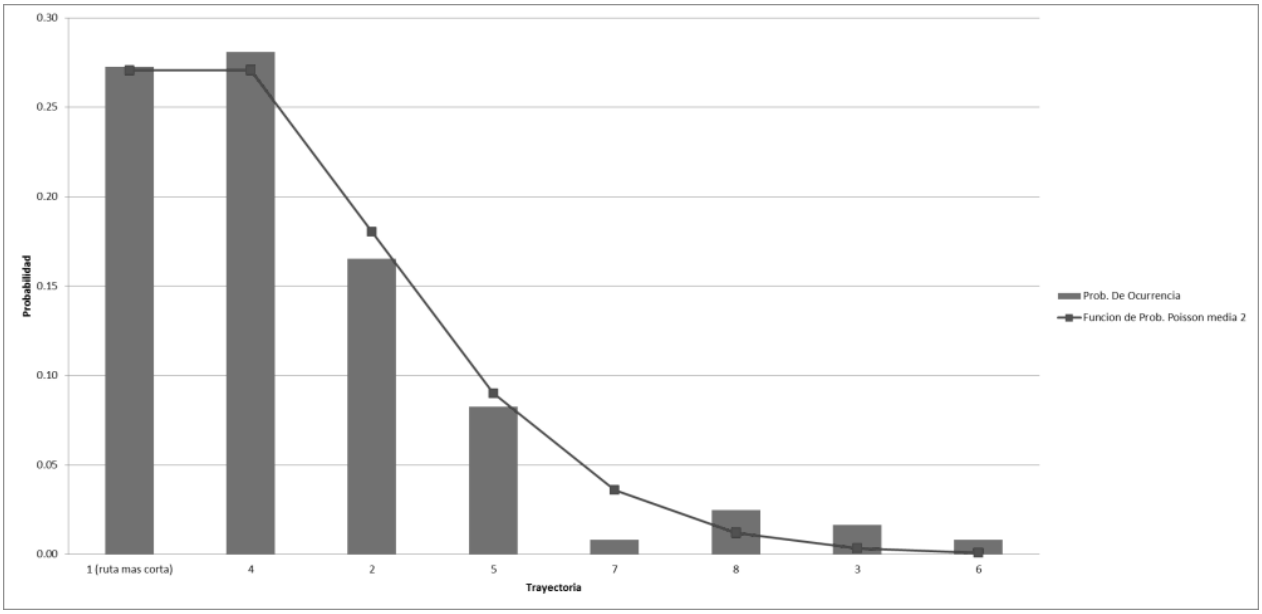

Figura 42. Ocurrencia de elección de trayectoria con saturación en la ruta más corta.

Estadísticamente el comportamiento para ambos casos de tráfico alto y saturado es correcto ya que siguen la distribución esperada, de las Figuras 41 y 42. Sin embargo en la práctica para estos casos, los enlaces ya no cuentan con recursos para asignar y el mismo recurso se distribuye entre una mayor cantidad de usuarios, por lo tanto degradan el servicio en el flujo de la información. Para el caso donde la trayectoria se encuentre al límite de su capacidad (tráfico alto) o congestionado (tráfico saturado) no se debe considerar esa ruta, ya que no se cuentan con los recursos para dar servicio a nuevo flujos de información.

5.2.1 Evaluación del algoritmo propuesto

El objetivo principal del algoritmo propuesto es que el controlador pueda elegir la ruta con mejores características en el momento que lo requiera, por medio del monitoreo constante y al evitar las rutas saturadas. Este algoritmo propuesto permite llegar a la selección de la trayectoria con mejores características. El proceso de selección de trayectoria de la Figura 27 y 28, ilustra el diagrama de flujo que describe el algoritmo presentado en este documento en el apartado 4.4. El algoritmo propuesto se utiliza para elegir la ruta para el caso de tráfico alto (Figura 41), donde cambia la distribución debido a la restricción de tasa de transmisión (ecuación 47), la distribución resultante se ilustra en la Figura 43. Para esta nueva distribución ya no se consideran las rutas que no tienen recursos disponibles, además el caso de saturación (Figura 42) debería evitarse por parte del algoritmo propuesto. Las probabilidades se re-distribuyen conforme a la función Poisson, la media regresa a una primer ruta entre las disponibles. El

conjunto de las trayectorias válidas se ilustran en la Figura 43 en donde se descartan las trayectorias en condiciones de tráfico alto (límite del enlace).

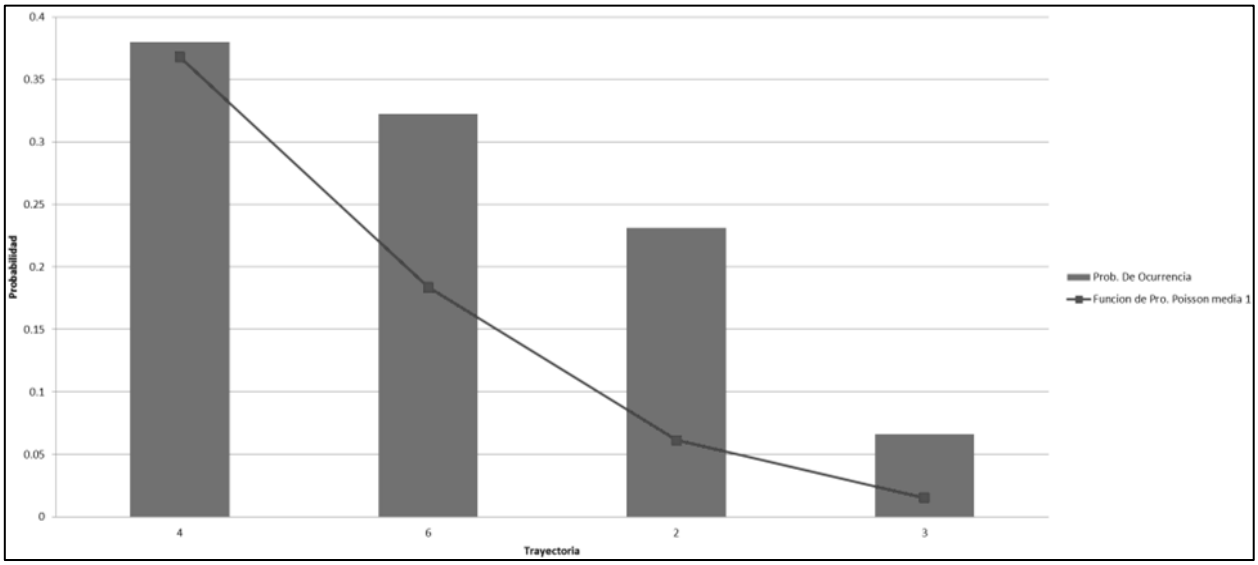

Figura 43. Ocurrencia de elección de trayectoria con tráfico alto en la ruta más corta con el algoritmo propuesto.

Como se ha comprobado la selección de la misma ruta da como resultado el caso de saturación, el cual debe evitarse. Para este trabajo de tesis se considera el parámetro de tasa de transmisión como condicionante en la ecuación (47) para evitar el estado de saturación en la selección de la ruta por asignar a una aplicación de HPC. De las ***m*** posibles rutas, cualquiera puede ser elegida en algún momento basada en el retardo y que una ruta tenga la mayor probabilidad de ser seleccionada no garantiza que tenga las condiciones requeridas. Para mantener un rendimiento óptimo el mayor tiempo posible es necesario que se determinen las condiciones de las rutas periódicamente para poder realizar el cambio de ruta cuando sea necesario. En el Anexo B se encuentra un análisis de rendimiento para diferentes trayectorias del subconjunto ***(Q)***, así como una breve comparación del retardo real con el retardo estimado.

5.2.1.1 Selección de trayectoria con el algoritmo propuesto para tráfico bajo

Para el escenario de este caso de estudio, se tiene una de tasa máxima de 30Mbps para las ***n*** trayectorias entre el cliente (H2) y el servidor (H1) de la Figura 35. Así como las restricciones de pérdida de paquetes y retardo definidas para la aplicación en el apartado 4.5.2. La aplicación envía la solicitud para utilizar la red, junto con sus requerimientos de QoS, el controlador recibe la petición y abstrae las condiciones de la red utilizando el algoritmo propuesto, en la primer etapa del algoritmo (Figura 27) se obtiene el

conjunto ***(Q)*** de trayectorias que cumple las restricciones de retardo y tasa de transmisión al mismo tiempo, donde el valor de ***m*** es igual a 8 rutas. La distribución inicial de probabilidades en la selección corresponde a la Figura 38.

En la segunda etapa del algoritmo (Figura 28) del conjunto ***(Q)***: (1) se elige la ruta con menor retardo, (2) de la ruta elegida se monitorean las pérdidas de paquetes, (3) se descarta la ruta si excede la restricción de pérdidas de paquetes, (4) el conjunto ***(Q)*** se reduce y el valor de ***m*** cambia, (5) se repite el proceso, se elige la ruta con menor retardo del conjunto ***(Q)*** entre las rutas restantes para cada ciclo. Este proceso de selección de la ruta con base en el retardo para cada ciclo para el caso de tráfico bajo se ilustra en la Figura 44.

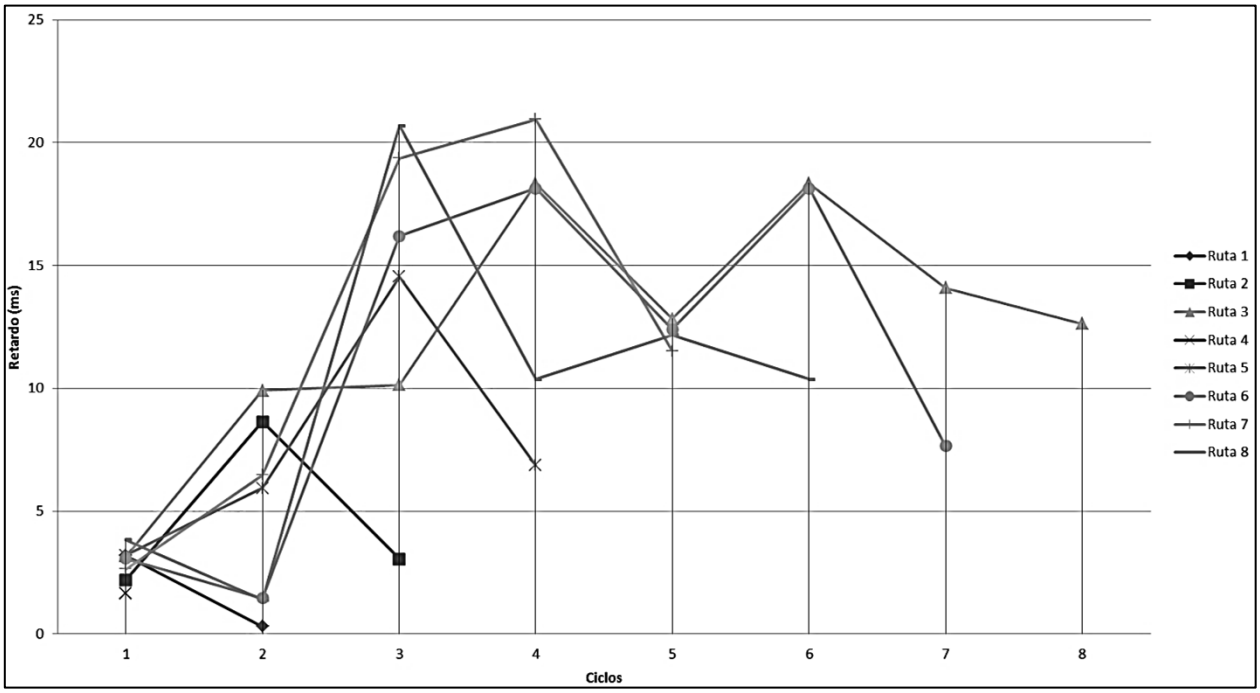

Figura 44. Selección de trayectoria con base en el menor retardo en condiciones de trafico bajo.

Con el proceso del algoritmo (Figura 28) se obtiene que en el primer ciclo la trayectoria con menor retardo sea la ruta 4 y se monitorea las pérdidas de paquetes, donde se tiene 1% de pérdidas. La ruta excede la restricción y es descartada del conjunto ***(Q)***, se reduce el valor de ***m*** a 7 rutas posibles y se selecciona una nueva trayectoria. En el segundo ciclo se elige la ruta 1 (la más corta) y se repite el proceso para cada ciclo, hasta descartar las rutas que excedan la restricción de pérdidas. En el ciclo 8 solo queda disponible la ruta 3 y se selecciona, el monitoreo de las pérdidas sobre la ruta cumple la restricción con $1x10^{-3}$ y se elige como trayectoria para la transmisión de la aplicación. En la Figura 45 se

ilustra el proceso de selección de trayectoria, con base en el retardo en condiciones de tráfico bajo sobre la ruta más corta y el descarte de rutas con base en la pérdida de paquetes.

