UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO

MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

DISEÑO DE UN MODELO PARA LA DISTRIBUCIÓN DE TRABAJOS BIGDATA PARA CÓMPUTO DE ALTO DESEMPEÑO ENTRE CENTROS DE DATOS BASADOS EN REDES DEFINIDAS POR SOFTWARE

T E S I S

QUE PARA OBTENER EL GRADO DE
DOCTOR EN CIENCIAS

PRESENTA

JOSÉ ELENO LOZANO RIZK

ENSENADA, BAJA CALIFORNIA, MÉXICO JUNIO, 2019

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO
UNIDAD ENSENADA

Diseño de un modelo para la distribución de trabajos Big Data para cómputo de alto desempeño entre centros de datos basados en redes definidas por software

TESIS

Que para obtener el grado de doctor en ciencias presenta:

José Eleno Lozano Rizk

Aprobada por:

Dr. Juan Iván Nieto Hipólito
Director de tesis

Dr. Raúl Rivera Rodríguez
Co-Director de tesis

Dra. Mabel Vazquez Briseño
Miembro del comité

Dra. Maria de los Angeles Cosio León
Miembro del comité

Dr. Juan Carlos Chimal Eguía
Miembro del comité

Ensenada Baja California, México. Junio 2019

Resumen

En los últimos años, el cómputo intensivo de datos ha evolucionado como una necesidad en la investigación científica, lo que ha provocado un crecimiento exponencial de volúmenes de datos almacenados en centros de datos distribuidos geográficamente.

Los científicos han diseñado aplicaciones de cómputo de alto desempeño y (HPC, High Performance Computing por sus siglas en inglés) y las de BigData, para procesar, analizar y transferir estos volúmenes de datos entre centros de datos distribuidos buscando que sea en el menor tiempo posible, por lo que es necesario proporcionarles de calidad de servicio (QoS, Quality of Service por sus siglas en inglés) con requerimientos específicos para cada tipo de aplicación.

Por otra parte, los centros de datos definidos por software, y en especial la red definida por software (SDN, Software Defined Network por sus siglas en inglés), que es uno de sus componentes principales, están siendo utilizados para hospedar aplicaciones HPC buscando cumplir con sus requerimientos específicos de calidad de servicio de manera dinámica y automatizada.

En un esquema de centro de datos distribuido geográficamente, la SDN se convierte en un factor de suma importancia que impacta directamente en el desempeño de aplicaciones que se ejecuten entre dichos centros de datos, especificamente, en el tiempo de terminación de sus trabajos que involucran procesos de transferencia de datos entre nodos de cómputo hospedados en cada uno de los centros de datos.

En la transferencia de datos de aplicaciones HPC-BigData ejecutándose entre centros de datos, la selección de trayectorias para redireccionamiento de flujos de datos es típicamente llevado a cabo por el controlador, el cual utiliza el algoritmo de la ruta más corta o en algunos casos, solo considera el ancho de banda que requieren dichas aplicaciones. En aplicaciones HPC-BigData, este esquema puede afectar el

tiempo de terminación de sus trabajos debido a que no se consideran otros factores en la red como el retardo punto a punto, la fluctuación del retardo, el porcentaje de pérdida de paquetes y métricas adicionales que pueden utilizarse para proporcionar QoS en la red de comunicaciones.

Como parte de esta investigación, se realizó un estudio sobre trabajos que abordan el tema de QoS en la SDN, identificando tres componentes principales:

- Si requieren de modificar o programar un módulo adicional en el controlador SDN.
- Su método para selección de trayectorias factibles (los parámetros de red que utiliza).
- Su arquitectura, si solo trabajan en un dominio con un solo controlador o si es un esquema inter-dominio, es decir, entre centros de datos distribuidos cada uno con su controlador SDN.

Como resultado del estudio, se identificaron una serie de oportunidades y retos para proporcionar de QoS a aplicaciones HPC-BigData que usan SDN.

Tomando en consideración las oportunidades y retos identificados en la presente investigación, se diseño una solución denominada QoSComm, que se define como una estrategia de asignación de flujos de datos basados en los requerimientos de QoS de las aplicaciones HPC-BigData que estén ejecutándose entre centros de datos distribuidos con el objetivo de minimizar su tiempo de ejecución.

QoSComm opera en dos fases: i) en la primer fase se calculan las trayectorias óptimas para cada centro de datos utilizando un método de optimización que considera cuatro parámetros de red: el ancho de banda disponible, el retardo punto a punto, las fluctuaciones y el porcentaje de pérdida de paquetes; ii) en la segunda fase se realiza la configuración de sus switches OpenFlow de manera dinámica, usando las trayectorias generadas en la primer fase.

En la presente investigación, se desarrolló un modelo de simulación para experimentación y pruebas utilizando el simulador SDN Mininet, donde se efectuaron dos clases de pruebas: el desempeño de la red y el desempeño de aplicaciones HPC. En esta última fase, se realizó la configuración de un cluster de cómputo para la ejecución de aplicaciones MPI (Message Passing Interface por su término en inglés). En

ambas pruebas se comparó el redireccionamiento de flujos de datos tradicional del controlador SDN contra QoSComm. Los resultados de la simulación mostraron que QoSComm mejoró el tiempo de terminación de aplicaciones HPC en un promedio de 35 %.

Palabras clave: Calidad de Servicio, Redes Definidas por Software, Cómputo de Alto Desempeño, BigData, Centros de Datos.

Resumen aprobado por:

Dr. Juán Iván Nieto Hipólito, Director de tesis

Dr. Raúl Rivera Rodríguez, Co-Director de tesis

Agradecimientos

Quiero agradecer a mis directores de tesis, al Dr. Juan Iván Nieto Hipólito y al Dr. Raúl Rivera Rodríguez, por darme la oportunidad de continuar con mis estudios de posgrado y aceptarme para realizar esta tesis bajo su dirección. Su apoyo, consejos, confianza, y sobre todo, el tiempo y dedicación que me brindaron han sido factores que considero muy importantes en mi formación académica y científica.

También, agradezco a los miembros de mi comité, a la Dra. María de los Ángeles Cosio León, a la Dra. Mabel Vazquez Briseño y al Dr. Juan Carlos Chimal Eguía, por sus comentarios, consejos, tiempo y dedicación durante mi desarrollo académico.

Agradezco a la UABC y al CICESE, por la celebración de convenios institucionales que permiten a su personal continuar con sus estudios de posgrado, propiciando y motivando el desarrollo académico y profesional del mismo.

Y un agradecimiento a mi familia, a mis padres, a mis hermanos y en especial a mi esposa, que gracias a su apoyo incondicional me fue posible continuar con mis estudios de doctorado.

Y a mis hijos, que son el motor de mi vida.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Planteamiento del problema

En los últimos años, el almacenamiento, transferencia, y principalmente el análisis intensivo de un gran volumen de datos (desde los Gigabytes hasta los Terabytes) se ha convertido en una necesidad en la investigación científica, lo que ha provocado un crecimiento exponencial de volúmenes de datos almacenados en centros de datos distribuidos. Para procesar y analizar estos grandes conjuntos de datos entre centros de datos distribuidos, los científicos han diseñado y desarrollado aplicaciones que se pueden clasificar como de HPC y del tipo BigData.

Las aplicaciones HPC y BigData que se ejecutan entre centros de datos requieren capacidades de cómputo específicas, como tiempo de CPU, capacidad de almacenamiento, RAM, entre otros, con la finalidad de poder cumplir con el procesamiento y análisis de datos en el menor tiempo posible. Sin embargo, cuando los trabajos de las aplicaciones requieren de transferir datos entre nodos de procesamiento ubicados en centros de datos distribuidos, la red de comunicaciones se convierte en un factor crítico que puede influir en los tiempos de terminación de dichos trabajos.

En algunos de los algoritmos de distribución de trabajos para el procesamiento de aplicaciones HPC entre centros de datos conectados por una red definida por software, se considera el ancho de banda como el factor principal a tomar en cuenta para la distribución, buscando obtener el menor tiempo posible en la terminación de dichos trabajos. Esta consideración en los algoritmos de distribución puede funcionar

para el procesamiento de trabajos en ciertas aplicaciones que involucren consultas a bases de datos relacionales y no-relacionales, sin embargo, para trabajos relacionados con aplicaciones de telefonía IP, video streaming, videovigilancia e inclusive trabajos BigData intensivos y de Cómputo de Alto Desempeño (HPC-BigData) [4] pudieran no verse beneficiados en su desempeño al no considerar políticas de calidad de servicio en el controlador de la red definida por software.

Además, dentro de los trabajos de investigación en SDN, uno de los puntos que se considera importante es el *control dirigido por usuario* [10], donde la intención es que las aplicaciones puedan interactuar con el controlador de la SDN y puedan solicitar servicios *bajo demanda* en la red, menciona por ejemplo, el caso de aplicaciones HPC-BigData del tipo MapReduce [11] donde estas pudieran solicitar garantía de ancho de banda (o servicios en la red) con la finalidad de mejorar su desempeño en la fase donde se mezclan los datos que son obtenidos de los diferentes nodos de procesamiento del cluster MapReduce.

De acuerdo a lo mencionado anteriormente, se formulan las siguientes preguntas de investigación:

1. ¿Qué parámetros de QoS en la SDN pueden contribuir a mejorar el desempeño de aplicaciones HPC-BigData?

2. ¿Qué modelo de QoS se puede proponer en la red definida por software para el procesamiento y análisis de aplicaciones del tipo HPC-BigData?

3. ¿Cómo incorporar QoS en el proceso de distribución de trabajos en aplicaciones HPC-BigData de una forma no intrusiva?

La presente investigación se enfoca en como minimizar el tiempo de terminación de aplicaciones HPC-BigData tomando en cuenta los aspectos de conectividad. Se busca diseñar un modelo intermedio entre este tipo de aplicaciones y el escenario de conectividad entre centros de datos, de esta forma se considera como caso de investigación los parámetros de calidad de servicio de una red definida por software para la distribución de trabajos HPC-BigData con la finalidad de disminuir su tiempo de terminación. Para el caso de esta investigación se consideran aplicaciones HPC que utilizan MPI y las de BigData que hacen uso de algoritmos MapReduce, tal como se ilustra en la Figura 1.1.

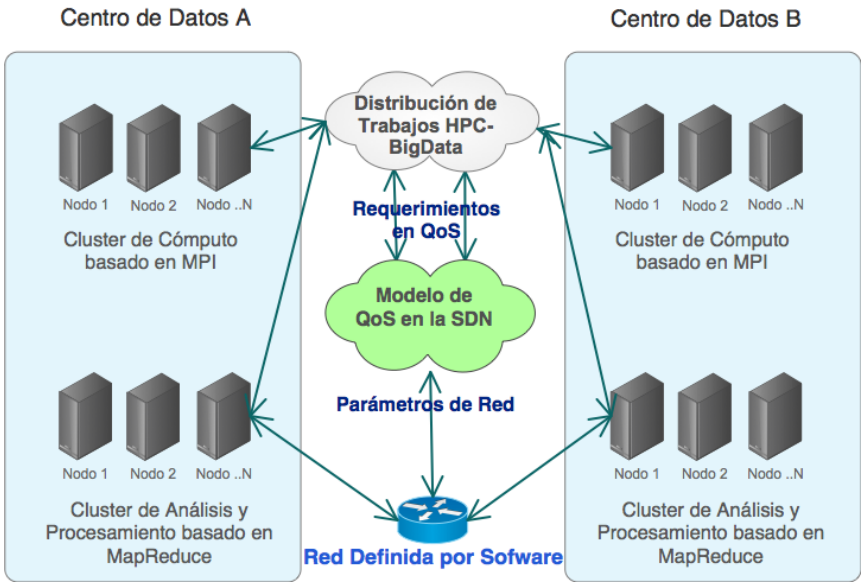

Figura 1.1: Representación del problema

1.2. Justificación

Dentro de las aplicaciones HPC se encuentran las del tipo MPI, que es un estándar utilizado para el desarrollo de aplicaciones de cómputo paralelo para la comunicación entre servidores o nodos de cómputo [12]. MPI es muy utilizado para el desarrollo de aplicaciones científicas de gran escala, y uno de los factores principales a mejorar es en la comunicación colectiva en MPI [13], es decir, a la comunicación entre más de dos procesos MPI de manera simultánea.

Referente al tema de aplicaciones HPC-BigData, como se mencionó anteriormente, una de las tecnologías que es muy utilizada para el procesamiento y análisis de grandes volúmenes de datos para cómputo de alto desempeño es el paradigma de computación llamado MapReduce. Este paradigma de computación es usado en universidades y centros de investigación para el análisis y procesamiento de datos científicos de diversas áreas como bioinformática, oceanografía, ciencias de la tierra, ciencias de la computación, entre muchas otras. También es muy utilizado por las grandes empresas como Google, Yahoo!, Facebook, entre otras, para analizar y procesar los datos que son generados en sus aplicaciones web por millones de usuarios.

Una de las implementaciones muy populares de MapReduce, es Apache Hadoop

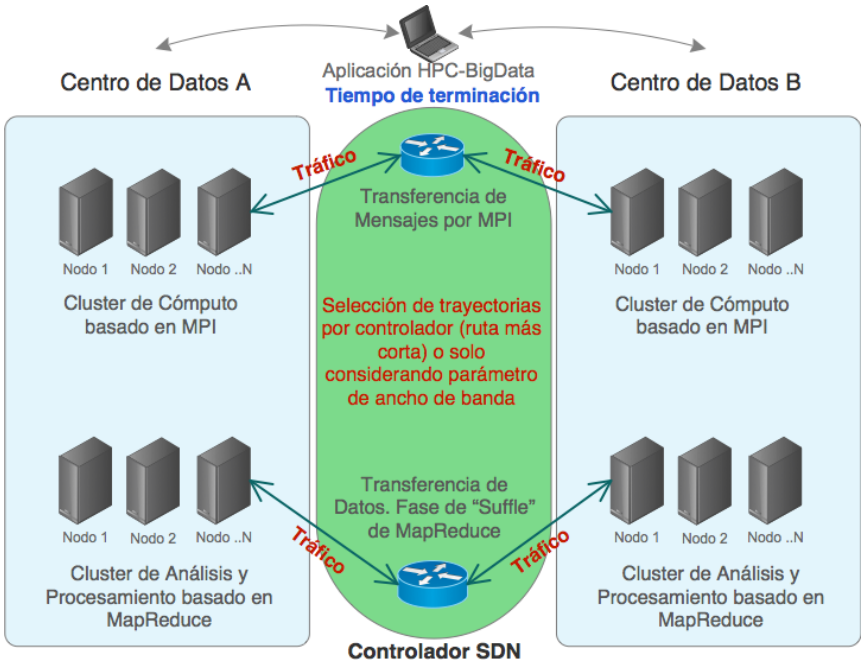

Figura 1.2: Distribución de trabajos HPC-BigData del tipo MPI y MapReduce

[14]. Apache Hadoop, es comúnmente utilizado en cluster locales, es decir, en un solo cluster hospedado dentro del mismo centro de datos, sin embargo, hoy en día, además de utilizarse en un solo cluster, existe la necesidad de poder distribuir trabajos MapReduce entre centros de datos con la finalidad de evitar el *costo* de transferir y centralizar grandes volúmenes de datos. Algunas iniciativas o proyectos como G-Hadoop [15] que se basa en reemplazar el sistema de archivos nativo de Hadoop por el sistema de archivos Gfarm y utiliza el mismo proceso de distribución de tareas del cluster, además utiliza redes tradicionales para conectar clusters distribuidos. BASS [2] es un planificador de tareas que combina Hadoop con la red definida por software y se limita a considerar solo la capacidad de control de ancho de banda de SDN en la planificación de tareas de Hadoop.

La Figura 1.2 muestra a las aplicaciones HPC-BigData transfiriendo datos entre centros de datos distribuidos conectados por una red definida por software donde utiliza el método tradicional del controlador SDN (ruta más corta) o solo considerando el control de ancho de banda para la distribución de trabajos lo que puede impactar su tiempo de terminación debido a la congestión de tráfico que se puede

generar por la saturación del recurso o por no considerar otros parámetros como retardo, fluctuaciones, porcentaje de pérdida de paquetes, por mencionar algunos.

En el presente trabajo se investigará sobre la distribución de trabajos HPC-BigData usando el caso de estudio de aplicaciones MPI y MapReduce que se ejecutan entre centros de datos conectados por una red definida por software que además de considerar la capacidad de control de ancho de banda, también considere otros parámetros de QoS como disponibilidad del servicio, latencia, fluctuaciones (jitter, por su término en inglés), por mencionar algunos, en el controlador de SDN de acuerdo al requerimiento de las aplicaciones.

1.3. Objetivo

El objetivo principal de esta investigación es diseñar un modelo para mejorar la distribución de trabajos en aplicaciones HPC-BigData entre centros de datos basados en redes definidas por software que considere parámetros de calidad de servicio (QoS) que contribuyan a disminuir el tiempo de terminación de dichos trabajos.

Objetivos específicos:

1. Identificar los parámetros de QoS de la SDN que pueden ser utilizados para disminuir el tiempo de terminación de trabajos de aplicaciones del tipo HPC-BigData

2. Diseñar un modelo para proporcionar QoS a aplicaciones del tipo HPC-BigData basado en SDN.

3. Diseñar un modelo para la distribución de trabajos HPC-BigData entre centros de datos distribuidos conectados por una red definida por software, que utilice el modelo de QoS.

1.4. Metodología

La metodología a seguir se describe a continuación:

1. Búsqueda de fabricantes que estén impulsando el desarrollo de SDN. Realizar una investigación sobre proyectos relacionados a SDN donde participan dichos fabricantes como es el caso de OpenDayLight, OpenFlow, Mininet, por mencionar algunos.

2. Realizar una revisión de artículos publicados sobre las redes definidas por software, su desarrollo y aplicación, así como un análisis y estudio de calidad de servicio en SDN para identificar el área de oportunidad objeto de la presente investigación.

3. Plantear un modelo de QoS en la red definida por software enfocado a aplicaciones del tipo HPC-BigData de acuerdo con la revisión de trabajos efectuados.

4. Revisar de artículos publicados en revistas arbitradas, congresos y bibliotecas privadas sobre el caso de estudio de MPI y MapReduce en aplicaciones de cómputo de alto desempeño. Realizar un estudio sobre su funcionamiento, operación y su modelo de programación, así como su modelo de distribución de trabajos de forma local y distribuida para determinar los criterios del modelos de simulación en la presente investigación.

5. Desarrollar un análisis de los algoritmos de distribución a nivel de aplicación HPC entre centros de datos conectados por una SDN, para identificar los parámetros de red que serán utilizados para la distribución de trabajos de dichas aplicaciones.

6. Realizar un análisis de ambientes de simulación de redes SDN para diseñar el modelo de simulación y experimentación.

7. Revisar métodos para selección de trayectorias y redireccionamiento de flujos de datos en una red definida por software para determinar el algoritmo de selección de trayectorias de acuerdo a requerimientos de QoS de las aplicaciones.

8. Diseñar un modelo para simular la ejecución de trabajos HPC entre centros de datos conectados por una red definida por software tomando en cuenta parámetros de QoS para la ejecución de experimentos.

9. Análizar y comparar resultados. Implementar la simulación del modelo generando tráfico de acuerdo al tipo de aplicación HPC-BigData. Comparar el modelo propuesto contra otros modelos de los diferentes algoritmos para el redireccionamiento de flujos de datos.

1.5. Organización de la tesis

- Capítulo 1. En este capítulo se presenta el planteamiento del problema, justificación, el objetivo general y los específicos así como la metodología empleada para la solución del problema.
- Capítulo 2. Presenta una descripción de los componentes de los centros de datos definidos por software haciendo énfasis en la red definida por software.
- Capítulo 3. Se presenta una descripción sobre el tipo de aplicaciones de cómputo de altas prestaciones objeto de la presente investigación.
- Capítulo 4. Presenta un revisión de trabajos sobre calidad de servicio en las redes definidas por software y al tipo de aplicaciones al que va enfocado.
- Capítulo 5. Presenta la propuesta de modelo de calidad de servicio para aplicaciones HPC entre centros de datos distribuidos habilitados por una red definida por software, el diseño y simulación del modelo, así como los experimentos que fueron llevados a cabo y el análisis de resultados.
- Capítulo 6. Presenta las conclusiones del trabajo de investigación así como el planteamiento de líneas de acción futuras.

Capítulo 2

Centros de Datos Definidos por Software

Los centros de datos tradicionales han sido utilizados para hospedar infraestructura de HPC. Los recursos computacionales utilizados en los clusters de cómputo provienen de una gran variedad de fabricantes. Este esquema requiere que los administradores de dicha infraestructura realicen de manera manual varias reglas de configuración o la aplicación de políticas de QoS en cada uno de los dispositivos con el objetivo de poder cumplir con los requerimientos de cómputo, almacenamiento, red y seguridad para cada aplicación HPC. Este proceso puede afectar el desempeño de dichas aplicaciones, además de presentarse una asignación de recursos computacionales desbalanceada o que no cumpla con las políticas de QoS requeridas para cada aplicación.

En los últimos años, los centros de datos definidos por software han surgido como una solución para automatizar la administración y el auto-provisionamiento de recursos de cómputo, almacenamiento, red y seguridad de manera dinámica, de acuerdo a las configuraciones particulares y requerimientos de cada una de las aplicaciones que se ejecutan en estos centros de datos.

El centro de datos definido por software es una arquitectura que integra cuatro componentes principales: i) almacenamiento, ii) cómputo, iii) redes y iiii) seguridad. Los citados componentes son abstractos y definidos por software para que funcionen como recursos que puedan ser solicitados o programados de manera dinámica por

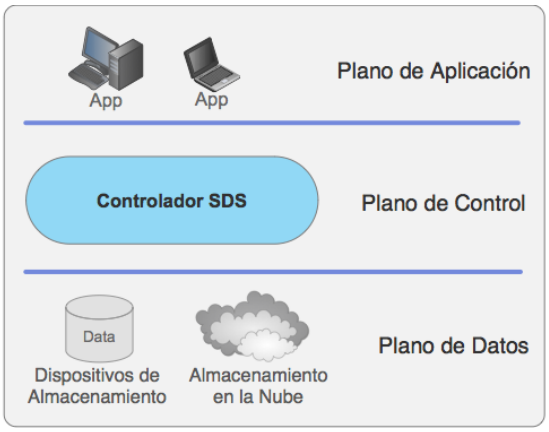

Figura 2.1: Vista general de la arquitectura de almacenamiento definido por software

las mismas aplicaciones. Dichos componentes pueden ser identificados como: Almacenamiento Definido por Software (Software Defined Storage, SDS, por su término en inglés), Cómputo Definido por Software (Software Defined Compute, SDC, por su término en inglés), Seguridad Definida por Software (Software Defined Security, SDSec, por su término en inglés) y Redes Definidas por Software (Software Defined Network, SDN, por su término en inglés). En la siguiente sección se describe de manera general cada uno de estos componentes.

2.1. Almacenamiento definido por software

SDS facilita y simplifica la abstracción de recursos y dispositivos de almacenamiento para cumplir con los requerimientos de las aplicaciones. SDS separa el plano de datos y el plano de control. El plano de datos (también conocido como capa de infraestructura), refiere al uso de dispositivos de almacenamiento de diversos fabricantes y modelos, tanto físicos como virtuales. En el plano de control, se definen las políticas para proveer de manera automática a las distintas solicitudes de las aplicaciones a través de una API (Application Programming Interface por su término en inglés). La Figura 2.1 muestra la arquitectura general de los SDS.

Dentro de las principales funcionalidades de los SDS se encuentra la automatización, la programabilidad, las trayectorias virtualizadas para datos, la escalabilidad y la transparencia [16]. Una de las características principales de SDS es la habilidad

de controlar y administrar las políticas de cada dispositivo en el sistema, ya que con dicho control es posible mantener los niveles de QoS requeridos por cada aplicación.

2.2. Cómputo definido por software

SDC está basado en la virtualización de servidores. La virtualización de servidores puede ser definido como un tecnología basada en software que permite la ejecución de distintos sistemas operativos al mismo tiempo (máquinas virtuales), dentro de un servidor físico [17]. La virtualización permite a múltiples aplicaciones (hospedadas en máquinas virtuales o contenedores) que sean ejecutadas en un mismo servidor físico mientras que se les provee de recursos de cómputo de manera aislada. Existen dos tipos de virtualización, que se describen a continuación:

1. En la virtualización a nivel de hardware, un sistema operativo ejecuta un proceso llamado hipervisor, el cual es capaz de comunicarse directamente con el kernel del host. Encima del hipervisor reside la capa de hardware virtual, y luego reside el kernel para el sistema operativo de la máquina virtual [18].