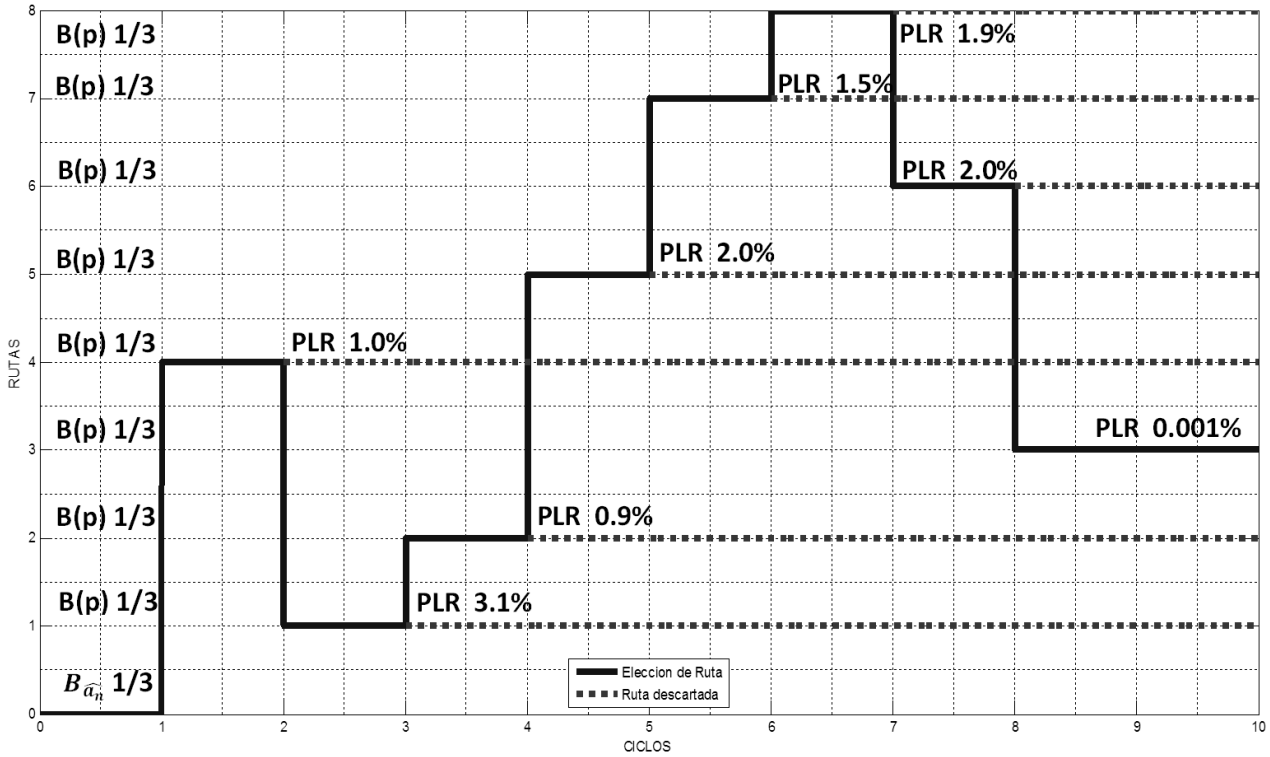

Figura 45. Proceso de selección de trayectoria en ciclos con trafico bajo.

En cada ciclo donde se descarta una trayectoria el valor de ***m*** del conjunto ***(Q)*** disminuye, esto provoca que la probabilidad de ocurrencia se re-distribuya entre las ***m*** rutas para cada ciclo. Manteniéndose la distribución de la probabilidad con la media en una primera ruta que puede cambiar conforme se descarte las rutas del conjunto ***(Q)***.

5.2.1.2 Selección de trayectoria con el algoritmo propuesto para tráfico moderado con negación de servicio

Para el caso de tráfico moderado, el proceso de selección se repite (Figura 28) y el comportamiento es muy similar al caso de tráfico bajo. La distribución inicial de probabilidades en la selección corresponde a la Figura 39. Para ejemplificar el proceso con un caso diferente se muestran las condiciones donde ninguna trayectoria cumple el requerimiento de pérdidas, todas las rutas del conjunto ***(Q)*** son descartadas y se niega el acceso para trasmitir ya que no se cuentan con los recursos necesarios para otorgar el servicio. En la Figura 46 se ilustra el proceso de selección de trayectoria con base en el retardo en condiciones de tráfico moderado sobre la ruta más corta para el caso de negación de servicio.

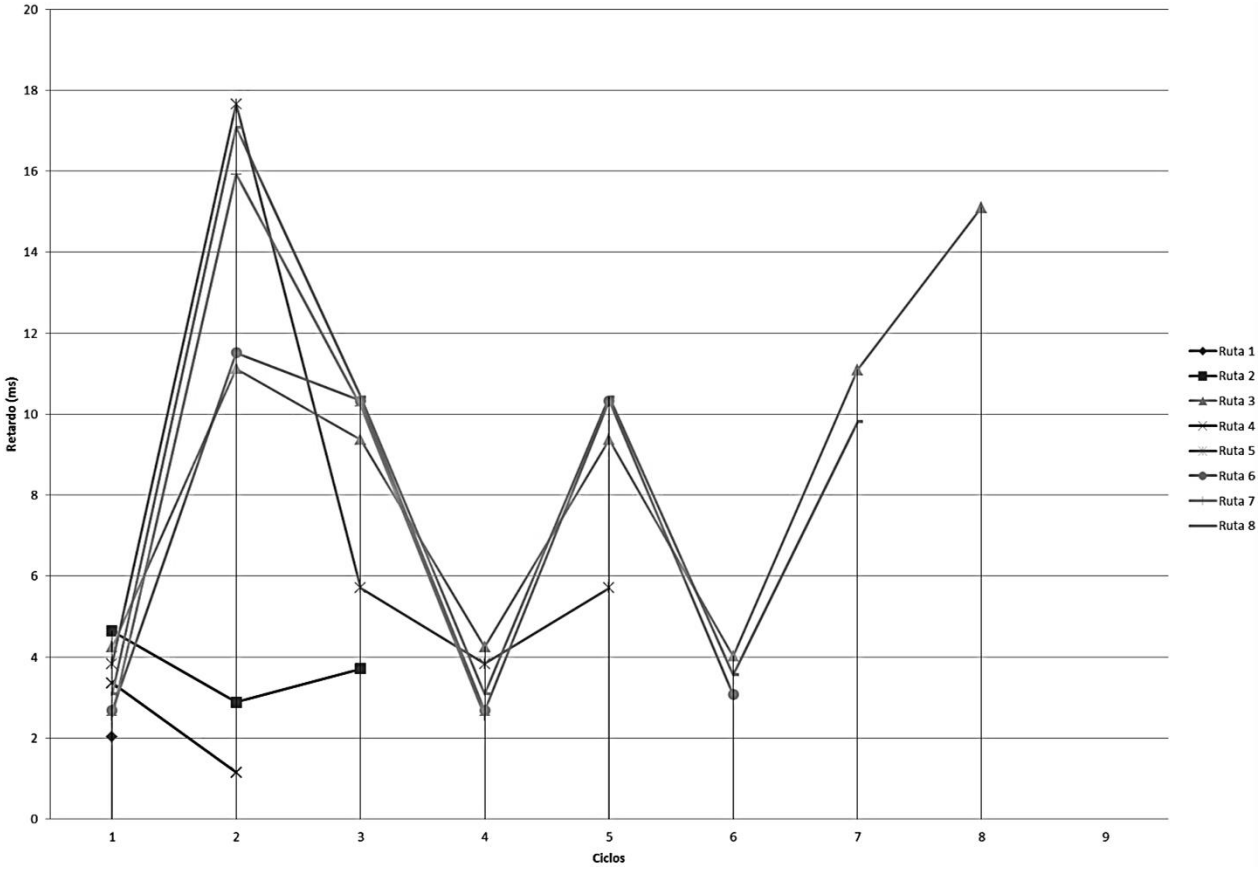

Figura 46. Selección de trayectoria con base en el menor retardo en condiciones de tráfico moderado.

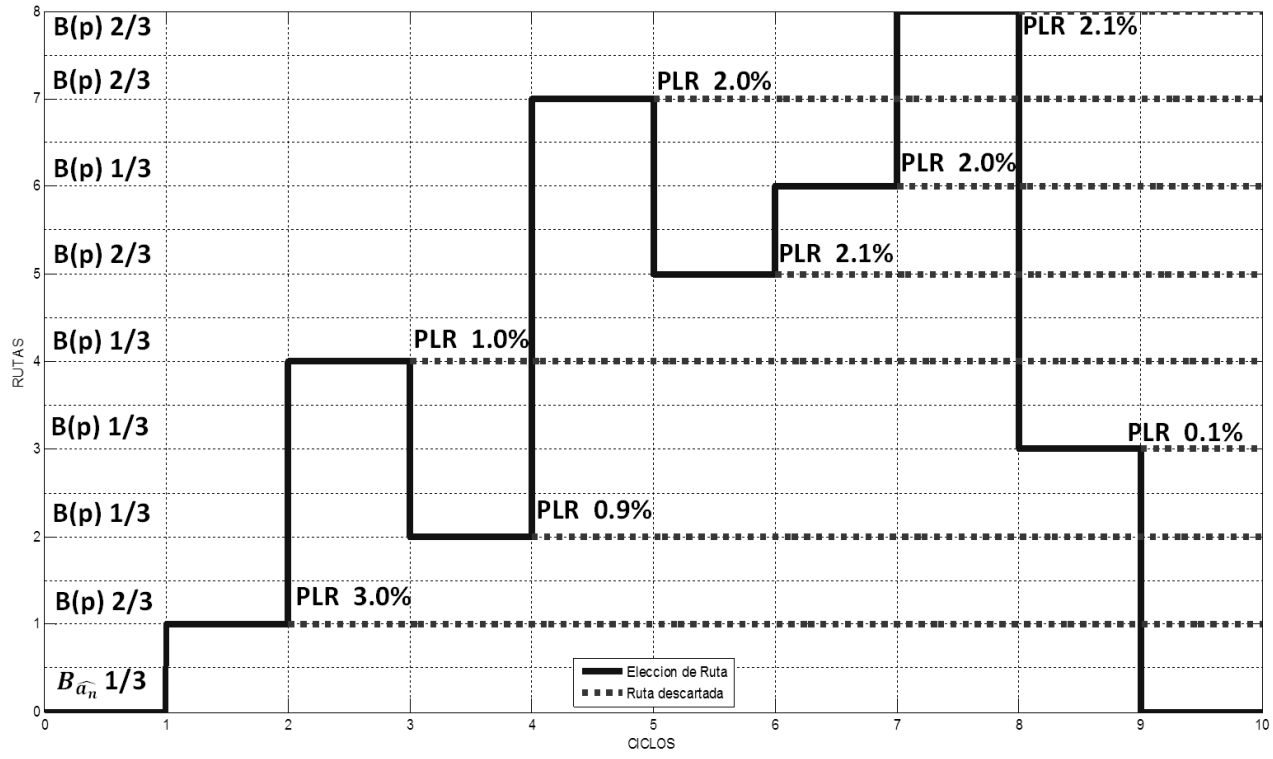

Figura 47. Proceso de selección de trayectoria en ciclos con tráfico moderado para el caso de negación de servicio.

Al igual que el caso de tráfico bajo, del conjunto ***(Q)*** se elige la ruta con menor retardo para cada ciclo, se elige la ruta y se monitorea las pérdidas de paquetes de la misma, se descartan las rutas que excedan la restricción. Como en este caso todas las rutas infringen la restricción, se niega el servicio para la transmisión de una aplicación HPC. En la Figura 47 se ilustra el proceso de selección de trayectoria con base en el retardo en condiciones de tráfico moderado y descarte de rutas con base en la pérdida de paquetes para el caso de negación de servicio.

5.2.1.3 Selección de trayectoria con el algoritmo propuesto para tráfico alto

Para el caso donde se tiene tráfico alto el proceso de selección (Figura 27) se repite, el cual comienza a descartar las rutas no disponibles por la restricción de la tasa de transmisión y el número de rutas posibles de inicio, con una ***m*** inicial igual a cuatro (***m***=4) para el conjunto ***(Q)***. La distribución inicial de probabilidades en la selección corresponde a la Figura 43. Esta reducción en el conjunto, disminuye el tiempo y el proceso de selección de rutas para llegar a la trayectoria que cumpla los requerimientos. En la Figura 48 se ilustra el proceso de selección de trayectoria con base en el retardo en condiciones de tráfico alto sobre la ruta más corta. En la Figura 49 se ilustra el proceso de selección de trayectoria con base en el retardo en condiciones de tráfico alto y el descarte de rutas con base en la pérdida de paquetes, donde la restricción por congestión cambia las condiciones iniciales.