2. La tecnología de virtualización ligera ha surgido como una alternativa a la virtualización basada en hipervisores [19]. A este tipo de virtualización también se le conoce como virtualización basada en contenedores. Estos contenedores crean abstracciones de procesos -invitados- directamente sin la necesidad de un sistema opererativo -invitado- reduciendo la sobrecarga que puede existir al crear una nueva máquina virtual con un sistema operativo -invitado- para cada aplicación [20].

La Figura 2.2 muestra los componentes principales de SDC, en donde ambas tecnologías son administradas por el controlador SDC con la finalidad de habilitar la orquestación del servicio de acuerdo a requerimientos de las aplicaciones.

En SDC no solo se contempla la virtualización de servidores de cómputo y de recursos del sistema operativo, sino que también considera la abstracción de recursos de cómputo a través de las API, permitiendo que la infraestructura de SDC sea requerida por las aplicaciones HPC de manera dinámica y automatizada.

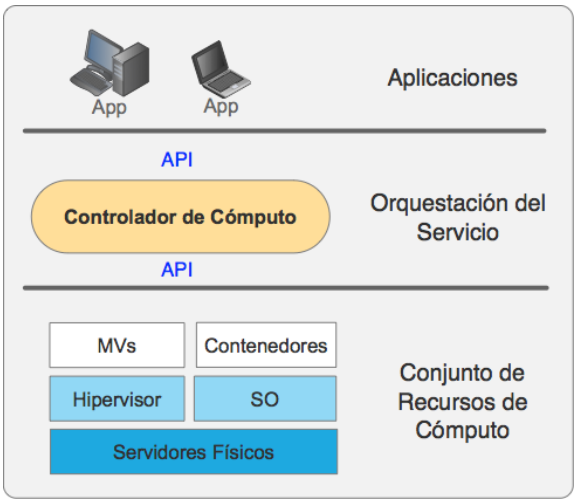

Figura 2.2: Componentes de SDC

2.3. Seguridad definida por software

SDSec proporciona una solución flexible y centralizada al abstraer los mecanismos de seguridad de la capa de hardware a una capa de software [21]. SDSec separa el redireccionamiento y procesamiento del plano de datos, del plano de control, virtualizando las funciones de seguridad. Uno de sus principales objetivos es que desde un solo punto de control, se puedan crear políticas de seguridad con la finalidad de proteger a los elementos virtuales que interactuan en el plano de datos, como son el almacenamiento, la red y los servidores.

Además, SDSec puede facilitar el desarrollo de políticas de seguridad que cubran varios elementos en la infraestructura de TI, como puede ser el respaldo seguro de aplicaciones catalogadas con datos sensibles, transferir flujos de datos utilizando trayectorias seguras, recuperación de máquinas virtuales y contenedores en caso de fallas, entre otros. SDSec puede proporcionar un solo punto de control para administrar estos elementos en un centro de datos definido por software, como se ilustra en la Figura 2.3.

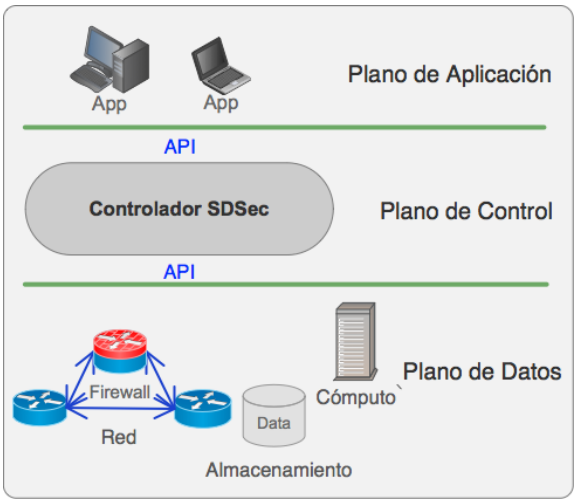

Figura 2.3: Elementos de SDSec

2.4. Redes definidas por software

En las redes de comunicaciones, el flujo de los paquetes de datos que constituyen una comunicación entre dos dispositivos conectados por una red, ha sido controlado por mucho tiempo por protocolos estándar. Cuando un paquete de datos llega a un dispositivo de red, los paquetes son conmutados basados en reglas de redireccionamiento estándar, los cuales ni el administrador de la red ni la aplicación tienen todo el control sobre ellos. Aunque estos modelos de redireccionamiento han escalado para el propósito general de Internet, algunas aplicaciones y administradores de la red requieren contar con un control más granular sobre el tratamiento del flujo de datos de una aplicación en específico. Este paradigma de proporcionar una API que permita que las aplicaciones y los sistemas que administran la red puedan controlar por programación y de manera dinámica la forma en que fluyen los paquetes, es conocido como Redes Definidas por Software o SDN por su término al inglés [22].

En consecuencia, SDN ofrece la posibilidad de diseñar aplicaciones científicas que interactúen con el controlador de la red solicitando el redireccionamiento de sus flujos de datos a nodos de red que cumplan sus requerimientos en ancho de banda, políticas de calidad de servicio, seguridad, por mencionar algunos, con la finalidad de mejorar sus tiempos de ejecución o respuesta.

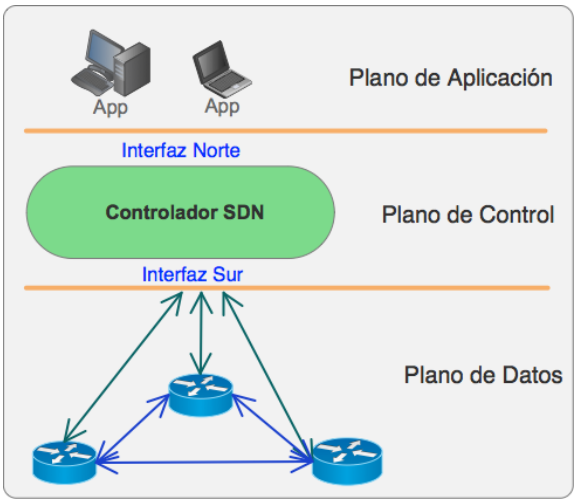

Figura 2.4: Vista simplificada de la arquitectura de la SDN

2.4.1. Arquitectura de las redes definidas por software

SDN es una nueva arquitectura de red que separa la lógica de control de la red (plano de control) de los enrutadores y switches (plano de datos) que administran el tráfico en la red. Al separar los planos de control y de datos, los switches se convierten en elementos de reenvío y la lógica de control es implementada en un controlador centralizado, lo que simplifica la reconfiguración de la red. La Figura 2.4 representa la arquitectura general de la Red Definida por Software.

El controlador de la SDN se puede comunicar con los switches (en el plano de datos) a través de una API que forma parte de la Interfaz Sur (SBI: Southbound Interface en inglés) y un buen ejemplo de ello es OpenFlow [23], un estándar que ha sido diseñado para SDN y que actualmente ya está siendo implementado en una gran variedad de redes (sobre todo para investigación) y en equipos de comunicaciones. El controlador de la red puede ofrecer una API en la Interfaz Norte (NBI: Northbound Interface en inglés) para los desarrolladores de aplicaciones donde les permite enviar instrucciones a los elementos de reenvío. Los conceptos de SBI y de NBI se definen en la siguiente sección. En el plano de aplicación o gestión se encuentran las aplicaciones que se comunican a través de la NBI para implementar el control de la red y la lógica de la operación. En este plano se encuentran aplicaciones de redes como enrutadores, balanceadores de carga, firewalls, monitoreo, por mencionar algunas, e

Tabla 2.1: Listado de controladores SDN y características generales

Característica	**ODL**[1]	**FL**[2]	**Ryu**	**Beacon**	**POX**
Soporte OpenFlow	Si	Si	Si	Si	Si
Virtualización. Mininet y Open vSwitch	Si	Si	Si	Si	Si
Lenguaje de desarrollo	Java	Java	Python	Java	Python
Proporciona REST API	Si	Si	Si	No	No
Código abierto	Si	Si	Si	Si	Si

[1] OpenDayLight.
[2] Floodlight.

inclusive también se pueden encontrar aplicaciones externas o de negocio (business applications en inglés).

2.4.2. Controladores de redes definidas por software

Los controladores SDN son responsables de la gestión del redireccionamiento de flujos de datos en los switches en las redes de comunicaciones. En general, los controladores SDN cuentan con una Interfaz Sur, que proporciona una API que permite establecer comunicación con los switches (o elementos de reenvío de datos) y un ejemplo de ello es el estandar OpenFlow [23]. También cuentan con una Interfaz Norte, que proporciona comunicación desde el plano de aplicación por medio de un servicio REST API (transferencia de estado representacional) que permite a las aplicaciones poder interactuar con los servicios que ofrece el controlador.

La Tabla 2.1 presenta un resumen de controladores SDN basado en [24] y se consideraron aquellas características que fueron objeto de la presente investigación. Se puede decir que el controlador SDN se encarga de dirigir el comportamiento general de la red, incluso a partir de los requerimientos de las aplicaciones tal como se ilustra en la Figura 2.5. Considerando este punto, el controlador que es utilizado en la presente investigación es OpenDayLight [25] ya que proporciona una REST API que de acuerdo a su documentación permite la configuración de reglas de flujo en los switches por medio del protocolo OpenFlow, para el redirecconamiento de flujos de datos en la SDN.

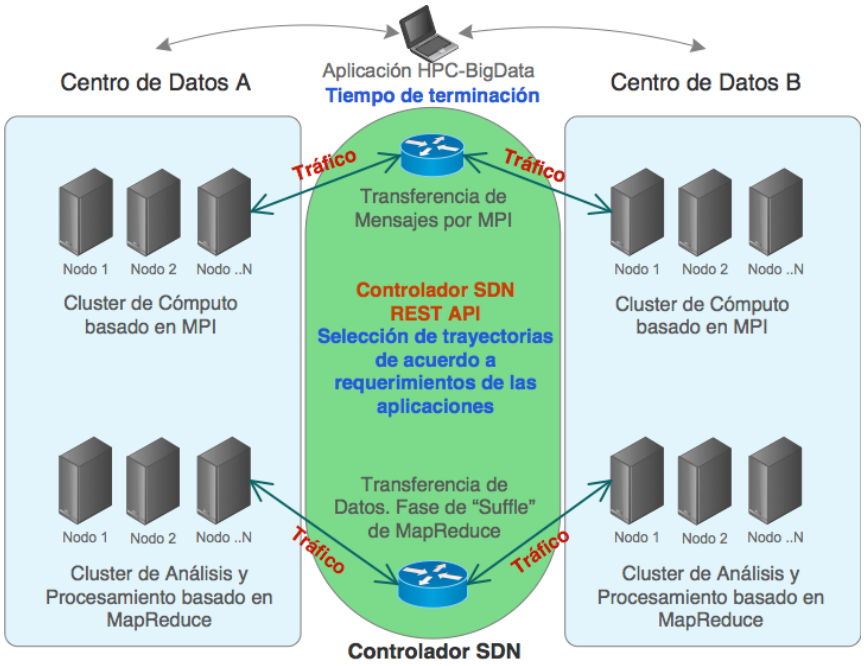

Figura 2.5: Selección de trayectorias por medio de la REST API del Controlador SDN

Como se puede observar en la Figura 2.5, las aplicaciones HPC-BigData, en específico los trabajos MPI y MapReduce respectivamente, que requieren de procesar y analizar información entre centros de datos distribuidos, pueden solicitarle al controlador de la SDN el redireccionamiento de flujos de datos por trayectorias que cumplan con sus requerimientos de red, como la calidad de servicio. Dicha solicitud, en la presente investigación, se propone que sea a través de la REST API del controlador, ya que permite obtener la topología de red y en un proceso externo al controlador, realizar el cálculo de trayectorias óptimas de acuerdo a los parámetros de calidad de servicio que requiera cada aplicación. Una vez que se obtengan las trayectorias óptimas (resultantes), se procede a realizar la configuración de dichas trayectorias también a través de dicha API, tal como se muestra en el diagrama de la Figura 2.6.