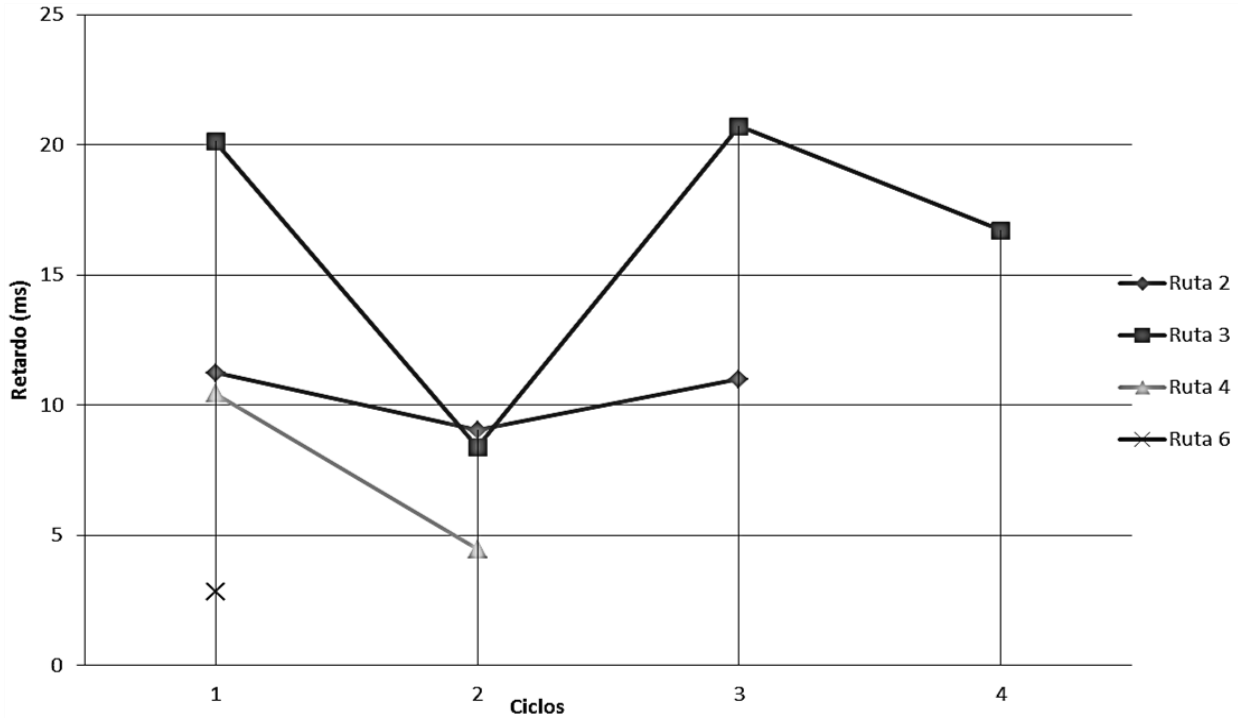

Figura 48. Selección de trayectoria con base en el menor retardo en condiciones de tráfico moderado.

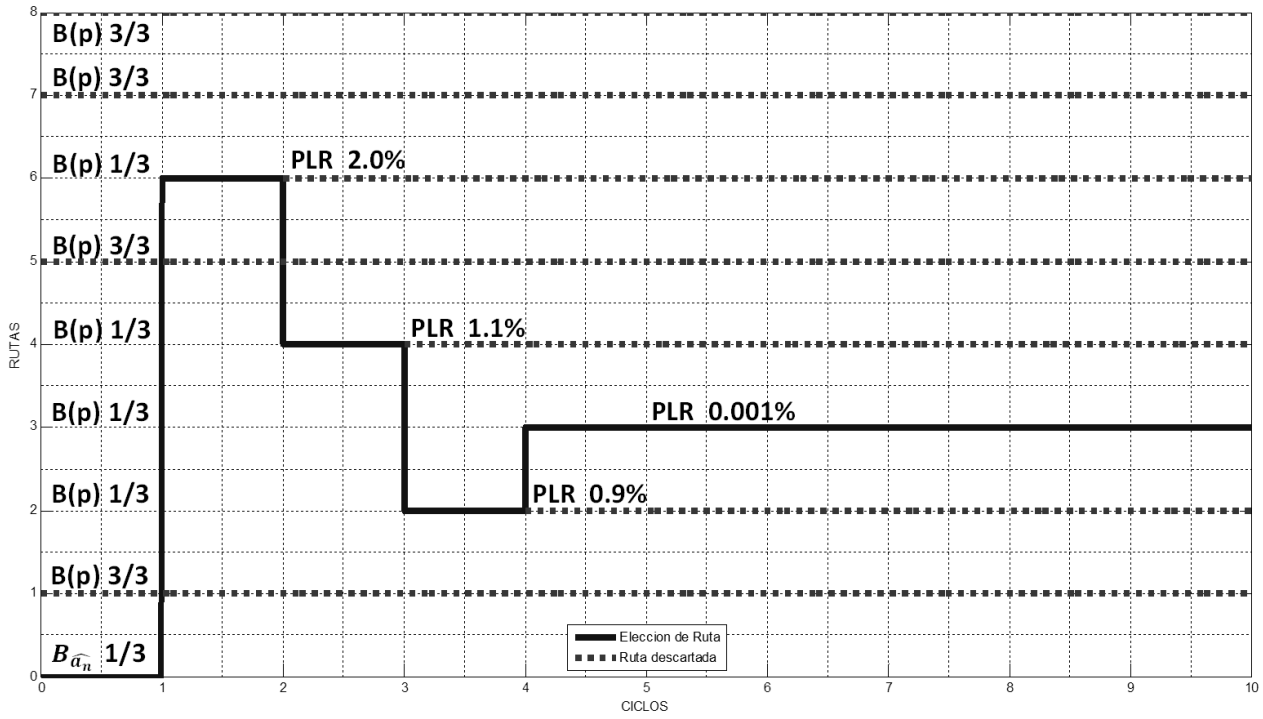

Figura 49. Proceso de selección de trayectoria en ciclos con tráfico alto sobre la ruta más corta.

5.2.2 Evaluación del desempeño de la aplicación con el algoritmo propuesto

Al realizarse la selección de trayectoria con el algoritmo propuesto, el tráfico de la aplicación sobre la ruta seleccionada (trayectoria alterna, Ruta 3) comparado contra la ruta más corta (trayectoria por defecto, Ruta 1). Se realizan pruebas de transferencia de información para observar las diferencias del rendimiento y la mejora que conlleva la selección de trayectoria en base a las métricas de QoS.

5.2.2.1 Aplicación de HPC sobre UDP

La prueba inicial se realiza con una aplicación HPC sobre el protocolo UDP donde se envía la misma cantidad de información por la trayectoria elegida por el algoritmo de las SDN y el algoritmo propuesto (954 Mbyte de información). Sobre la ruta alterna (Ruta 3, selección por algoritmo propuesto) se recibe el total de la información enviada prácticamente sin pérdidas, a diferencia de la ruta más corta (Ruta 1, selección por algoritmo de SDN) se recibe menos información debido a la mayor tasa de pérdidas que tiene esta ruta. Se pierde alrededor de 29 Mbyte, que es el 3% del total de información enviada al igual que el porcentaje de las pérdidas de paquetes. Ambas transmisiones utilizan mismo periodo de tiempo y la degradación en la transmisión es con base en el PLR. En la Tabla 11 se puede observar la diferencia en desempeño para la transmisión de datos utilizando UDP por la ruta más corta comparada con la trayectoria elegida por el algoritmo propuesto.

Tabla 11. Transmisión de datos con UDP por diferentes rutas.

Ruta	Protocolo	Tiempo de Tarea (segundos)	Información Enviada (Mbyte)	Información Recibida (Mbyte)	Tasa Promedio (Mbit/seg)	PLR Promedio (%)
1	UDP	285.7	954	925	27.2	3
3	UDP	285.7	954	954	28	0.01

En la Figura 50 se puede observar la información que se ha recibido de forma acumulada para cada segundo sobre UDP, donde ambos casos dura el mismo periodo de tiempo y se envía la misma cantidad de información. Pero a consecuencia de las pérdidas de los paquetes la cantidad de información recibida en la ruta 1, es menor que la que se envía. Serían necesarias las retransmisiones para poder obtener los paquetes perdidos.

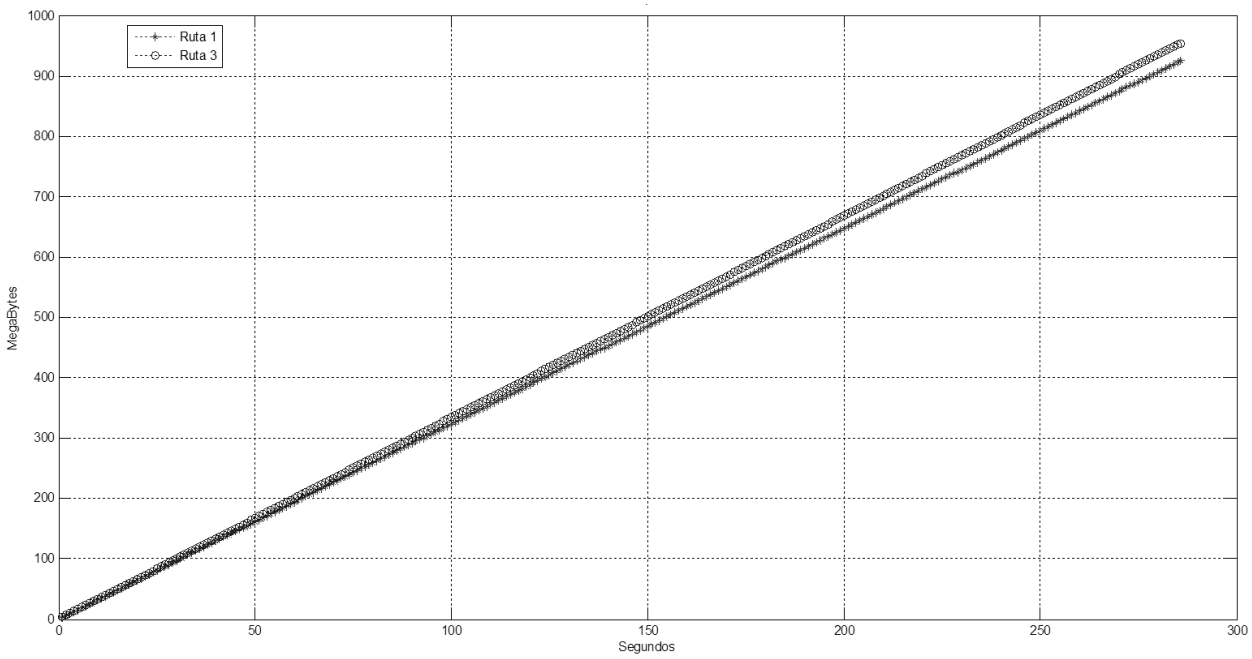

Figura 50. Recepción de información sobre UDP de forma acumulada en el tiempo como comparación de la ruta más corta y la trayectoria seleccionada por el algoritmo propuesto.

5.2.2.2 Aplicación de HPC sobre TCP

Una segunda prueba se realiza con el mismo procedimiento, sin embargo se utiliza el protocolo TCP, a diferencia de UDP se asegura que todos los paquetes lleguen, se realiza retransmisiones de los paquetes perdidos. La alta tasa de pérdidas de la ruta más corta incrementa el número de retransmisiones. Se reduce en 33.2 segundos el tiempo de tarea que significa una mejora del 10.74%. Por lo tanto, a

diferencia del algoritmo de la ruta más corta seleccionada por defecto, una misma tarea utiliza menos tiempo en realizarse a través de la trayectoria alterna elegida por el algoritmo propuesto. En la Tabla 12 se puede observar la diferencia en desempeño para la transmisión de datos utilizando TCP por la ruta más corta comparada con la trayectoria elegida por el algoritmo propuesto.

Tabla 12. Transmisión de datos con TCP por diferentes rutas.

Ruta	Protocolo	Tiempo de Tarea (segundos)	Información Enviada (Mbyte)	Información Recibida (Mbyte)	Tasa Promedio (Mbit/seg)	PLR Promedio (%)
1	TCP	309	954	954	25.9	3
3	TCP	275.8	954	954	29	0.01

En la Figura 51 se ilustra la información recibida de forma acumulada en cada segundo sobre TCP, en ambos casos se envía la misma cantidad de información. Sin embargo al igual que en el caso de la transmisión con UDP (sección 5.2.2.1) como consecuencia de las pérdidas de los paquetes, el tiempo de tarea para recibir la información es mayor sobre la ruta más corta (ruta 1) debido a las retransmisiones de los paquetes perdidos. La misma tarea, se puede realizar en menor tiempo al elegir una ruta alterna con mejores condiciones.

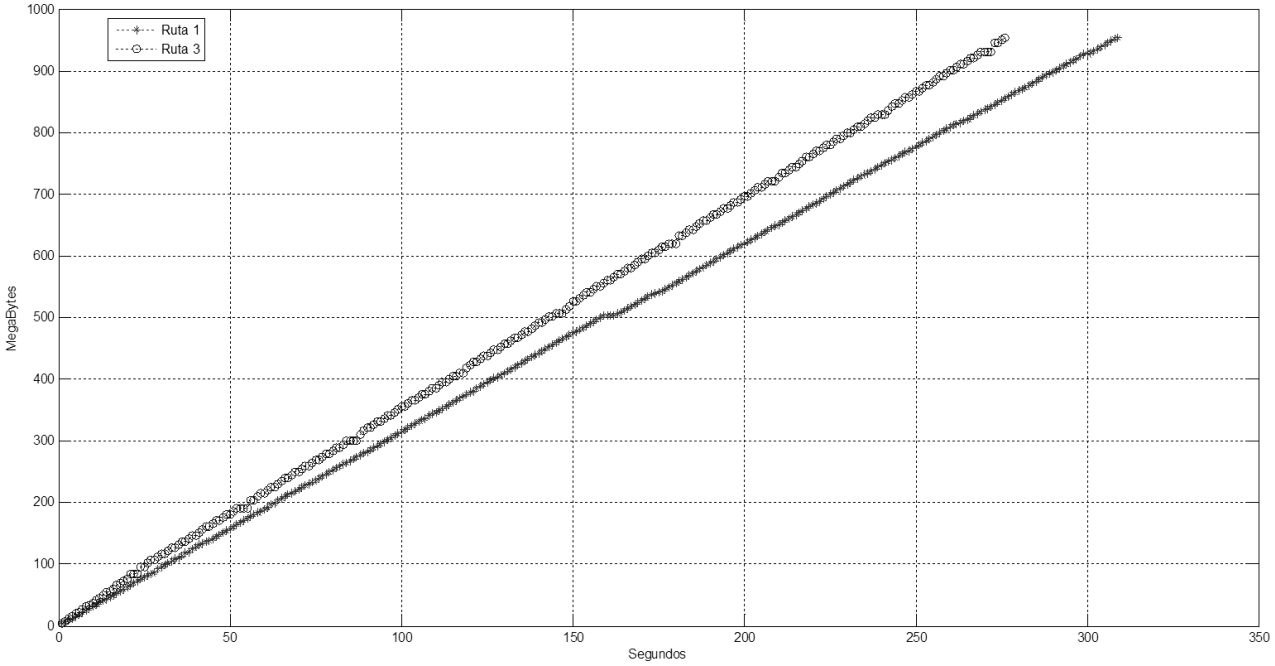

Figura 51. Recepción de información sobre TCP de forma acumulada en el tiempo como comparación de la trayectoria 1 (ruta más corta) y la trayectoria seleccionada por el algoritmo propuesto.

5.2.2.3 Aplicación de HPC sobre TCP en trayectorias similares solo con diferencia en retardo

Con la finalidad de comparar el algoritmo que utiliza el controlador de las SDN y el algoritmo propuesto se realizó una última prueba para el caso donde se tienen trayectorias en condiciones iguales de distancia, tasa de transmisión y pérdida de paquetes, con la única diferencia en el retardo para una trasferencia de información extremo a extremo.

Las trasmisiones de información se realizan durante un mismo periodo de tiempo de un minuto por ambas trayectorias. A pesar que las trayectorias tienen las mismas características de pérdida de paquetes la ruta B tiene menor retardo que la ruta A, el efecto que genera el alto retardo es la degradación del servicio para el flujo de información.

En la transmisión se observa el desempeño que se tiene por las diferentes trayectorias seleccionadas con base en la que tiene el menor retardo por cada algoritmo. Con el algoritmo de las SDN se elige una ruta A y con el algoritmo propuesto se elige una ruta B.

La mejora se obtiene en el rendimiento al realizar la trasferencia de información realizada por una trayectoria con menor retardo. Por la ruta B se tiene una mayor cantidad de información recibida, con 25Mbytes más que por la ruta A que corresponde a un 4.15% de mejora en el desempeño de la transferencia de la información. El desempeño de la ruta B se mantiene más tiempo estable comparado con la ruta A que tiene mayores caídas de velocidad.

En la Tabla 13 se muestra la información de la transmisión para ambas trayectorias. En la Figura 52 se muestra la comparación del desempeño con base en la cantidad de información recibida por cada segundo para ambas rutas.

Tabla 13. Transmisión de datos con TCP por diferentes rutas.

Ruta	Protocolo	Tiempo de Tarea (segundos)	Información Recibida (Mbyte)	PLR Promedio (%)
A	TCP	60	602	2
B	TCP	60	627	2

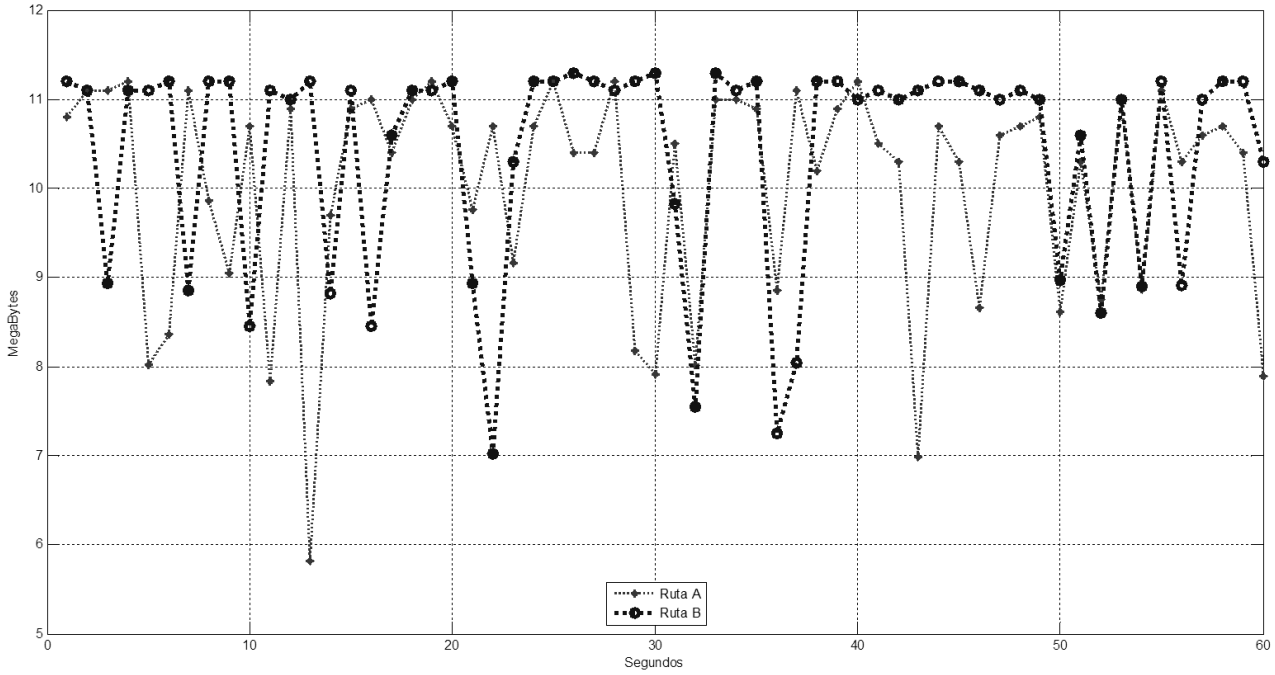

Figura 52. Información enviada en el tiempo, por la ruta alterna (ruta B) y la ruta más corta (ruta A).

5.2.2.4 Conclusión de desempeño de aplicaciones HPC con el algoritmo propuesto

Los resultados de las pruebas realizadas demuestran que efectuar la selección de la trayectoria con base en el retardo tiene ventajas:

(1) Aumenta el rendimiento de las aplicaciones agilizando los flujos de información.
(2) Disminuye el tiempo en que se realiza una tarea.
(3) Distribuye el tráfico por diferentes rutas.
(4) Por lo tanto disminuye la probabilidad de bloqueo.

Así como las restricciones de la tasa de transmisión evitan el estado de saturación y con las pérdidas de paquetes se descartan las trayectorias para mantener el desempeño de la aplicación.

Además se concluye que a mayor eficiencia en la tasa de transmisión, se refleja un mayor caudal eficaz (thoughput) al seleccionar la ruta con menor retardo y pérdidas de paquetes, el tiempo requerido para realizar una misma tarea o transferencia de información es menor, por lo tanto se mejora el desempeño de la aplicación.

En las Figuras 53 y 54, se ilustran las diferencias de desempeño para las trayectorias, la ruta más corta (Ruta 1 o ruta por defecto) y la trayectoria seleccionada con el algoritmo propuesto (Ruta 3).

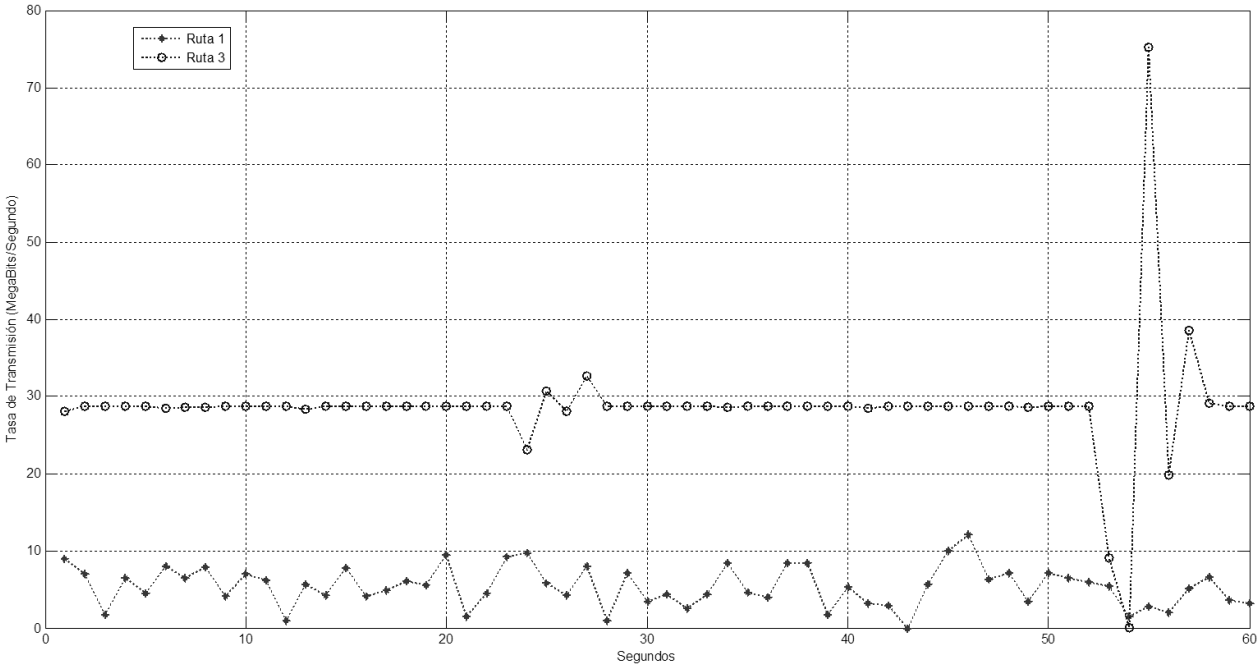

Figura 53. Tasa de transmisión en el tiempo de la ruta alterna (ruta 3) y la ruta más corta (ruta 1).