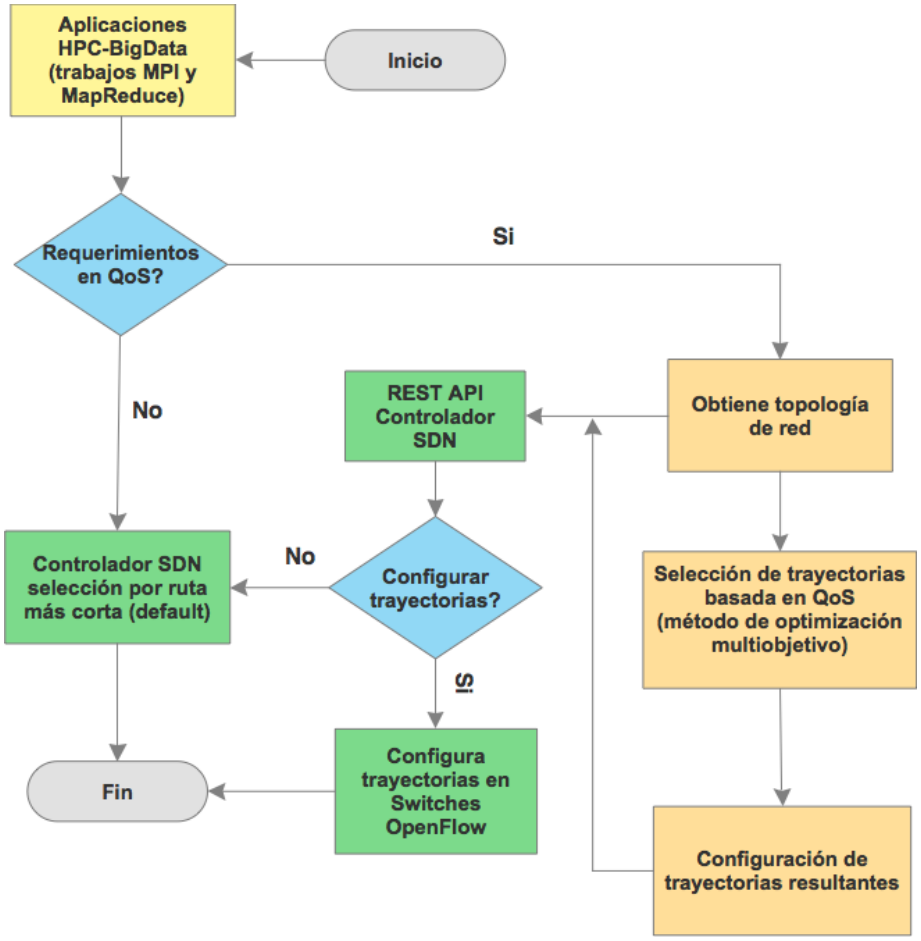

Figura 2.6: Diagrama de flujo para selección de trayectorias de acuerdo a requerimientos de QoS

Capítulo 3

Aplicaciones de Cómputo de Alto Desempeño

El cómputo de alto desempeño es utilizado para resolver grandes problemas en la ciencia e ingeniería, proporcionando capacidades en recursos computacionales para mejorar el desempeño de aplicaciones del tipo HPC. En relación a dichas aplicaciones, desde el punto de vista computacional y como parte de esta investigación, se identificaron aquellas en donde la red de comunicaciones puede impactar su desempeño, dividiéndose en:

1. Aplicaciones secuenciales, las que hacen uso intensivo de la red de comunicaciones para transferir datos entre clústers o centros de datos distribuidos geográficamente.

2. Aplicaciones de cómputo paralelo, las que utilizan Interfaz de Paso de Mensajes (MPI: Message Passing Interface en inglés) que hacen uso intensivo de la red de comunicaciones para transferir mensajes de datos entre nodos distribuidos.

3. Aplicaciones BigData con algoritmos MapReduce, que hacen uso intensivo de la red de comunicaciones para transferir datos entre clusters o centros de datos distribuidos geográficamente.

Asimismo, desde el punto de vista de red y para fines de la presente investigación, este tipo de aplicaciones se pueden subdividir de acuerdo a sus requerimientos de calidad de servicio en la red:

1. Aplicaciones del tipo "mejor esfuerzo" en las que con el monitoreo, ajuste o administración del ancho de banda es suficiente para su ejecución, se enfocan principalmente en la transferencia masiva de datos.

2. Aplicaciones del tipo "HPC" en las que es necesario *garantizar* servicios en la red, tomando en cuenta algunos parámetros de calidad de servicio como: ancho de banda, latencia, fluctuación (jitter en inglés), disponibilidad del servicio, tasa de transmisión efectiva, tasa de pérdida de paquetes y RTT (Round-Trip Delay Time en inglés), por mencionar algunos.

La presente investigación está orientada al estudio de aplicaciones HPC que requieran de ciertos servicios en la red para mejorar su desempeño. Para el caso de esta investigación se consideran aplicaciones HPC que utilizan MPI y las de BigData que hacen uso de algoritmos MapReduce.

3.1. Aplicaciones de cómputo paralelo basadas en MPI

Como se mencionó anteriormente, MPI es un estándar utilizado para el desarrollo de aplicaciones de cómputo paralelo para la comunicación entre servidores o nodos de cómputo [12]. Aunque MPI es usado para el desarrollo de aplicaciones científicas de gran escala, uno de los factores principales a mejorar es en la comunicación colectiva en MPI [13], es decir, a la comunicación entre más de dos procesos MPI de manera simultánea.

Dentro de las operaciones colectivas de MPI se encuentra las de computación colectiva, como MPIReduce y MPIALLreduce, las cuales se utilizan para reducir y combinar valores entre los procesos MPI por lo que requieren de transferir datos entre ellos. El desempeño de este tipo de operaciones pudiera impactar la velocidad de respuesta de las aplicaciones paralelas a gran escala, si las aplicaciones MPI que están realizando estas operaciones pudieran conocer el estado de la SDN, consultando parámetros como ancho de banda, latencia, o inclusive solicitando calidad de servicio de acuerdo a sus requerimientos, lo que se traduciría en menores tiempos de terminación de sus trabajos.

3.2. Aplicaciones HPC-BigData: algoritmos MapReduce

El análisis de HPC-BigData actualmente es considerado uno de los principales retos en ciertas áreas de investigación. MapReduce, que fue desarrollado por Google, es uno de los paradigmas de la computación que es utilizado para procesar trabajos de análisis de BigData en clusters de manera local o inclusive distribuida, ya que permiten obtener resultados en un menor tiempo comparado con un procesamiento tradicional (secuencial). El modelo de programación de MapReduce está basado en dos funciones principales: mapear (map) y reducir (reduce). De manera muy general, la función de mapear procesa pares de llaves/valores para generar un conjunto intermedio de pares de llaves/valores y la función reducir combina todos los valores intermedios que sean iguales. Una de las implementaciones más utilizadas de MapReduce es Hadoop [26], desarrollado por Apache Foundation, la cual está basada en Java y utiliza HDFS (Hadoop Distributed File System, en inglés). En este contexto, aplicaciones tan complejas como las de HPC-BigData pueden ser procesadas por medio de MapReduce en un ambiente distribuido. Existen ya trabajos que han investigado el uso de tareas MapReduce entre centros de datos conectados por una SDN considerando parámetros de la red, como el ancho de banda en su proceso de planificación de trabajos/tareas (scheduling por su término en inglés) [27], o en algunos casos, la latencia. En la siguiente sección se estudian este tipo de aplicaciones y su interacción con la SDN.

3.3. Aplicaciones HPC-BigData y su Interacción con la red definida por software

En esta sección se responde a las siguientes preguntas: ¿Las aplicaciones HPC-BigData que interactúan con SDN mejoran su desempeño?, ¿Consideran parámetros de QoS en la planificación de sus tareas? La respuesta a estas preguntas permitirá identificar las oportunidades en la interacción de la SDN y las aplicaciones HPC y BigData.

A continuación, se describen una serie trabajos de investigación que buscan op-

timizar el tiempo de ejecución de aplicaciones que utilizan MPI ejecutándose en clusters interconectados por SDNs.

GSBT [28], es un algoritmo que tiene como objetivo minimizar el número de saltos de la ruta más larga en el árbol de reducción. El beneficio que puede obtener este algoritmo depende del número de enlaces disponibles por el switch de más alto nivel y el grado de concurrencia en el envío y recepción de paquetes. Si se encuentran muchos hosts conectados en los switches de más bajo nivel, la congestión en el tráfico degradará el desempeño de la aplicación. Las aplicaciones MPI que utilicen este algoritmo, al iniciar su ejecución envían la dirección IP y MAC de cada proceso al controlador de la SDN y este se encarga de encontrar el camino más corto entre un par de hosts e instala los flujos de entrada en todos los switches, así como la construcción de los caminos más cortos y el árbol binomial para todos los "roots" de reducción. Después de esta operación, les retransmite los árboles de reducción a todos los hosts. Este algoritmo solo se enfoca en obtener las rutas disponibles en la SDN y calcula las rutas más cortas para utilizarlas en el árbol de reducción. En la evaluación del modelo concluyen que, si bien, en la comunicación de mensajes con tamaño menor a 68K se logra una mejora, en los que son mayores se afecta su desempeño debido a la congestión en la red.

En BLAR [1], realizaron un esquema de redireccionamiento de los paquetes de datos en la SDN de acuerdo a los requerimientos de las aplicaciones, enfocándose en dos parámetros de la red: ancho de banda y latencia, como se ilustra en la Figura 3.1. Si la aplicación solicita cierto ancho de banda, esta lo comunica al controlador de la SDN y este realiza los ajustes en los switches para redireccionar los flujos de datos de la aplicación en las rutas que cumplan con los requerimientos de ancho de banda. En las aplicaciones que requieran de cierta latencia, es el mismo proceso que el anterior, solo que se enfoca en redireccionar los flujos a las rutas que cumplan con los requerimientos en latencia. Para ciertos casos demostraron que el tiempo de terminación de las aplicaciones era menor si consideraban algunos de estos parámetros en el proceso de redirecionamiento de flujos en vez de utilizar el proceso de redireccionamiento de la ruta más corta, que es el algoritmo implementado en el controlador de la SDN. Si bien, esta investigación considera dos de los parámetros más importantes para aplicaciones de HPC [29], existieron casos en donde no se logró una mejora en el

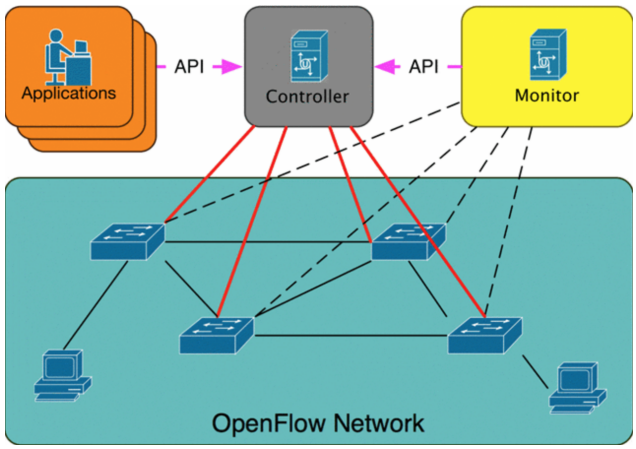

Figura 3.1: Estructura de una red basada en ancho de banda y latencia [1]

desempeño de la aplicación y al contrario afectó su desempeño.

En lo referente a las aplicaciones BigData, a continuación, se describen una serie de algoritmos y estrategias con aplicaciones MapReduce entre centros de datos conectados por una SDN.

BASS [2], es un planificador de tareas que utiliza a la SDN para obtener información sobre el ancho de banda de la red, con la finalidad de que los trabajos MapReduce en Hadoop puedan interactuar con la SDN, tal como se representa en la Figura 3.2. Su objetivo es utilizar a la SDN para administrar el ancho de banda y después BASS, dependiendo del tiempo de terminación de los trabajos, decide si asigna las tareas de manera "local" o de manera "remota" en clusters distribuidos. Lo más interesante de BASS es que verifica el ancho de banda de la red en el controlador de la SDN y lo clasifica como un parámetro a utilizar en la planificación de las tareas de la aplicación. De acuerdo a sus resultados, BASS puede mejorar el tiempo de ejecución de un trabajo MapReduce entre clusters distribuidos sí las condiciones de ancho de banda son las adecuadas, ya de que no ser así, se puede afectar su desempeño.