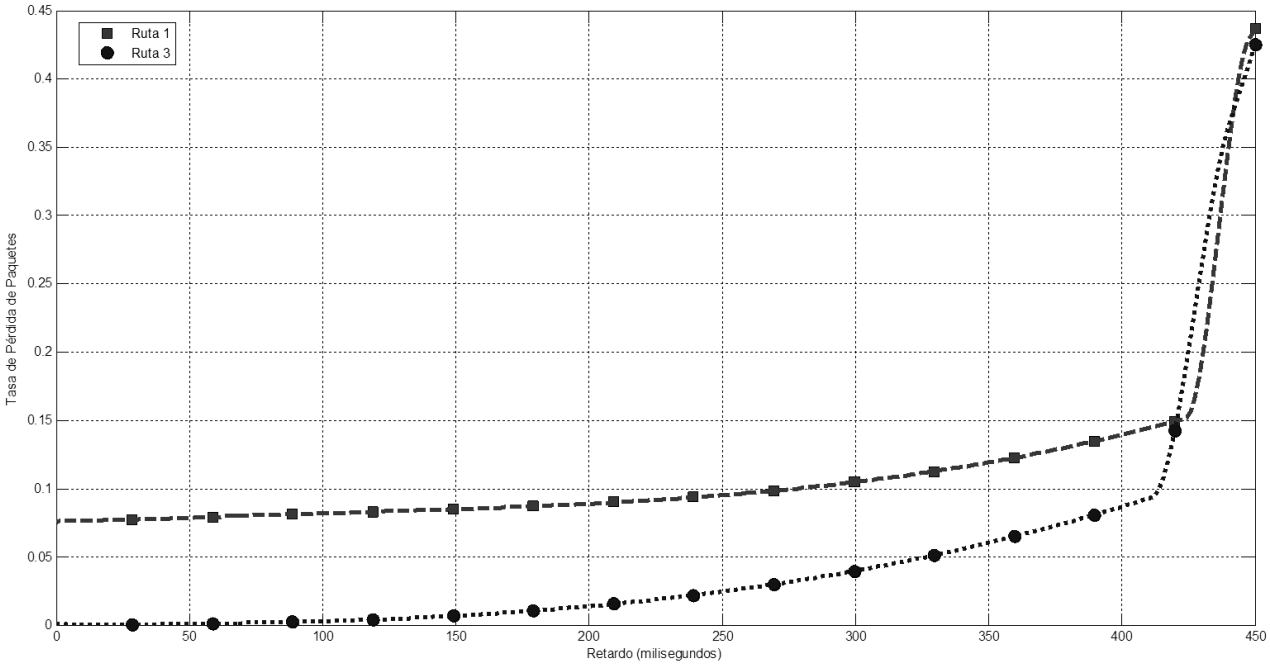

Figura 54. Desempeño con base al retardo y la pérdida de paquetes, entre la ruta más corta (Ruta 1) y la trayectoria elegida por el algoritmo propuesto (Ruta 3).

Esta página se ha dejado intencionalmente en blanco.

Capítulo 6. Análisis y Conclusiones

En este capítulo se exponen tanto el análisis como las conclusiones de la investigación realizada, recomendaciones y sugerencias del trabajo a futuro. La discusión se enfoca en las métricas de la emulación evaluada, el proceso del algoritmo para elegir una trayectoria que utiliza técnicas de capas cruzadas, la factibilidad de llevar a la práctica los resultados obtenidos y las respuestas al planteamiento del problema original. Finalmente se propone el posible trabajo a futuro tomando como punto de partida la investigación realizada en este documento.

6.1 Análisis e implicaciones en la selección de la ruta en SDN

La evaluación del comportamiento en la toma de decisiones para diferentes condiciones en las trayectorias, permite comprender el impacto que provoca el uso de métricas de calidad de servicio en este proceso, así como la mejora que se puede obtener en el desempeño de la red y en el rendimiento de las aplicaciones en una red SDN. Por un lado estabas proponiendo un algoritmo de enrutamiento que tomará en cuenta algunos parámetros de QoS pero por otro lado, un esquema para obtener la medición de los retardos de forma indirecta, sin inducir más tráfico de señalización.

La propuesta de aproximación del retardo no es exacta, se tiene un margen de error ya que se realiza el cálculo de manera indirecta mediante OpenFlow. El valor de retardo estimado en promedio es de 7.77 ms y el retardo real de 6.37 ms en promedio para la misma ruta en el mismo periodo de tiempo (Figura 57), por lo tanto se obtiene una aproximación al comportamiento que se tiene en la red.

Sin embargo, la aproximación del retardo tiene desventajas, ya que no es precisa y se tiene un leve retraso por el tiempo que tarda el proceso de cálculo. Es funcional y puede proveer las métricas de la red, como son solicitadas por los algoritmos de enrutamiento que proporcionan calidad de servicio. Los requisitos de este tipo de servicio tienden a ser estrictos y demandan una gran precisión de las métricas en la red.

La selección de trayectoria con base en el retardo estimado, es un acierto ya que la correlación entre las métricas de QoS es alta. El buen desempeño de un parámetro, se refleja en las demás métricas. Utilizar una métrica de QoS siempre lleva a una mejora en el rendimiento de la red, aunque algunas aplicaciones de baja prioridad pueden ser afectadas en su desempeño. Sin embargo para una solución completa se

requiere considerar más de una sola métrica. La QoS idealmente debería solo usarse en condiciones de saturación de la red, en la realidad debería utilizarse para evitar este estado en la red y la degradación del servicio. En general el enrutamiento en las redes se tiene que realizar en base a las métricas de QoS cuando sea necesario, a pesar que existen diferentes algoritmos para este propósito, ninguno se ha implementado o se utiliza con éxito.

Utilizar esquemas de enrutamiento basados en parámetros de QoS requieren de mecanismos para realizar métricas en tiempo real de las condiciones de la red, sin embargo las redes actuales no proveen dichos mecanismos. Por esto en la arquitectura de las SDN es necesaria la incorporación del retardo y las fluctuaciones en el monitoreo base (como ya se realiza para la tasa de transmisión y perdidas de paquetes) para que el controlador adquiera la información completa de las condiciones de la red como la propuesta en este trabajo de tesis.

6.2 Distribución del tráfico por diversas rutas

La ocurrencia en la selección de trayectoria con base en el retardo sigue una distribución de Poisson. Al tener un amplio margen de error en la estimación del retardo, se refleja en la distribución y se aleja del valor esperado. Aunque la distribución no es tan cercana a la esperada, se mantiene y se logra distribuir la ocurrencia entre las diferentes rutas posibles, así como para no elegir la ruta más corta por defecto.

La ruta más corta conforme aumenta el tráfico hacia el punto de la saturación, la distribución cambia, disminuye la ocurrencia de la misma y desplaza la media a una segunda ruta. Cuando la ruta de menor distancia se encuentra en su máxima capacidad, la probabilidad de ser seleccionada debería ser cero, a pesar que disminuye se sigue considerando como opción a elegir. Por esto es necesario considerar más de un solo parámetro al elegir una ruta. Cada parámetro considerado tiene un objetivo particular. La tasa de transmisión para limitar los recursos disponibles y evitar la saturación de los enlaces. La pérdida de paquetes ayuda a mantener el mayor caudal eficaz, además la selección de ruta con base en el retardo permite mantener un rápido y ágil flujo de la información.

6.3 Sobre el algoritmo propuesto

El algoritmo propuesto es funcional, llega a la meta propuesta de elegir la trayectoria con mejores condiciones, dando mayor agilidad al flujo de información, reduce las pérdidas de paquetes, por lo tanto

reduce el tiempo que se requiere para realizar una tarea y así como mejorar el rendimiento de la aplicación.

Para los diferentes casos presentados en el capítulo 6 se logró una mejora en el desempeño de la transmisión de información extremo a extremo. Con el algoritmo propuesto, en un mismo periodo de tiempo se transfiere de 3% a 4.15% más de información para diferentes casos, comparado con el uso del algoritmo por defecto de la ruta más corta. También se logra incrementar la tasa de transferencia al reducir la pérdida de paquetes, al igual se logra garantizar un retardo bajo de paquetes enviados, así mismo se logró mejorar el tiempo que tarda en realizarse una transmisión de información.

La tasa alta de pérdidas de la ruta más corta incrementa el número de retransmisiones. Para un caso mostrado en el presente trabajo se redujo en 33.2 segundos el tiempo de una misma cantidad de información, que corresponde a una mejora del 10.74% del tiempo de la tarea. Por lo tanto, una misma tarea tarda menos tiempo en realizarse a través de la trayectoria alterna elegida por el algoritmo propuesto, a diferencia del algoritmo de la ruta más corta seleccionada por defecto.

La desventaja de utilizar los tres parámetros se refleja en el tiempo polinómico (tiempo que tarda en terminar el proceso y/o procesamiento de selección para las ecuaciones) que se requiere para grandes cantidades de tráfico, no es viable su implementación. Se requieren métodos o algoritmos que reduzcan estos tiempos de procesamiento en el enrutamiento para la calidad de servicio.

El uso de técnicas de capas cruzadas es una parte esencial en el proceso de selección de trayectorias y una herramienta precisa para optimizar capas en la red. Sin este mecanismo el controlador SDN no es capaz de conocer los requerimientos de las aplicaciones, ni considerar las condiciones en la red como parte de la toma de decisiones.

6.4 Conclusiones generales

En general, queda demostrada la importancia de los parámetros de calidad de servicio en el enrutamiento de las redes. El utilizar mecanismos de enrutamiento que consideren parámetros de QoS puede ser una alternativa para mantener un desempeño adecuado de aplicaciones que requieran ciertas prestaciones cuando la red presenta condiciones de saturación. En condiciones de no saturación, los algoritmos de la ruta más corta ofrecerán un mejor desempeño para todas las aplicaciones. Tal vez sería

adecuado que la red cambiara dinámicamente el mecanismo de enrutamiento dependiendo de las condiciones de saturación de la misma.

La tendencia en las redes se ha enfocado en simplificar el trabajo de configuración y manteamiento, SDN es un claro ejemplo de esta tendencia. Además no ha sido explotada la calidad de servicio, ha sido relegada por esta misma razón, ya que las propuestas de QoS tienen un grado alto de complejidad que a pocos les interesa implementar y configurar. Aún y cuando los beneficios que se tienen son excelentes tanto para el rendimiento de la red como para el desempeño de las aplicaciones. Un proveedor de servicios de red puede mejorar su gama de servicios ofrecidos utilizando los mismos recursos de su infraestructura de red.

La idea general es seguir esta tendencia, así como generar una base para la calidad de servicio, simplificarla en su implementación y configuración junto a SDN, ya que sebe ser parte indispensable en la esencia de la arquitectura y por lo tanto que su uso sea fácil, simple y no vuelva a ser un componente olvidado por las redes de datos.

6.5 Aportaciones

La aportación de este trabajo de investigación se da principalmente en la obtención de un algoritmo/mecanismo simple de selección de trayectoria con base en las métricas de QoS de la red, para evitar utilizar la ruta más corta por defecto. La obtención de este algoritmo se realizó considerando las aproximaciones resultantes del monitoreo de bajo nivel, de las latencias y retardos en la red utilizando OpenFlow. Adicionalmente del uso de técnicas de capas cruzadas hace las funciones de abstracción de los parámetros en la red y los requerimientos de las aplicaciones. Estas abstracciones se utilizan para tomar la decisión de la trayectoria.

6.6 Trabajo a futuro

Como trabajo a futuro se recomienda realizar el monitoreo de bajo nivel de manera nativa a través de OpenFlow, como es realizado para los parámetros de pérdida de paquetes y tasa de transmisión. Una alternativa a esto es mejorar la precisión en los cálculos de la estimación para los parámetros de latencia, retardo y fluctuaciones del retardo.

Mejorar e implementar de manera nativa las técnicas de capas cruzadas en el controlador, o un módulo que permita la abstracción de los requerimientos y las condiciones de red, para que las decisiones en el controlador se realicen con base en métricas de QoS por defecto o activarla de manera más sencilla. Además otorgarle la importancia y peso a cada métrica de QoS, no solo dársela a la tasa de transmisión.

La implementación de QoS en SDN es de gran prioridad sobre todo en el enrutamiento, esto es indispensable para evitar la congestión de rutas, mejorar el servicio de las redes y el desempeño de las aplicaciones de alta prioridad o misión crítica. Así también se requiere la implementación de otros mecanismos de QoS en SDN como la priorización, clasificación de flujos, calendarizadores, la garantía de la tasa de transmisión mínima, etc., así como otras que permitan al controlador mejorar la gestión del tráfico.

También se observó que hacen falta trabajar en mecanismos de seguridad, como encriptación, canales seguros, redundancia, etc. Existen propuestas y herramientas para ello, pero al igual que QoS son herramientas en un segundo plano, ya que no se les ha dado la importancia que debería y no han sido explotadas ampliamente.

Esta página se ha dejado intencionalmente en blanco.

Literatura Citada

Ab, E. (2014). The real-time cloud. *Ericsson News*, 1–10. Retrieved from http://docplayer.net/87266-The-real-time-cloud-ericsson-white-paper-uen-284-23-3219-rev-b-february-2014.html

Akyildiz, I. F., Lee, A., Wang, P., Luo, M., & Chou, W. (2014). A roadmap for traffic engineering in SDN-OpenFlow networks. *Computer Networks*, *71*, 1–30. http://doi.org/10.1016/j.comnet.2014.06.002

Al-Fares, M., Radhakrishnan, S., & Raghavan, B. (2010). Hedera: Dynamic Flow Scheduling for Data Center Networks. *Nsdi*, *10*, 19. Retrieved from https://www.usenix.org/legacy/event/nsdi10/tech/full_papers/al-fares.pdf

Al-somaidai, M. B., & Yahya, E. B. (2014). Survey of software components to emulate OpenFlow protocol as an SDN implementation. *American Journal of Software Engineering and Applications*, *3*(6), 74–82. http://doi.org/10.11648/j.ajsea.20140306.12

Arefin, A., Rivas, R., Tabassum, R., & Nahrstedt, K. (2013). OpenSession: SDN-based cross-layer multi-stream management protocol for 3D teleimmersion. *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 1–10. http://doi.org/10.1109/ICNP.2013.6733616

Astuto, B., Nunes, A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *Ieee Communications Surveys & Tutorials*, *16*(3), 1617–1634. http://doi.org/10.1109/SURV.2014.012214.00180

Azizi, M., Benaini, R., & Mamoun, M. Ben. (2015). Delay Measurement in Openflow-Enabled MPLS-TP Network, *9*(3), 90–101. http://doi.org/10.5539/mas.v9n3p90

Belter, B., Binczewski, A., Dombek, K., Juszczyk, A., Ogrodowczyk, L., Parniewicz, D., … Olszewski, I. (2014). Programmable Abstraction of Datapath. *2014 Third European Workshop on Software Defined Networks*, 7–12. http://doi.org/10.1109/EWSDN.2014.10

Belter, B., Parniewicz, D., Ogrodowczyk, L., Binczewski, A., Stroinski, M., Fuentes, V., … Jacob, E. (2014). Hardware Abstraction Layer as an SDN-enabler for Non-OpenFlow Network Equipment. *2014 Third European Workshop on Software Defined Networks*, 117–118. http://doi.org/10.1109/EWSDN.2014.16

Berliner, B., Clark, B., & Hartono, A. (2011). QoS Requirements of Multimedia Applications. *Department of Computer Science and Engineering- The Ohio State University*, 1–10. Retrieved from http://web.cse.ohio-state.edu/~xuan/courses/679/topic_1_rep.doc

Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., … Seskar, I. (2014). GENI: A federated testbed for innovative network experiments. *Computer Networks*, *61*, 5–23. http://doi.org/10.1016/j.bjp.2013.12.037

Bianchi, G., Bonola, M., Capone, A., & Cascone, C. (2014). OpenState. *ACM SIGCOMM Computer Communication Review*, *44*(2), 44–51. http://doi.org/10.1145/2602204.2602211

Brocade. (2012). The Brocade OpenScript Engine : Carrier-Grade Flexibility for Scalable and Reliable Application Delivery. *Brocade Communications Systems Inc.*, 1–9. Retrieved from https://www.brocade.com/content/dam/common/documents/content-types/feature-guide/brocade-openscript-engine-ga-tb.pdf

Buh, T., Trobec, R., & Ciglič, A. (2014). Adaptive network-traffic balancing on multi-core software networking devices. *Computer Networks*, *69*, 19–34. http://doi.org/10.1016/j.comnet.2014.04.015

Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., & Shenker, S. (2007). Ethane: taking control of the enterprise. *ACM Sigcomm '07 Computer Communication Review*, *37*(4), 1–12. http://doi.org/http://doi.acm.org/10.1145/1282380.1282382

Cisco. (2014). OpFlex: An Open Policy Protocol. *IETF*, 1–6. Retrieved from http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731302.html

Cisco. (2015). Cisco Open SDN Controller 1.2. *Cisco Documentation*, 1–5. Retrieved from http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/open-sdn-controller/datasheet-c78-733458.html

Clark, D. D., & Fang, W. (1998). Explicit Allocation of Best-Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking*, *6*(4), 362–373. http://doi.org/10.1109/90.720870

Cummings, U. (2006). A Low-Latency , High-Bandwidth Ethernet Switch Chip. *Hot Chips 18 Symposium (HCS), 2006 IEEE*, 1–26. Retrieved from http://ieeexplore.ieee.org/abstract/document/7477870/

Curtis, A. R., Kim, W., & Yalagandula, P. (2011). Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. *IEEE INFOCOM*, *INFOCOM*, *2*, 1629–1637. http://doi.org/10.1109/INFCOM.2011.5934956

DeCusatis, C. (Ed.). (2013). Handbook of fiber optic data communication: a practical guide to optical networking (pp. 427–445). Academic Press.

Dietz, T. (2012). Trema Tutorial. *NEC*, 1–73. Retrieved from http://exp.i2cat.fp7-ofelia.eu/assets/Uploads/201203xx-TremaTutorial.pdf

Duan, Q. (2014). Network-as-a-Service in Software-Defined Networks for end-to-end QoS provisioning. *2014 23rd Wireless and Optical Communication Conference (WOCC)*, 1–5. http://doi.org/10.1109/WOCC.2014.6839919

E. Haleplidis, E., K. Pentikousis, E., Denazis, S., Salim, J. H., Meyer, D., & Koufopavlou, O. (2015). Software-Defined Networking (SDN): Layers and Architecture Terminology. *IRTF*, RFC 7426. Retrieved from https://tools.ietf.org/html/rfc7426

Egilmez, H. E. (2012). *Adaptive video streaming over openflow networks with quality of service (Doctoral dissertation, Koç University).* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.707.4303&rep=rep1&type=pdf

Egilmez, H. E., & Dane, S. T. (2012). OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, *IEEE*, 1–8. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6411795

Erickson, D. (2013). The beacon openflow controller. *Second ACM SIGCOMM Workshop …*, 13–18. http://doi.org/10.1145/2491185.2491189

Farhady, H., Lee, H., & Nakao, A. (2015). Software-Defined Networking: A survey. *Computer Networks*, *81*, 79–95. http://doi.org/10.1016/j.comnet.2015.02.014

Farinacci, D., Fuller, V., Meyer, D., Lewis, D., & Cisco Systems. (2013). The Locator/ID Separation Protocol (LISP). *The Internet Protocol J*, *11*(1), 23–36. Retrieved from https://tcp0.com/rfc/rfc6830.txt.pdf

Fortz, B., Rexford, J., & Thorup, M. (2002). Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, *40*(10), 118–124. http://doi.org/10.1109/MCOM.2002.1039866

Fujitsu. (2014). SDN and the Future of Service Provider Networks. *The Optical Networking and Communication Conference & Exhibition*, 1–8. Retrieved from https://www.ofcconference.org/getattachment/68378ba8-f0a1-4b02-adae-65ee28993ee2/SDN-and-the-Future-of-Service-Provider.aspx

Göransson, P., & Black, C. (2014). Software Defined Network A Comprehensive Approach. In *Elsevier* (pp. 37–168). Elsevier.

Greenberg, A., Hjalmtysson, G., Maltz, D. a., Myers, A., Rexford, J., Xie, G., … Zhang, H. (2005). A clean slate 4D approach to network control and management. *ACM SIGCOMM Computer Communication Review*, *35*(3), 41. http://doi.org/10.1145/1096536.1096541

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). NOX: towards an operating system for networks. *SIGCOMM Computer Communication Review*, *38*(3), 105–110. http://doi.org/10.1145/1384609.1384625

Guo, Q., Feng, R., Wu, Y., & Yu, N. (2016). Measurement of the AFDX Switch Latency Based on FPGA. *Aircraft Utility Systems, IEEE International Conference*, 45–49. Retrieved from http://ieeexplore.ieee.org/abstract/document/7748018/?reload=true

Guo, Z., Xu, Y., Cello, M., Zhang, J., Wang, Z., Liu, M., & Chao, H. J. (2015). JumpFlow : Reducing flow table usage in software-defined networks. *Computer Networks*, *92*, 300–315. http://doi.org/10.1016/j.comnet.2015.09.030

Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., & Gayraud, T. (2014). Software-Defined Networking: Challenges and research opportunities for Future Internet. *Computer Networks*, *75*, 453–471. http://doi.org/10.1016/j.comnet.2014.10.015

Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., & Mckeown, N. (2012). Mininet Performance Fidelity Benchmarks. *Tech. Rep.*, 1–11. Retrieved from https://pdfs.semanticscholar.org/5ac2/6965cd23a2658f6a64ac2bdf99a16d04642e.pdf

Hewlett-Packard. (2013). HP VAN SDN Controller Installation Guide. *HP NEWS*, 1–14. Retrieved from http://h20566.www2.hp.com/portal/site/hpsc/template.BINARYPORTLET/public/kb/docDisplay/resource.process/?spf_p.tpst=kbDocDisplay_ws_BI&spf_p.rid_kbDocDisplay=docDisplayResURL&javax.portlet.begCacheTok=com.vignette.cachetoken&spf_p.rst_kbDocDisplay=wsrp-re

Hu, Y., Wang, W., Gong, X., Que, X., & Cheng, S. (2013). BalanceFlow: Controller load balancing for OpenFlow networks. *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, IEEE CCIS 2012*, *2*, 780–785. http://doi.org/10.1109/CCIS.2012.6664282

Jammal, M., Singh, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking : State of the art and research challenges. *COMPUTER NETWORKS*, *72*, 74–98. http://doi.org/10.1016/j.comnet.2014.07.004

Jokanovic, A., Sancho, J. C., Labarta, J., Rodriguez, G., & Minkenberg, C. (2012). Effective Quality-of-Service Policy for Capacity High-Performance Computing Systems. *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference*, 598–607. http://doi.org/10.1109/HPCC.2012.86

Kiran, S., & Kinghorn, G. (2015). Cisco Open Network Environment: Bring the Network Closer to Applications. *Cisco*, 1–14. Retrieved from http://www.cisco.com/c/en/us/products/collateral/switches/nexus-1000v-switch-vmware-vsphere/white_paper_c11-728045.pdf

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., … Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. *9th USENIX Conference on Operating Systems Design and Implementation*, *10*, 1--6. http://doi.org/10.1.1.186.3537

Kreutz, D., Ramos, F. M. V, Verissimo, P., Rothenberg, C. E., Azodolmolky, S., Uhlig, S., … Ramos, F. (2014). Software-Defined Networking: A Comprehensive Survey. *IEEE*, *103*(1), 14–76. http://doi.org/10.1109/JPROC.2014.2371999

Kritzinger, P. (1986). A performance model of the OSI communication architecture. *IEEE Transactions on Communications*, *34*(6), 554–563. http://doi.org/10.1109/TCOM.1986.1096579

Luo, M., Luo, M., Tian, Y., Li, Q., Wang, J., & Chou, W. (2012). SOX - A Generalized and Extensible Smart Network Openflow Controller SOX - A Generalized and Extensible S mart Network O penflow Controller (X). *First SDN World Congress*, 1–9. http://doi.org/10.13140/2.1.2130.5123

Marchese, M. (2007). QoS over heterogeneous networks. In *John Wiley & Sons* (p. 7).

Maugendre, M. (2015). Development of a performance measurement tool for SDN. *Universitat Politecnica de Catalunya (UPC) ` – BarcelonaTech*, 1–57. Retrieved from http://upcommons.upc.edu/handle/2117/79127

Mckeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., … Louis, S. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, *38*(2), 69–74. http://doi.org/10.1145/1355734.1355746

Metzler, J., & Metzler, A. (2015). The 2015 Guide to SDN and NFV. *A10*, 1–89. Retrieved from https://www.a10networks.com/sites/default/files/resource-files/2015Ebook-A10-all.pdf

Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., Curtis, a R., & Banerjee, S. (2010). Devoflow: Cost-effective flow management for high performance enterprise networks. *9th ACM SIGCOMM*

Workshop on Hot Topics in Networks, 1–6. http://doi.org/10.1145/1868447.1868448

Moura, H., Bessa, G. V. C., Vieira, M. A. M., & Macedo, D. F. (2015). Ethanol: Software Defined Networking for 802.11 Wireless Networks. *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, *IEEE*, 388–396.