PYTHIA [3], se pudiera considerar como un sistema distribuido que se compone de dos partes principales: i) instrumentación del middleware de Hadoop (MapReduce), ii) una entidad que se encarga de manejar los eventos de transferencias futuras aleatorias de MapReduce. PYTHIA hace uso de la funcionalidad de la SDN para poder utilizar los recursos de la red para las transferencias aleatorias, que es uno de los

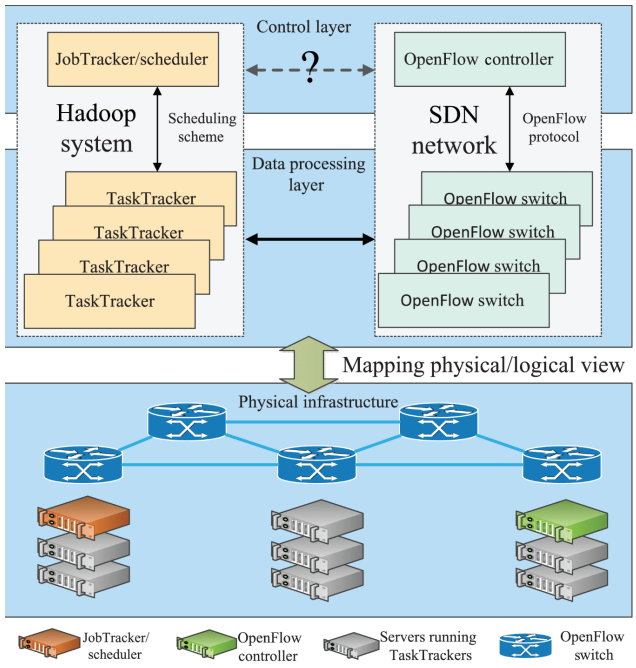

Figura 3.2: Arquitectura de BASS [2]

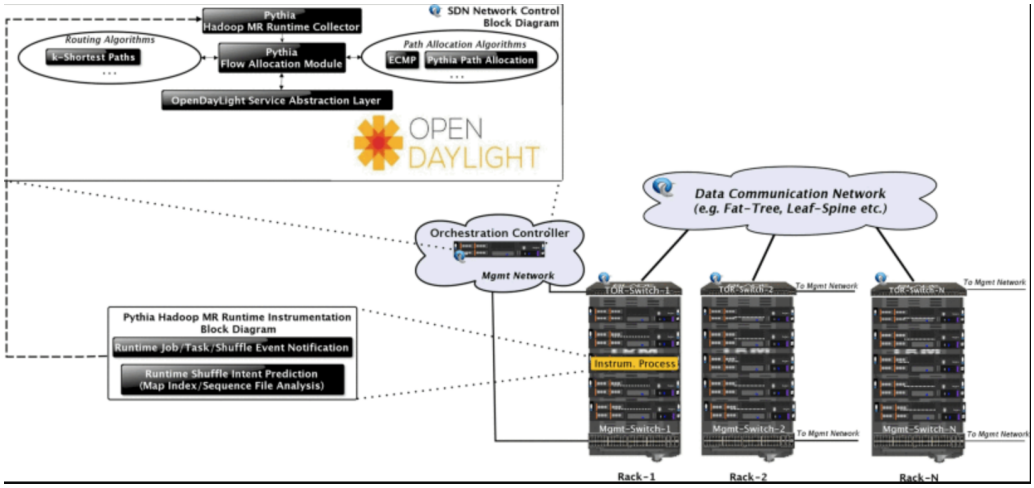

Figura 3.3: Arquitectura de PYTHIA [3]

procesos de MapReduce en donde se hace uso intensivo del canal de comunicaciones, como se representa en la Figura 3.3. El modelo de planificación de PYTHIA, se basa en obtener información sobre la topología física de la red, el uso del enlace de red y la intención de comunicación de la aplicación para calcular una asignación optimizada de flujos a rutas de red, de modo que se reducen los tiempos de las transferencias aleatorias de los trabajos MapReduce, mostrando una mejora en el desempeño de algunas aplicaciones; sin embargo, en algunos casos puede no mejorar el desempeño debido al tiempo que se tarda en responder entre eventos de mapeo y tareas de reducción propios de MapReduce, por lo que los autores lo consideran como trabajo futuro de investigación.

CLS [30], es una propuesta basada en un planificador Cross-Layer donde el planificador a nivel de aplicación para tareas/trabajos puede interactuar con el planificador de la SDN que se encarga de asignar las rutas de red. La intención de esta propuesta, es el colocar las tareas y una selección de rutas que, en conjunto, puedan alcanzar un alto rendimiento para la aplicación. El planificador a nivel de aplicación utiliza información de ancho de banda que obtiene del planificador de la SDN, y de esta manera distribuye las tareas en los servidores. Los resultados obtenidos, indicaron una mejora en el desempeño de aplicaciones de Hadoop y de Storm [31].

La Tabla 3.1 muestra una comparativa de los trabajos de investigación objeto del presente estudio y su funcionalidad general. Dichos trabajos se enfocaron en la distribución de trabajos entre clusters o nubes distribuidas que utilizan redes definidas por software. En GSBT, su algoritmo solo se enfoca en obtener las rutas más cortas en la SDN para utilizarlas en el árbol de reducción, sin considerar condiciones adversas de la red, como congestionamiento de tráfico, retardo y pérdida de paquetes. Los trabajos se enfocaron en optimizar el ancho de banda (BASS y CLS). En BLAR, se optimizó el ancho de banda y también la latencia de la SDN. Si bien, con solo considerar estos parámetros en sus procesos de planificación de tareas lograron una mejora en el desempeño de la aplicación, los resultados de algunas investigaciones muestran que existieron casos donde no se logró una mejora y al contrario, se afectó el desempeño de la misma, lo cual se puede deber a diversas condiciones en la red que no fueron parte de su estudio como congestionamiento de los enlaces, la pérdida de paquetes, el retardo, las fluctuaciones, la disponibilidad de servicio, la tasa de

Tabla 3.1: Aplicaciones HPC-BigData que interactuan con la SDN

Funcionalidad	**GSBT**	**BLAR**	**BASS**	**PYTHIA**	**CLS**
Aplicaciones consultan parámetros de la red SDN	Si	Si	Si	Si	Si
Utiliza SDN entre clusters o nubes distribuidas geográficamente	No	Si	Si	Si	Si
Planifica trabajos considerando el parámetro de ancho de banda en la SDN	Si	Si	Si	Si	Si
Mejora el desempeño de la aplicación si considera parámetros de la SDN	Si[1]	Si[1]	Si[1]	Si	Si
Planifica trabajos considerando políticas QoS en la SDN	No	No	No	No	No

[1] En algunos casos se afectó el desempeño de la aplicación.

transmisión efectiva, por mencionar algunos. Por lo tanto, la oportunidad en estos trabajos de investigación sería considerar este tipo de parámetros de red para proporcionar calidad de servicio a dichas aplicaciones, como se representa en la Figura 3.4.

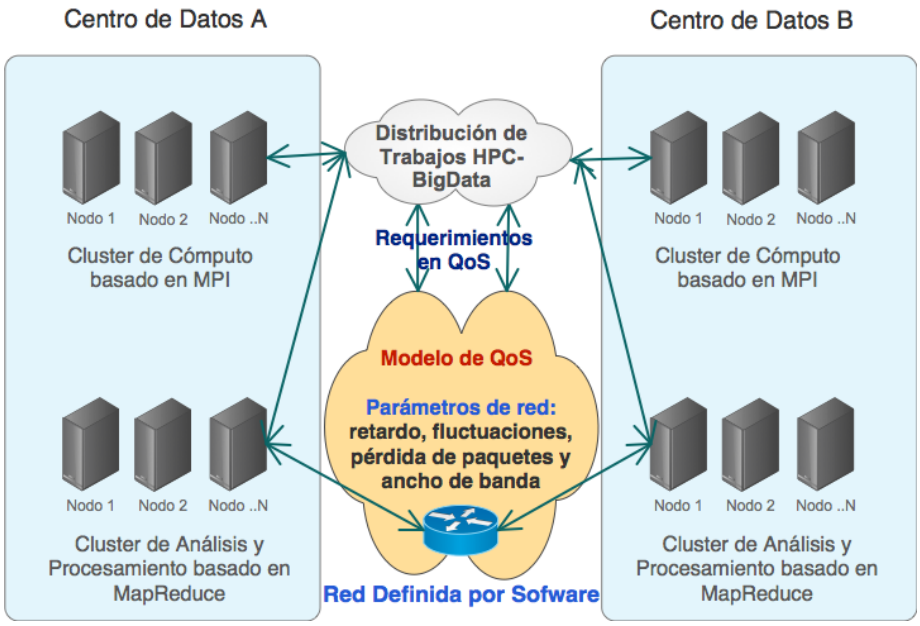

Figura 3.4: Parámetros de QoS para aplicaciones HPC-BigData

Capítulo 4

Calidad de Servicio en Redes Definidas por Software

La calidad de servicio, desde el contexto de redes, se puede definir como la habilidad de proporcionar un servicio [4] bajo ciertas condiciones fijadas por el cliente. Los parámetros más utilizados para garantizar QoS, son ancho de banda, latencia, porcentaje de pérdida de paquetes y control de congestión [29]. También existen otros parámetros de QoS como: disponibilidad del servicio, tasa de transmisión efectiva, fluctuaciones, RTT (del inglés Round-Trip Delay Time) y PSNR (del inglés Peak Signal-to-Noise Ratio). El parámetro de QoS a garantizar depende del tipo de aplicación. A continuación, se mencionan algunas de las propuestas donde se modela la arquitectura de una SDN con un enfoque de QoS, además de identificar los elementos que interactúan en la arquitectura, los parámetros de red utilizados en el algoritmo para el enrutamiento y si la propuestra trabaja en un solo dominio o de manera de inter-dominio.

4.1. Revisión de trabajos en QoS

Q-Ctrl [4], puede controlar la calidad de servicio enfocada a una infraestructura de nube computacional dentro de un mismo dominio o centro de datos, donde propone una arquitectura que contempla un manejador de QoS, un monitor de topología de red y un inyector de flujo basado en QoS. El controlador de QoS puede operar de dos

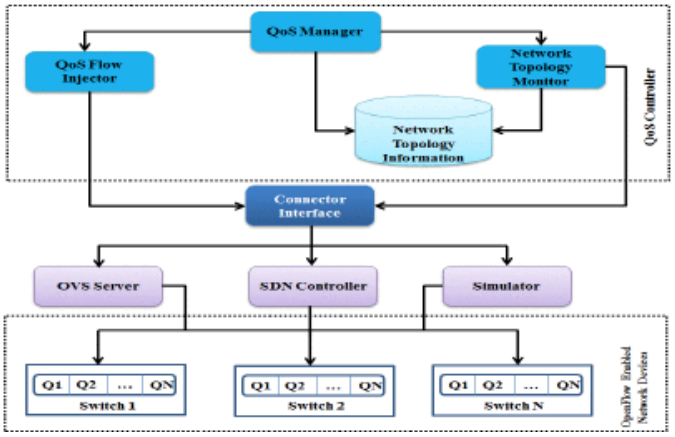

Figura 4.1: Arquitectura de referencia de QoS Controller [4]

formas: modo directo y modo controlador. En el modo directo, el controlador de QoS se comunica con cada switch e inyecta los flujos de calidad de servicio en los switches OpenFlow. En el modo de controlador, el controlador de QoS se comunica con el controlador de la red y éste se encarga de inyectar los flujos de calidad de servicio a los switches OpenFlow. La Figura 4.1 muestra su arquitertura general. El algoritmo de enrutamiento para QoS se basa en controlar el ancho de banda. Los resultados muestran mejoras para aplicaciones de multimedia y web, pero no consideraron en su experimento aplicaciones de cómputo de altas prestaciones.

En OpenQoS [5], se diseña un controlador que permite la calidad de servicio 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. La Figura 4.2 muestra su arquitectura. El tráfico lo clasifican en flujos de datos los cuales utilizan el algoritmo de la ruta más corta y en flujos multimedia que utilizan el algoritmo de enrutamiento de calidad de servicio dinámico. Este algoritmo de enrutamiento de calidad de servicio dinámico, lo presentan como un problema del algoritmo de la ruta más corta con restricciones (Constrained Shortest Path, CSP, por sus siglas en inglés), donde se asocia minimizar la función de coste sujeto a la variación del retraso del enlace, el cual debe ser menor o igual al retraso especificado como requerimiento de la aplicación. Dado que el problema CSP es conocido como NP-Completo, utilizan el algoritmo LARAC (Lagrangian Relaxation Based Agregated Cost por sus siglas en inglés) para el cálculo de las rutas. Además,

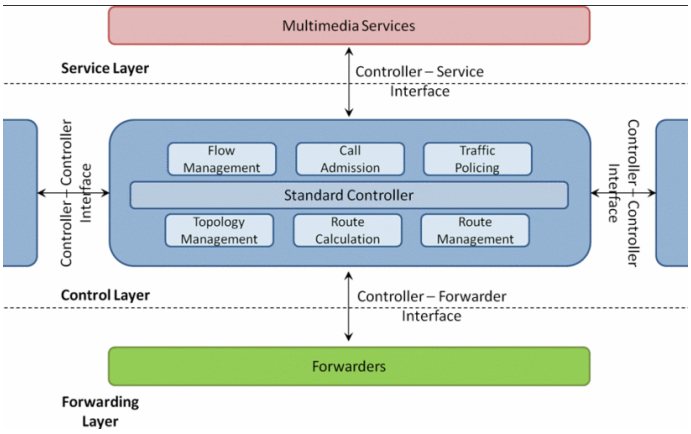

Figura 4.2: Arquitectura de referencia de OpenQoS Controller [5]

es comparado con otros modelos de QoS de las redes convencionales como IntServ y DiffServ. Los resultados muestran mejoras para aplicaciones multimedia del tipo streaming.