Oktian, Y. E., Lee, S., Lee, H., & Lam, J. (2015). Secure Your Northbound SDN API. *2015 Seventh International Conference on Ubiquitous and Future Networks*, *IEEE*, 919–920. Retrieved from http://ieeexplore.ieee.org/document/7182679/?arnumber=7182679&tag=1

Olivares, O. (2008). *Ajuste Automático De Parámetros De Qos En Videoconferencia Móvil Utilizando Cross Layer Design Para Redes Inalámbricas 3g Ev-Do*. Retrieved from http://biblioteca.cicese.mx/catalogo/tesis/ficha.php?id=17861

ONF. (2012). OpenFlow Switch Specification 1.3.1. *Open Network Foundation*, 1–128. Retrieved from https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

ORACLE. (2014). *Oracle SDN Controller User Guide*. Retrieved from https://docs.oracle.com/cd/E48586_01/pdf/E49478.pdf

Paul, S., & Jain, R. (2012). OpenADN: Mobile apps on global clouds using OpenFlow and Software Defined Networking. *2012 IEEE Globecom Workshops, GC Wkshps 2012*, 719–723. http://doi.org/10.1109/GLOCOMW.2012.6477663

Pfaff, B., B. Davie, E., & VMware, I. (2013). The Open vSwitch Database Management Protocol. *IETF*, *RFC7047*, 1–35. Retrieved from https://tools.ietf.org/html/rfc7047

Prete, L. R., Shinoda, A. A., Schweitzer, C. M., & De Oliveira, R. L. S. (2014). Simulation in an SDN network scenario using the POX Controller. *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference*, *IEEE*, 1–6. http://doi.org/10.1109/ColComCon.2014.6860403

Quintana, A. A. (2001). *Encaminamiento en redes con calidad de servicio*. Retrieved from http://www.biblioteca.uma.es/bbldoc/tesisuma/16272614.pdf

R. Enns, E., & Juniper. (2006). NETCONF Configuration Protocol. *IETF*, *RFC4741*, 1–95. Retrieved from https://tools.ietf.org/html/rfc4741

Raayatpanah, M. A., Fathabadi, H. S., Khalaj, B. H., Khodayifar, S., & Pardalos, P. M. (2014). Journal of Network and Computer Applications Bounds on end-to-end statistical delay and jitter in multiple multicast coded packet networks. *Journal of Network and Computer Applications*, *41*, 217–227. http://doi.org/10.1016/j.jnca.2013.12.004

Rumble, S. M., Ongaro, D., Stutsman, R., Rosenblum, M., & Ousterhout, J. K. (2011). It ' s Time for Low Latency. *HotOS*, *13*, 11. Retrieved from http://static.usenix.org/event/hotos11/tech/final_files/Rumble.pdf

Salih, M. A., Cosmas, J., & Zhang, Y. (2015). OpenFlow 1.3 Extension for OMNeT++. *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*, *2015 IEEE*, 1632–1637. http://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.246

Seddiki, M. S., Shahbaz, M., Donovan, S., Grover, S., Park, M., Feamster, N., & Song, Y.-Q. (2014). FlowQoS: QoS for the rest of us. *Third Workshop on Hot Topics in Software Defined Networking - HotSDN '14*, 207–208. http://doi.org/10.1145/2620728.2620766

Serrano, P., & Hern, A. (2016). Una introducción amable a la teoría de colas. *Departamenteo de Ingeniería Telemática - Universidad Carlos III de Madrid*, 1–156. Retrieved from http://www.it.uc3m.es/pablo/teoria-colas/introduccion-teoria-colas.pdf

Shenker, S., & Casado, M. (2011). The future of networking, and the past of protocols. *Open Networking Summit*, *20*, 1–30. Retrieved from http://www.opennetsummit.org/archives/apr12/site/talks/shenker-tue.pdf

Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., ... & Kang, B. B. (2014). Rosemary : A Robust, Secure, and High-Performance Network Operating System Categories and Subject Descriptors. *2014 ACM SIGSAC Conference on Computer and Communications Security*, 78–89. Retrieved from

http://dl.acm.org/citation.cfm?id=2660353

Singla, A., & Rijsman, B. (2013). Contrail Architecture. *Juniper Networks*, 1–44. Retrieved from http://www.juniper.net/us/en/local/pdf/whitepapers/2000535-en.pdf

Song, H. (2013). Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane. *Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 127–132. http://doi.org/10.1145/2491185.2491190

Spurgeon, C. (2000). *Ethernet: the definitive guide*. O'Reilly Media, Inc.

Srivastava, V., & Motani, M. (2005). Cross-layer design: a survey and the road ahead. *IEEE Communications Magazine*, *43*(12), 112–119. Retrieved from http://ieeexplore.ieee.org/document/1561928/?arnumber=1561928&tag=1

Tootoonchian, A., & Ganjali, Y. (2010). Hyperflow: a distributed control plane for openflow. *2010 Internet Network Management Conference on Research on Enterprise Networking*, 1–78. Retrieved from http://dl.acm.org/citation.cfm?id=1863133.1863136%5Cnhttps://www.usenix.org/legacy/event/inmwren10/tech/full_papers/Tootoonchian.pdf

Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. (2012). On controller performance in software-defined networks. *2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, 1–6. Retrieved from https://www.usenix.org/system/files/conference/hot-ice12/hotice12-final33_0.pdf

Tsung-Feng Yu, Kuochen Wang, & Yi-Huai Hsu. (2015). Adaptive routing for video streaming with QoS support over SDN networks. *2015 International Conference on Information Networking (ICOIN)*, *IEEE*, 318–323. http://doi.org/10.1109/ICOIN.2015.7057904

UIT-T. (2008). UIT-T E.800 Definiciones de términos relativos a la calidad de servicio. *Unión Internacional de Telecomunicaciones UIT-T*, *Serie E*, 1–34. Retrieved from http://www.itu.int/rec/T-REC-E.800-200809-I

VMware. (2015). VMware NSX for vSphere (NSX-V). *Network Virtualization Design Guide*, 1–93. Retrieved from http://www.icc-usa.com/resources/vmw-nsx-network-virtualization-design-guide.pdf

Wetteroth, D. (2001). *OSI reference model for telecommunications*. McGraw-Hill Professional.

World Wide Technology. (2014). Software-Defined Networking Guide. *Software-Defined Networking Solution Workshop*, 1–20. Retrieved from https://www2.wwt.com/wp-content/uploads/2015/06/White-Paper-SDN-Solution-Guide.pdf

Yang, L., Dantu, R., Anderson, T., & Gopal, R. (2004). Forwarding and control element separation (ForCES) framework. *IETF*, *RFC 3746*, 1–40. Retrieved from https://www.rfc-editor.org/rfc/pdfrfc/rfc3746.txt.pdf

Yeganeh, S. H., & Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. *First Workshop on Hot Topics in Software Defined Networks*, *ACM*, 19–24. http://doi.org/10.1145/2342441.2342446

Yu, C., Lumezanu, C., Sharma, A., Xu, Q., Jiang, G., & Madhyastha, H. V. (2015). Software-defined latency monitoring in data center networks. *Springer International Publishing, In International Conference on Passive and Active Network Measurement*, 360–372. http://doi.org/10.1007/978-3-319-15509-8_27

Yu, M., Rexford, J., Freedman, M. J., & Wang, J. (2010). Scalable flow-based networking with DIFANE. *ACM SIGCOMM Computer Communication Review*, *40*(4), 351. http://doi.org/10.1145/1851275.1851224

Zhou, W., Li, L., Luo, M., & Chou, W. (2014). REST API Design Patterns for SDN Northbound API. *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, 358–365. http://doi.org/10.1109/WAINA.2014.153

Zorba, N. , Skianis C. and Verikoukis, C. (2011). Cross Layer Designs in WLAN Systems. In *Troubador Publishing* (pp. 1–33).

Esta página se ha dejado intencionalmente en blanco.

Anexos

Anexo A

OpenFlow fue desarrollado y diseñado para permitir a los investigadores experimentar e innovar nuevos protocolos en la redes. OpenFlow es el protocolo de transporte o mejor conocido como interface *Southbound* para SDN, que mantiene la comunicación entre un controlador lógicamente centralizado y los conmutadores, además de delinear el comportamiento esperado del conmutador (Göransson & Black, 2014).

Las diferentes versiones que han desarrollado se muestran en la Tabla 14. Las versiones más utilizadas son es la versión 1.0 y 1.3, que se utilizan en los conmutadores físicos y simuladores de las SDN. A pesar que la versión más nueva es la versión 1.4, ningún simulador SDN tiene soporte para esta versión.

Tabla 14. Características de las diferentes versiones de OpenFlow (Al-somaidai & Yahya, 2014).

Versión	**1.0**	**1.1**	**1.2**	**1.3**	**1.4**
Fecha de publicación	Dic. 31, 2009	Feb. 28,2011	Dic. 5, 2011	Jun.25, 2012	Oct. 15, 2013
Ampliamente desplegado	Si	No	No	No	No
Tabla de flujos	Individual	Múltiple	Múltiple	Múltiple	Múltiple
Grupo de tablas	No	Si	Si	Si	Si
Tabla de metro	No	No	No	Si	Si
Etiqueta VLAN y MPLS	No	Si	Si	Si	Si
Fallo de conexión del controlador	Emergency	Stand alone	Stand alone /secure	Stand alone /secure	Stand alone /secure
Soporte IPV6	No	No	Si	Si	Si
Múltiples controladores	No	No	Si	Si	Si
Desalojo/Vacante /Sincronización	No	No	No	No	Si

La mensajería de OpenFlow se dividen en tres categorías generales que se muestran en la Figura 55a: simétrica, controlador-conmutador y asíncrona.

Estos mensajes se utilizan normalmente durante los 3 procesos básicos de OpenFlow para el diálogo entre controlador y conmutador, los cuales son: inicialización, operación y monitoreo, como se muestra en la Figura 55b (Göransson & Black, 2014).

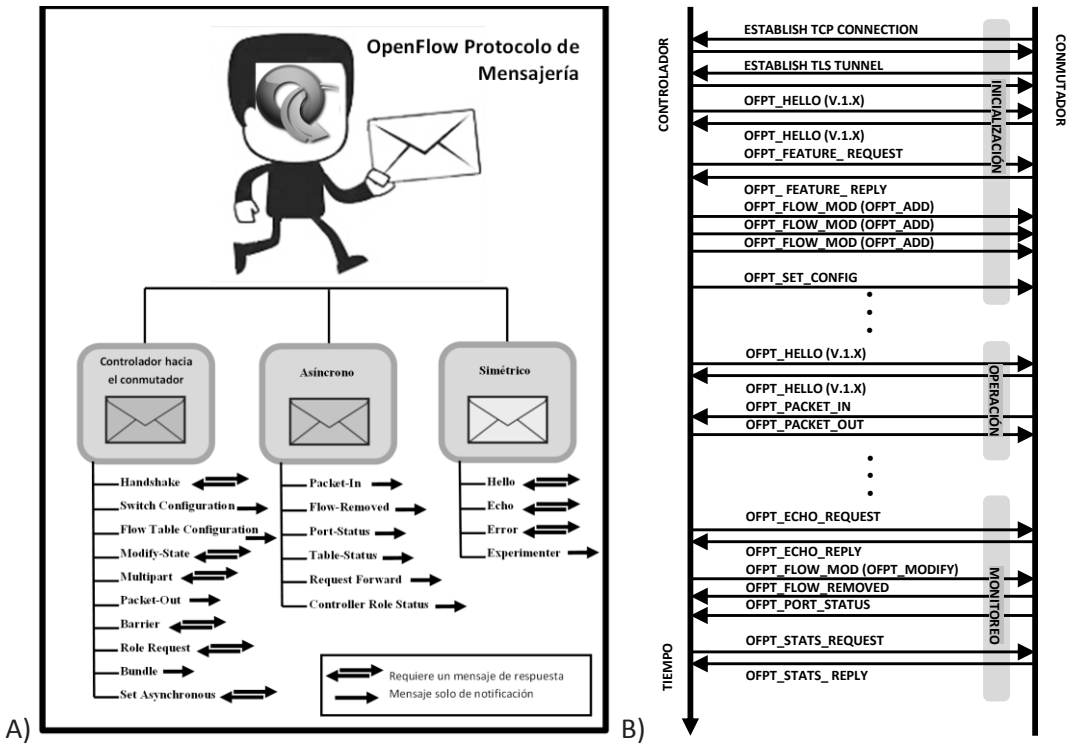

Figura 55. Tipos de mensaje de OpenFlow A) (Al-somaidai & Yahya, 2014) B) (Göransson & Black, 2014).

Los mensajes del controlador hacia el conmutador se utilizan para afirmar su control sobre el conmutador, leer el estado del conmutador y realizar modificaciones de los estados de conmutación, que incluye la edición de las tablas de flujo. Los mensajes del conmutador para el controlador se utilizan para informar al controlador acerca de un nuevo flujo de entrada, un cambio en un estado de conmutación, o una petición para modificar una entrada de la tabla de flujo (Al-somaidai & Yahya, 2014).

Durante la fase de monitoreo, el controlador obtiene información que le proporciona una vista global de la red, las características de los puertos, flujos, errores, conectividad, prioridad y tasas de transmisión (Al-somaidai & Yahya, 2014).

Anexo B

La estimación del retardo es una aproximación al retardo real de los paquetes, esta estimación no es exacta pero es una buena aproximación de las condiciones de la red. En la Figura 56 se compara el retardo real que se tiene durante una transmisión de tráfico moderado sobre la ruta más corta y el retardo estimado de la misma. Donde el retardo promedio real es de 6.37 ms y el retardo promedio estimado es de 7.77 ms.

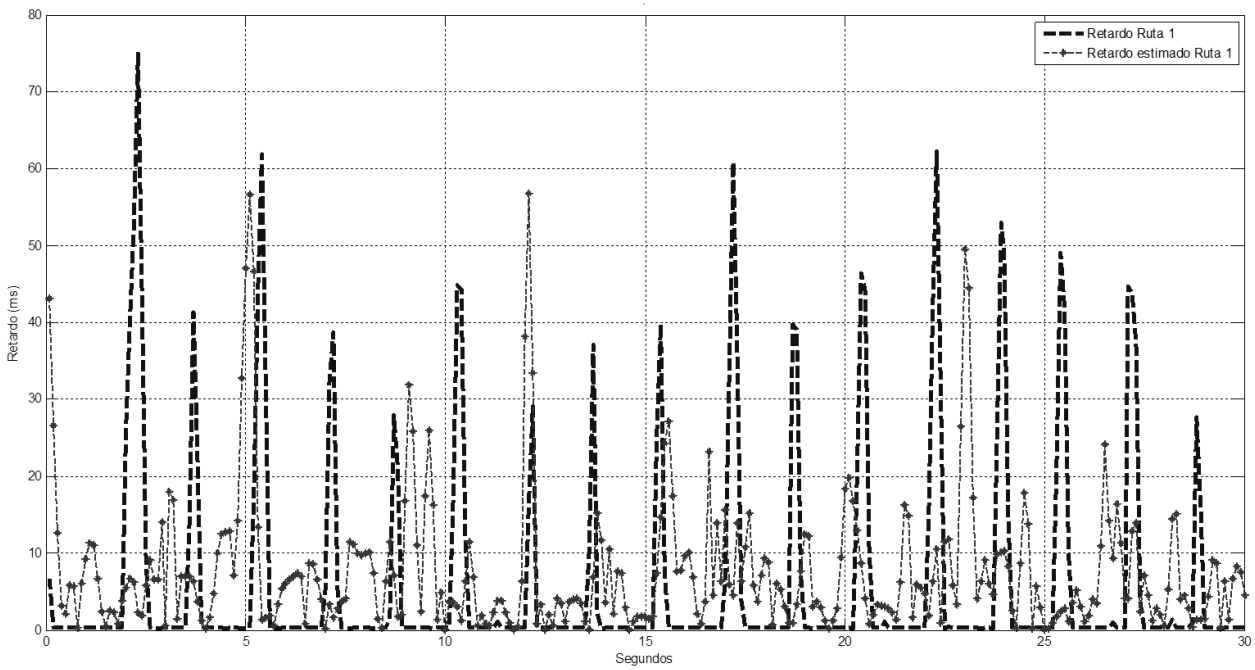

Figura 56. Retardo en transferencia de información con tráfico moderado en la ruta más corta.

Al ser un ambiente controlado las diferencias en el rendimiento promedio por cada trayectoria es ligeramente notable, ya que las rutas mantienen un comportamiento similar por la estructura del simulador.

Al observar el desempeño de algunas de las rutas, estas tienen un comportamiento similar pero con valores promedio separados ligeramente. Esto es una gran diferencia en rendimiento, donde se requiere elegir la mejor entre ellas que cumpla las restricciones.

El comportamiento de las trayectorias al ser variables aleatorias dependientes su comportamiento muestra que el retardo es directamente proporcional al tráfico en la trayectoria. Cuanto mayor sea el tráfico o se encuentre al límite de su capacidad el enlace el retardo aumenta exponencialmente. Al igual las pérdidas de paquetes también tienen una relación directa con el retardo al aumentar éste, las perdidas también se incrementan de la misma manera.

El caudal eficaz (thoughput) mantiene una relación inversa con el retardo, donde éste disminuye cuando el retardo aumenta. El desempeño promedio de retardo contra las pérdidas de paquetes que se ilustra en la Figura 57 y el retardo contra el caudal eficaz normalizado en la Figura 58.

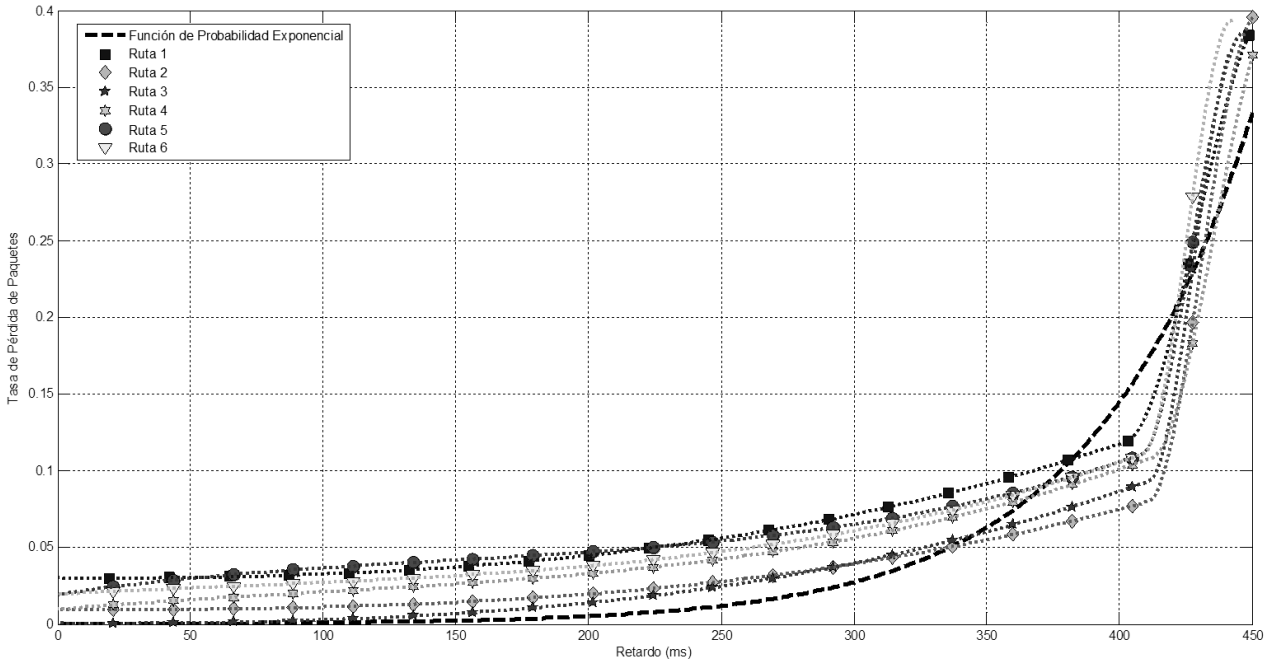

Figura 57. Desempeño de algunas trayectorias con base al retardo y la pérdida de paquetes.

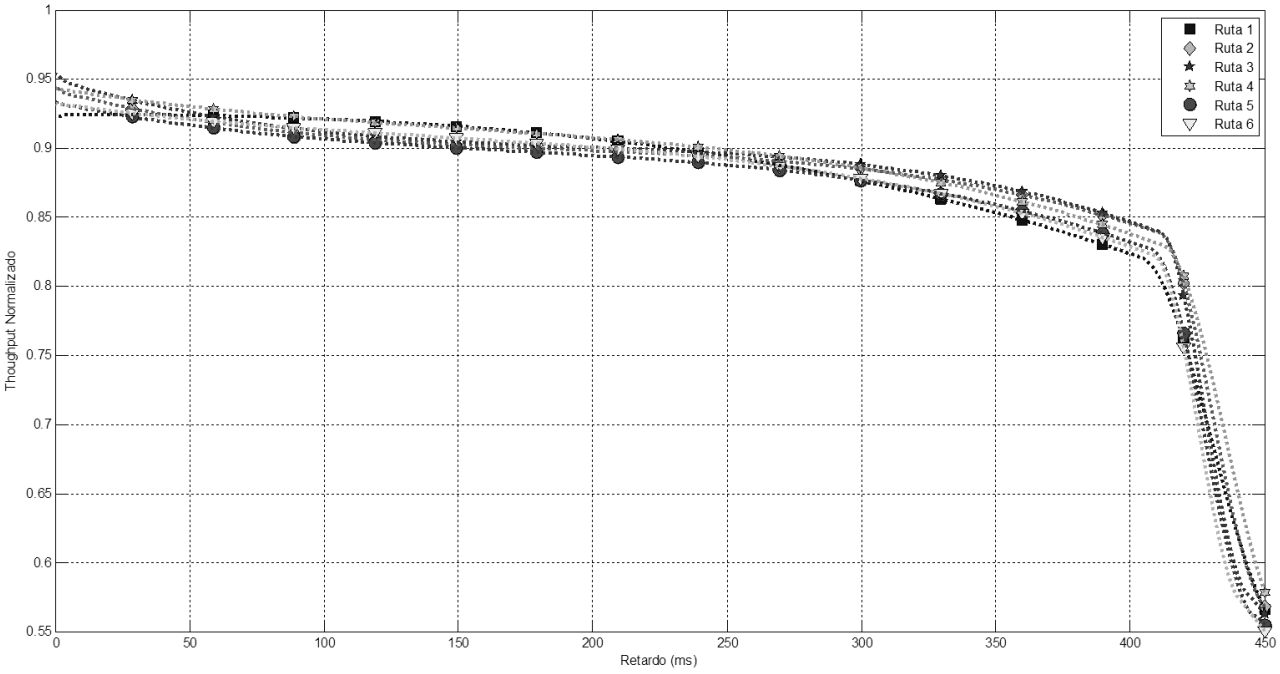

Figura 58. Desempeño de algunas trayectorias con base al retardo y el caudal eficaz normalizado.

Al igual que el retardo la perdida de paquetes tiene una relación inversa con el caudal eficaz, a mayores pérdidas el caudal eficaz disminuye como se ilustra en la Figura 59.

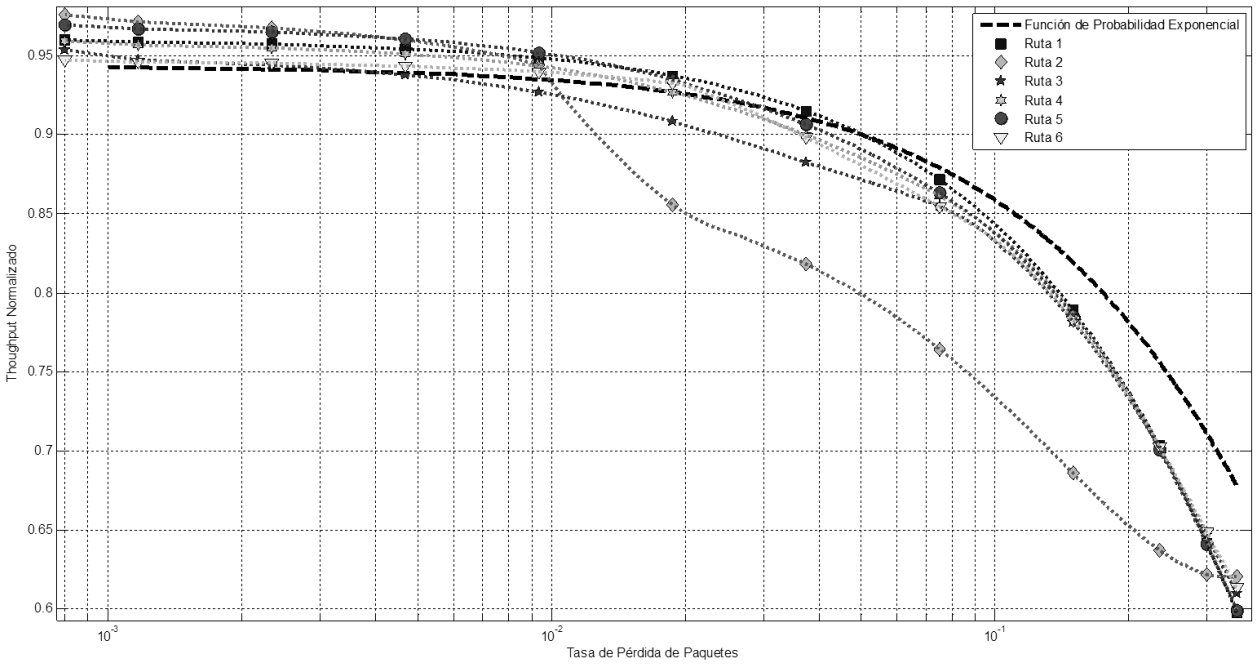

Figura 59. Desempeño de algunas trayectorias con base al caudal eficaz normalizado y la pérdida de paquetes.

Con base a las Figuras 57, 58 y 58, la trayectoria 3 a pesar que tiene menor rendimiento del caudal eficaz entre las trayectorias. Esta cuenta con mejor desempeño en retardo, además la pérdida de paquetes se mantiene dentro de la restricción.

Con esto se obtiene un buen acercamiento al comportamiento esperado, que cumple ambos requerimientos de retardo y pérdidas. A pesar que otras rutas cuentan con mejor desempeño del caudal eficaz tienen mayores pérdidas de paquetes cuando tienen bajo retardo, esto hace que violen la restricción de pérdidas y se descarte la ruta.

Esta página se ha dejado intencionalmente en blanco.