En QoSRP [6], presentan un caso de estudio para asignar QoS en aplicaciones de VoIP y Video. El modelo de QoS lo orientaron para los casos de emergencia donde falla el enlace de comunicaciones principal y se requiere utilizar un enlace de respaldo, en donde por lo general, el ancho de banda disponible en esta situación es limitado respecto al ancho de banda principal. La Figura 4.3 muestra la topología desarrollada para el caso de pruebas. En caso de presentarse una contingencia, priorizan el tráfico VoIP de usuarios "premium" controlando el ancho de banda. Para poder realizar el control de ancho de banda, se marcan los paquetes con cierta prioridad y se adapta el enrutamiento de manera dinámica a colas de salida con alta prioridad. Se realizaron mediciones a tres métricas: pérdida de paquetes, latencia y jitter, para determinar el funcionamiento de la calidad de servicio para este tipo de aplicaciones. Los experimentos realizados fueron dentro de un mismo dominio y utilizando un solo controlador. Sus resultados muestran mejoras para aplicaciones de video y VoIP.

En ERSDN [32], presentan una propuesta de enrutamiento explícito en SDN, en el cual su objetivo principal es evitar el problema de escalabilidad que se puede presentar en el controlador de la SDN, al reducir el número de eventos de red que son procesados en el plano de control. En lo referente a calidad de servicio se basa en VSDN [7]. Esta propuesta se compara con IntServ, DifServ y MPLS, donde resalta

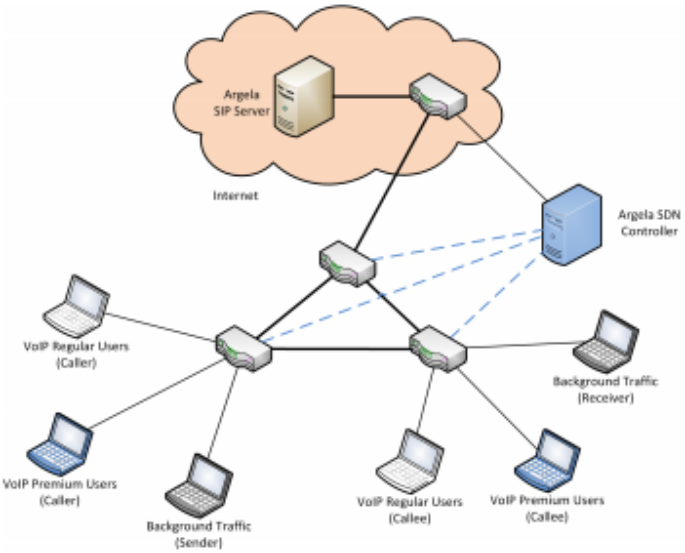

Figura 4.3: Topología desarrollada para control de ancho de banda [6]

sus diferencias al implementar calidad de servicio para la transmisión de video sobre SDN. En VSDN, se diseña una arquitectura dentro de un dominio para transmisión de video sobre SDN, tal como se ilustra en la Figura 4.4. Dicha arquitectura incluye un módulo de enrutamiento que contiene un monitor de topología, y este se encarga de verificar las condiciones de la red, tomando en cuenta parámetros como ancho de banda, fluctuaciones y retraso de los enlaces, y distribuye los paquetes de datos de acuerdo a algoritmos de enrutamiento basados en restricciones [33]. Sus resultados muestran mejoras para aplicaciones de video.

En QoSControl [8], proponen una arquitectura para QoS en SDN que contempla un monitor de recursos, un proceso para cálculo de rutas que cumplan con los requerimientos de calidad de servicio, un módulo para reservación de recursos y un proceso para control de llamadas de admisión, como se ilustra en la Figura 4.5. Dentro del proceso para realizar el cálculo de rutas se consideraron dos tipos de tráfico: "tráfico con prioridad" (con requerimientos estrictos en ancho de banda) y el tráfico del tipo "mejor esfuerzo". En lo que respecta al algoritmo de calidad de servicio, este se basa en controlar el ancho de banda utilizando la ruta más corta que satisfaga los requerimientos en ancho de banda de la aplicación y también selecciona el enlace que cuente con menos tráfico con la finalidad de balancear la carga de trabajo en la

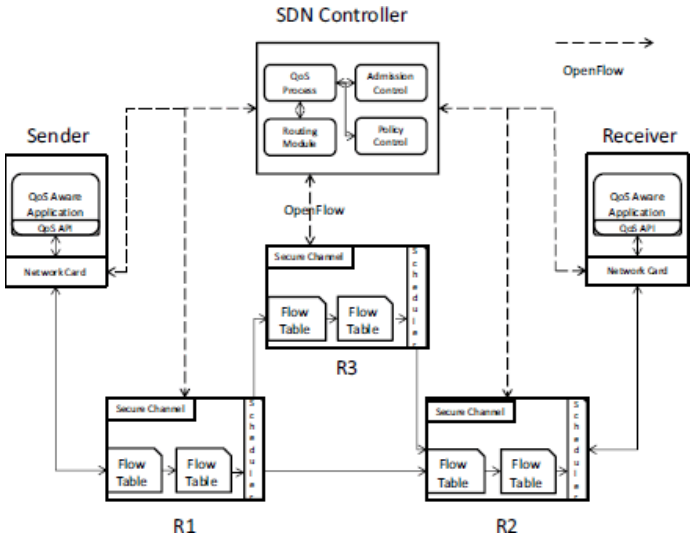

Figura 4.4: Vista general de VSDN [7]

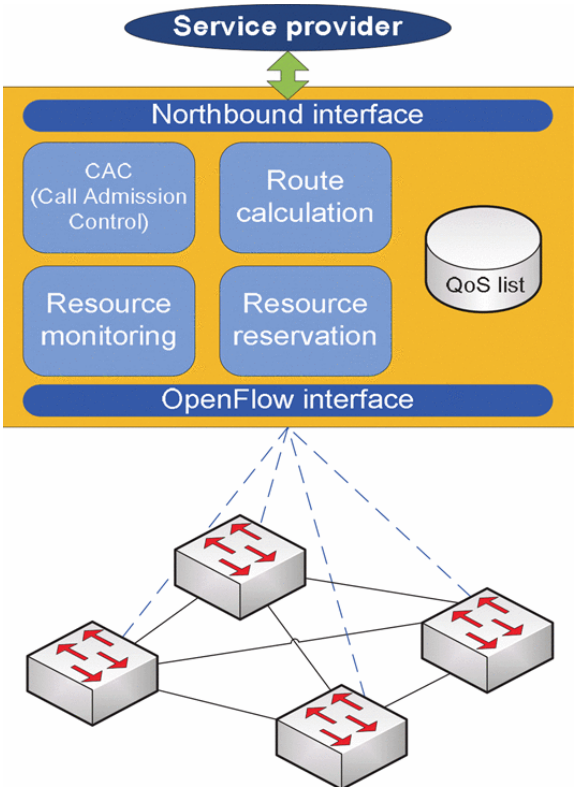

Figura 4.5: Arquitectura del sistema de control de QoS [8]

red. Los experimentos realizados fueron considerando un solo dominio o centro de datos, es decir, no consideraron la comunicación entre dos o más centros de datos. Sus resultados muestran mejoras para aplicaciones multimedia.

En CECT [34], proponen un esquema para minimizar la congestión de la red y reasignar los recursos de red de acuerdo a los requerimientos de las aplicaciones. El algoritmo de enrutamiento fue diseñado para un centro de datos definido por software enfocado a nubes computacionales, en donde la reasignación dinámica de flujos de datos en servicios como el movimiento de máquinas vituales de un servidor de procesamiento a otro, necesitan de garantías de calidad de servicio y éstas pueden generar una sobrecarga en el proceso de reasignación.

En AmoebaNet [35], los autores propusieron un servicio de red que garantice la calidad de servicio en redes de área local (LAN por su término en inglés) o redes tipo campus para manejo de grándes volúmenes de datos por medio de sistemas de reservación de trayectorias de redes de área amplia (WAN por su término en inglés). Algunos ejemplos de este tipo de redes especializadas son: ESNet (Energy Science Network), OSCARS (On-Demand Secure Circuits and Advance Reservation System) e I2 (Internet 2). AmoebaNet hace uso de una variante del algoritmo de la ruta más corta para calcular las trayectorias de red punto a punto, utilizando el parámetro de ancho de banda como una restricción.

4.1.1. Análisis de trabajos previos en QoS

La Tabla 4.1 presenta un resumen de las características de los algoritmos de decisión para la selección de trayectorias implementados en los trabajos de QoS consultados. Estas características fueron consideradas con la finalidad de poder determinar los parámetros de red utilizados en los algoritmos de selección de trayectorías. Derivado de dicha tabla y de acuerdo al estudio de las propuestas de calidad de servicio descritas anteriormente, se puede resumir que las propuestas existentes implementan un módulo adicional que interactúa con el controlador de la SDN para proporcionar calidad de servicio. El algoritmo de enrutamiento para calidad de servicio, en la mayoría de los casos, utiliza la ruta más corta basada en restricciones [5] y [7], en donde además de consultar la distancia de los nodos, en algunos casos también consideran el retardo que existe entre los mismos. El enfoque de la mayoría de los trabajos de

Tabla 4.1: Características de los algoritmos de decisión para la selección de trayectorias

Característica	**Q-Ctr [4]**	**Open QoS [5]**	**QoS Ctrl [6]**	**VSDN [7]**	**QoS CF [8]**
Considera la ruta más corta	No	Si	No	No	Si
Considera retardo punto a punto	No	No	Si	Si	No
Controla en ancho de banda	Si	No	Si	Si	Si
Algoritmo que considera hasta dos parámetros	No	Si[1]	No	Si[1]	Si
Algoritmo que considera más de dos parámetros	No	No	No	No	No

[1] Algoritmo de decisión de trayectorias basado en restricciones

calidad de servicio va dirigido hacia el redireccionamiento de los flujos para controlar el ancho de banda y mejorar la transmisión de aplicaciones multimedia (video streaming) e inclusive VoIP, pero en ninguno de ellos consideran otro tipo de aplicaciones como las de HPC-BigData, por lo que no está claro bajo qué circunstancias estos trabajos pueden beneficiar su desempeño.

Las aplicaciones HPC-BigData, dependiendo de su funcionalidad, características y complejidad en la solución de problemas para los cuales fueron diseñadas, pudieran requierir de un algoritmo de calidad de servicio multiobjetivo, que considere los parámetros de red como pueden ser ancho de banda, retardo punto a punto, pérdida de paquetes, fluctuaciones por mencionar algunos, además del diseño de estrategias que puedan aprovechar las ventajas que la SDN ofrece. En las propuestas consultadas, los experimentos realizados solo consideraron la calidad de servicio dentro de un solo dominio y con un controlador de la SDN. Ninguno de los casos consideró una propuesta de calidad de servicio entre dos o más dominios, es decir, bajo una arquitectura donde se contemplan aplicaciones HPC y BigData, ejecutándose entre

dos o más centros de datos distribuidos donde se requiere que el módulo de calidad de servicio establezca comunicación con los controladores de cada dominio que intervengan, así como la interacción con los proveedores de servicio de Internet (ISP, Internet Service Provider, por su término en inglés).

Cabe resaltar que el ancho de banda es el parámetro de la SDN que se utiliza en la mayoría de los trabajos consultados, pero no significa que controlar el ancho de banda sea igual a proporcionar calidad de servicio en la SDN.

4.2. QoS para aplicaciones HPC: Retos y oportunidades

En los trabajos consultados en la sección anterior, se puede resumir que la calidad de servicio puede contribuir a mejorar el desempeño de las aplicaciones HPC y BigData. Bajo esta premisa, se identifica como uno de los principales retos la incorporación de calidad de servicio en la SDN, específicamente en el proceso de selección de las rutas que mejor se adapten a los requerimientos de las aplicaciones. Este proceso de selección estaría basado en políticas de QoS definidas en un módulo adicional en el plano de aplicación o gestión que interactúe con el controlador de la SDN, de esta manera, el redireccionamiento de los flujos de datos se basaría en políticas de QoS en vez de solo utilizar la ruta más corta (que es el redireccionamiento tradicional del controlador de la red) o que solo considere algún parámetro de la red. Desde el punto de vista de las aplicaciones, dentro su proceso de planificación de tareas y/o trabajos, éstas deberán de considerar en su algoritmo de decisión, las políticas de QoS en la SDN para cada aplicación.

A continuación, se listan los principales retos que se identifican para proporcionar QoS en SDN para aplicaciones HPC-BigData:

1. Diseñar una clasificación de acuerdo al tipo de aplicación HPC-BigData y el nivel de prioridad de sus flujos de datos en la SDN. En [9], se realizó un estudio para clasificar las clases de QoS en redes de comunicaciones de acuerdo a las características de las aplicaciones y considerando los parámetros de retardo, fluctuaciones y pérdida de paquetes, como se muestra en la Tabla 4.2. De

Tabla 4.2: Clases de QoS [9]

Clase QoS	**Característica**	**Retardo**	**Fluctuaciones**	**Pérdida de paquetes**
0	Tiempo real, sensibles a fluctuaciones, altamente interactivo	100ms	50ms	1/1000
1	Tiempo real, sensibles a fluctuaciones, interactivo	400ms	50ms	1/1000
2	Transacciones de datos, altamente interactivo	100ms	N/A	1/1000
3	Transacciones de datos, interactivos	400ms	N/A	1/1000
4	Baja pérdida (transacciones cortas, lotes de datos, transmisión de video)	1s	N/A	1/1000
5	Aplicaciones tradicionales de redes IP por defecto)	N/A	N/A	N/A

acuerdo a los valores máximos de los parámetros de red, se propone clasificar al tipo de aplicaciones HPC-BigData de acuerdo a su funcionalidad, es decir, algunas de estas aplicaciones requieren transferencia masiva de datos y pudieran clasificarse como una clase de QoS nivel 0 o nivel 2, ya que puede existir una comunicación entre nodos de cómputo inter-dominio. Una vez clasificada la aplicación, se podría incorporar al controlador de la SDN que le asigne una mayor prioridad al flujo de datos de dicha aplicación, con la finalidad de que pueda realizar su transferencia de datos en el menor tiempo posible.

2. Establecer políticas de QoS que contemplen el nivel de prioridad de los flujos de datos y los parámetros de red requeridos por las aplicaciones, en el proceso de selección de rutas y redireccionamiento de flujos en la SDN. Dentro del proceso

de selección de rutas, es necesario definir un algoritmo de calidad de servicio multiobjetivo con la finalidad de considerar los parámetros de red más importantes que pueden afectar el desempeño de aplicaciones HPC y BigData, como pueden ser el ancho de banda, retraso, fluctuaciones y pérdida de paquetes.

3. Desarrollar un planificador inteligente que maneje dinámicamente las políticas de QoS definidas para aplicaciones HPC-BigData. Se puede diseñar un planificador inteligente que distribuya el flujo de datos de las aplicaciones a los nodos de red que satisfagan las políticas de QoS requeridas por las aplicaciones, pero que también pueda realizar ajustes en dicha distribución *al vuelo*, es decir, es necesario que el planificador cuente con información sobre el estado ejecución de la aplicación y pueda tomar decisiones ajustando de manera dinámica la política de QoS asignada previamente; es decir, se propone el diseño un algoritmo que pueda medir la calidad de experiencia (QoE, Quality of Experience en inglés) de la aplicación y realizar los ajustes necesarios en la calidad de servicio, con la finalidad mejorar su desempeño.

4. Diseñar una arquitectura de QoS que contemple a las aplicaciones HPC y BigData que se ejecutan entre centros de datos distribuidos (diferentes dominios o centros de datos). Bajo esta arquitectura, se plantea un modelo en donde hosts del centro de datos A requieren de transferir datos a hosts del centro de datos B y viceversa. Uno de los aspectos más importantes a considerar es la comunicación entre los controladores de la SDN en cada centro de datos para poder coordinar las políticas de QoS definidas para este tipo de aplicaciones, lo que puede implicar el diseño de algoritmos para manejo de multicontroladores o el diseño de estrategias para que permitan la coordinación de las políticas de QoS entre los distintos controladores de la SDN.

Cada uno de estos retos conlleva un proceso de análisis, estudio e inclusive el diseño de nuevos algoritmos, sin embargo, considerándolos en conjunto se propone una estrategia integral para proporcionar calidad de servicio con el objetivo de mejorar el desempeño de aplicaciones HPC y BigData, que son ejecutadas entre centros de datos conectados por una SDN, como se ilustra en la Figura 4.6.

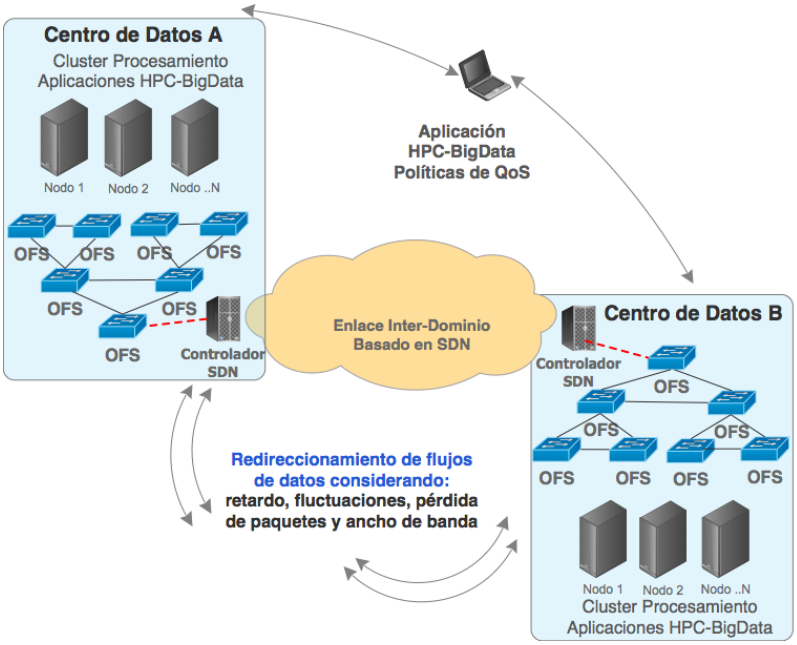

Figura 4.6: Esquema general de estrategia integral de QoS

Capítulo 5

QoSComm: Propuesta inter-dominio para la distribución de trabajos HPC-BigData

Considerando el diseño de un modelo de calidad de servicio que contemple a las aplicaciones HPC y BigData que se ejecutan entre centros de datos distribuidos (diferentes dominios o centros de datos) conectados por una SDN, se propone el diseño de una estrategia de asignación de flujos basados en calidad de servicio con la finalidad de mejorar el tiempo de terminación para este tipo de aplicaciones. A esta propuesta la denominamos QoSComm.

La Figura 5.1 muestra la vista general de QoSComm, donde el planificador de la aplicación envía una solicitud a QoSComm para obtener y configurar las trayectorias de red que cumplan con los requerimientos de las aplicaciones. Dichos requerimientos consideran cuatro parámetros de red: ancho de banda disponible, retraso, fluctuaciones y porcentaje de pérdida de paquetes.

Una vez que QoSComm obtiene y configura estas trayectorias en ambos centros de datos, el planificador de la aplicación envía el trabajo para que se ejecute en nodos de cómputo hospedados en centros de datos distribuidos y utilicen las trayectorias de red que cumplan con sus requerimientos previamente solicitados.

El objetivo principal de QoSComm es minimizar el tiempo de terminación de los trabajos de aplicaciones HPC-BigData tomando en consideración los parámetros de

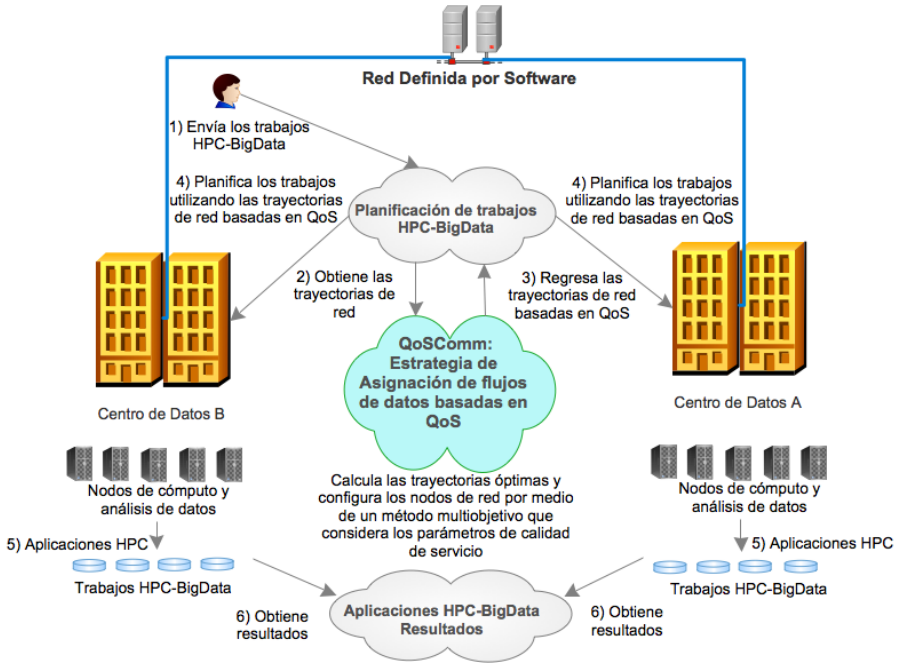

Figura 5.1: Vista general de QoSComm

red antes citados para proporcionar calidad de servicio en el proceso de asignación de flujos y así transferir sus datos entre los centros de datos distribuidos geográficamente. Con esto en mente, QoSComm selecciona las rutas óptimas que cuenten con el menor retardo punto a punto entre los nodos de cómputo distribuidos, por lo que esta propuesta va dirigida a la optimización de aplicaciones HPC sensibles al tiempo.

5.1. QoSComm: Diseño e implementación

QoSComm se puede definir como una estrategia de asignación de flujos de datos para aplicaciones HPC-BigData que se estén ejecutando entre centros de datos distribuidos basados en redes definidas por software. Su diseño consiste de dos funciones principales, como se muestra en la Figura 5.2.

1. Calcula las trayectorias de QoS de acuerdo a un algoritmo de decisión multiobjetivo basado en restricciones. Obtiene las trayectorias óptimas con el menor retardo punto a punto, y cumple con los requerimientos de la aplicación

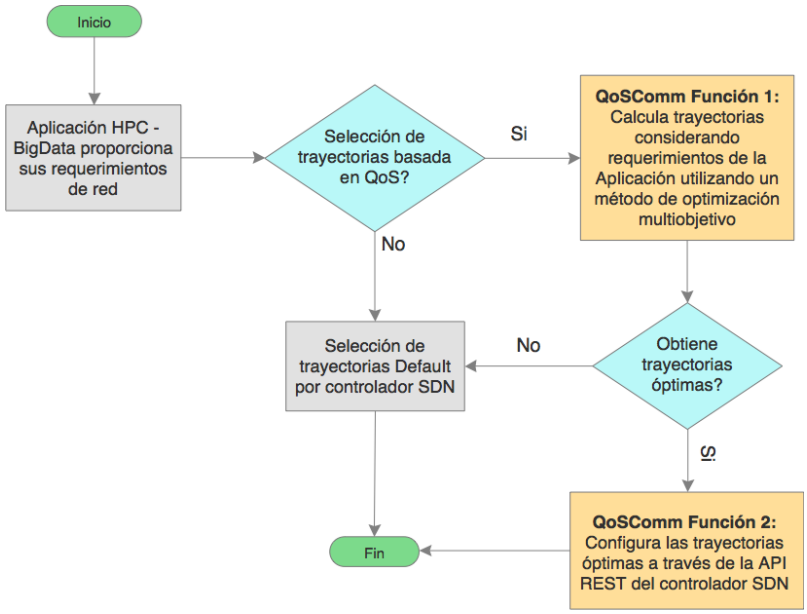

Figura 5.2: Proceso general de selección de trayectorias a través de QoSComm

considerando cuatro parámetros de red: ancho de banda disponible, retardo, fluctuaciones y porcentaje de pérdida de paquetes.

2. Configura la trayectoria óptima por medio del controlador SDN a cada uno de los switches OpenFlow. Se comunica con el controlador SDN para configurar una regla de flujo en cada uno de los switches OpenFlow (a nivel de puerto de entrada y salida). El controlador SDN instala una regla de flujo de datos en los switches que son especificados en la comunicación con la REST API (Representational State Transfer, por sus siglas en inglés) del controlador. La REST API permite la configuración dinámica de reglas de flujo de datos en los switches OF utilizando peticiones HTTP que son enviadas a un servicio web RestFul que proporciona el controlador SDN para consultar, crear, modificar y eliminar dichas reglas.

En un entorno de clusters de cómputo de alto desempeño, cuando el planificador de la aplicación HPC-BigData va a enviar trabajos a nodos de cómputo, en ese instante de tiempo, QoSComm obtiene y configura las trayectorias de red óptimas a los nodos de cómputo entre centros de datos conectados por una SDN.

5.1.1. Método de selección de trayectorias

Formalización del problema

En una topología de red típica, existen n número de trayectorias para transmitir datos de un nodo fuente a un nodo destino, lo que se conoce como trayectoria punto a punto, tal como se describe en [36], y se representa en la Ecuación (5.1):

$$P^{s,d} = \{p_1, p_2, ...p_n\} \tag{5.1}$$

Donde:
s = Nodo fuente
d = Nodo destino
P = Conjunto de trayectorias del nodo fuente s al nodo destino d
p = Trayectoria punto a punto $p \in P^{s,d}$

Cada trayectoria tiene distintos valores en sus parámetros de red. Estos parámetros sirven como métricas para obtener las condiciones de la trayectoria. Los parámetros de red considerados por las aplicaciones que requieren el uso intensivo de la red de comunicaciones son el retardo ($D(p)$), las fluctuaciones ($J(p)$), la pérdida de paquetes ($PLR(p)$) y el ancho de banda ($B(p)$). Al tomar en cuenta estos parámetros, se puede proporcionar calidad de servicio en la red. También se puede considerar el costo de la trayectoria ($Cost(p)$), y en conjunto estas condiciones de red para cada trayectoria se representan en la Ecuación (5.2).

$$p_i = [Cost(p_i), D(p_i), J(p_i), PLR(p_i), B(p_i)] \tag{5.2}$$

Donde:
i = $1, 2, 3, ..., n$
n = Total de trayectorias del nodo fuente al nodo destino

El enrutamento por defecto o tradicional, selecciona a la trayectoria tomando en cuenta solo el costo o la distancia entre los nodos de red. Esta métrica es la que utiliza el controlador SDN para seleccionar la ruta más corta ($\widetilde{p}$). Se representa en

la Ecuación (5.3) como el mínimo costo entre las n posible trayectorias.

$$\widetilde{p} = [min(Cost(P^{s,d}))] \tag{5.3}$$

En el algoritmo de enrutamiento tradicional, los parámetros que proveen las condiciones de la red no son considerados. Para cierto tipo de aplicaciones, el conjunto de estos parámetros deberían de considerarse para la selección de trayectorias para que puedan conseguir un mejor desempeño comparado con el obtenido al utilizar la ruta más corta. La trayectoria con las mejores condiciones ($\widehat{p_n}$) es la trayectoria que cuente con los valores de las métricas del conjunto de trayectorias disponibles, tal como se representa en la Ecuación (5.4).

$$\widehat{p_n} = [\widehat{D}(p_n), \widehat{J}(p_n), \widehat{PLR}(p_n), \widehat{B}(p_n)] \tag{5.4}$$

Donde:
$i = 1, 2, 3, ..., n$
n = Total de trayectorias del nodo fuente al nodo destino

La ruta más corta evaluada en la Ecuación (5.3) no siempre es la trayectoria con el menor costo en las métricas en uno o varios de sus parámetros. Se identifica a la trayectoria con las mejores condiciones en las métricas con el símbolo $\widehat{()}$. Debe de contener el valor mínimo de la suma del retardo punto a punto, las fluctuaciones, y la pérdida de paquetes entre las n trayectorias, representadas en las ecuaciones (5.5) (5.6) y (5.7). En el caso del ancho de banda disponible, se define como el valor mínimo entre las n trayectorias, considerando el escenario con las condiciones mínimas para asegurar la calidad de servicio requerida por la aplicación, y se representa en la Ecuación (5.8).

$$\widehat{D(p_i)} = [min(D(P^{s,d}))] \tag{5.5}$$

$$\widehat{J(p_i)} = [min(J(P^{s,d}))] \tag{5.6}$$

$$\widehat{PLR(p_i)} = [min(PLR(P^{s,d}))] \tag{5.7}$$

$$\widehat{B(p_i)} = [min(B(P^{s,d}))] \tag{5.8}$$

Donde:

$D(p_i)$ = Retardo en la trayectoria i

$J(p_i)$ = Fluctuaciones en la trayectoria i

$PLR(p_i)$ = Pérdida de paquetes en la trayectoria i

$B(p_i)$ = Ancho de banda disponible en la trayectoria i

$i = 1, 2, 3, ..., n$

n = Total de trayectorias del nodo fuente al nodo destino

La red proporciona servicios de conexión a un conjunto de aplicaciones, y cada una de ellas tiene sus propios requerimientos en parámetros de red con la finalidad de asegurar un desempeño adecuado. Este conjunto de aplicaciones se representa en la Ecuación (5.9) y los requerimientos en la Ecuación (5.10).

$$A = [\widehat{a_1}, \widehat{a_2}, \widehat{a_3}, ..., \widehat{a_n}] \tag{5.9}$$

$$\widehat{a_j} = [D(\widehat{a_j}), J(\widehat{a_j}), PLR(\widehat{a_j}), B(\widehat{a_j})] \tag{5.10}$$

Donde:

$j = 1, 2, 3, ..., m$

m = Total de aplicaciones

A = Conjunto de aplicaciones

$\widehat{a}$ = Requerimientos de parámetros de red de la aplicación

$D(\widehat{a_i})$ = Requerimiento de retardo por la aplicación

$J(\widehat{a_i})$ = Requerimiento de fluctuaciones de la aplicación

$PLR(\widehat{a_i})$ = Requerimientos de porcentaje de paquetes pérdidos de la aplicación

$B(\widehat{a_i})$ = Requerimientos en ancho de banda disponible

La ruta más corta, por lo regular, no proporciona la trayectoria que cumpla con los requerimientos de calidad de servicio de la aplicación, por lo que es necesario seleccionar dicha trayectoria tomando como base los requerimientos de dicha apli-

cación. Estos requirimientos son considerados como restricciones en el momento de seleccionar el conjunto de trayectorias óptimas que requiere la aplicacion. Como se muestra en las ecuaciones (5.11) (5.12) y (5.13), cada métrica tiene un límite máximo tolerable en los parámetros de retardo, fluctuaciones y porcentaje de pérdida de paquetes requerido por la aplicación. La Ecuación (5.14) representa el ancho de banda mínimo disponible y que cumple con los requerimientos de la aplicación. En la selección de trayectorias óptimas, cada uno de los parámetros de red debe de cumplir los requerimientos de la aplicación.

$$\widehat{D(p_i)} \leq D(\widehat{a_j}) \tag{5.11}$$

$$\widehat{J(p_i)} \leq J(\widehat{a_j}) \tag{5.12}$$

$$\widehat{PLR(p_i)} \leq PLR(\widehat{a_j}) \tag{5.13}$$

$$\widehat{B(a_j)} \leq B(\widehat{p_i}) \tag{5.14}$$

Donde:
$i = 1, 2, 3, ..., n$
n = Total de trayectorias del nodo fuente al nodo destino
$j = 1, 2, 3, ..., m$
m = Total de aplicaciones

En nuestro escenario, como considera un enfoque de redireccionamiento de datos inter-dominio el cual será utilizado solamente por las aplicaciones que requieran de QoS, se utiliza el método multiobjetivo restricción-ϵ para obtener el conjunto de trayectorias factibles, considerando los requerimientos de las aplicaciones como restricciones de los objetivos [37, 38]. Se propone utilizar el parámetro de retardo como el objetivo más importante sujeto a las restricciones de los otros objetivos. Se define la siguiente función objetivo, representada en las ecuaciones (5.15) y (5.16):

$$min(D(p_i), J(p_i), PLR(p_i), B(p_i)) \tag{5.15}$$

$s.a:$

$$f(x) = \begin{cases} \widehat{D(p_i)} \leq D(\widehat{a_j}) \\ \widehat{J(p_i)} \leq J(\widehat{a_j}) \\ \widehat{PLR(p_i)} \leq PLR(\widehat{a_j}) \\ \widehat{B(a_j)} \leq B(\widehat{p_i}) \end{cases} \tag{5.16}$$

Donde:

$i = 1, 2, 3, ..., n$
n = Total de trayectorias del nodo fuente al nodo destino
$j = 1, 2, 3, ..., m$
m = Total de aplicaciones

Como se mencionó anteriormente, el objetivo del método de restricción-ϵ es minimizar uno de los objetivos (considerado el más importante) y restringir el resto a un valor ϵ. La función general se representa en las ecuaciones (5.17) and (5.18):

$$min = f_j(x) \tag{5.17}$$

$s.a:$

$$f_i(x) \leq \epsilon_i \; i = 1, 2, 3, ..., M, i \neq j \tag{5.18}$$

Donde:

ϵ_i = Representa al límite superior del objetivo $_i$
M = Número de objetivos

De acuerdo al método de restricción-ϵ descrito anteriormente, del conjunto de trayectorias factibles, se obtiene la trayectoria óptima, que en este caso, es la que cuenta con menor retardo (objetivo más importante) y está sujeta a las restricciones de las fluctuaciones, porcentaje de pérdida de paquetes y ancho de banda disponible (resto de los objetivos), tal como se representa en las ecuaciones (5.19) y (5.20).

$$min = \widehat{D(p_i)} \tag{5.19}$$

$s.a$:

$$\begin{cases} \widehat{J(p_i)} \leq J(\widehat{a_j}) \\ \widehat{PLR(p_i)} \leq PLR(\widehat{a_j}) \\ \widehat{B(a_j)} \leq B(\widehat{p_i}) \end{cases} \tag{5.20}$$

Donde:

$i = 1, 2, 3, ..., n$
n = Total de trayectorias del nodo fuente al nodo destino
$j = 1, 2, 3, ..., m$
m = Total de aplicaciones

Este método fue implementado para obtener el conjunto de trayectorias factibles y a partir de este, obtener las trayectoría óptima. El Algoritmo 1 describe el proceso general de su implementación. Para el caso de la presente investigación, se minimiza el retardo del nodo fuente al nodo destino (también conocido como retardo punto a punto) cuando el porcentaje de pérdida de paquetes, las fluctuaciones y el ancho de banda disponible son menores o iguales al requerido por las aplicaciones y se selecciona la trayectoria que tenga el menor retardo.

En el modelo, como se muestra en la Figura 5.1, antes de enviar el trabajo a los nodos de cómputo, en ese instante de tiempo, QoSComm obtiene las condiciones de la red consultando al controlador SDN de cada uno de los centros de datos y su proceso matemático calcula la trayectoria óptima, si esto no es posible el algoritmo entrega la mejor conocida. Una vez que se obtienen dichas trayectorias, el proceso de comunicación de QoSComm crea, configura y asigna las reglas de flujo de datos que continene la trayectoria óptima basada en calidad de servicio, en cada uno de los switches de red. Este proceso lo efectúa antes de que el planificador de la aplicación HPC someta el trabajo a los nodos de cómputo en los centros de datos distribuidos. Ambas operaciones, prácticamente, no retrasan el proceso de envío de trabajos a los nodos de cómputo.

Algoritmo 1 Calcula las trayectorias óptimas del nodo fuente al destino utilizando las Ecuaciones 5.16, 5.17 y 5.18

ENTRADA:$[D(\widehat{app}), J(\widehat{app}), PLR(\widehat{app}), B(\widehat{app})]$ {requerimientos aplicación}
SALIDA:$ResultantPath$
Get $D(p_n), J(p_n), PLR(p_n), B(p_n)$ {obtiene el conjunto de trayectorias del nodo fuente al destino y sus parámetros de red}
$n \leftarrow totalpathsnumber$
$i \leftarrow 0$ {usado para el índice de la trayectoria}
$x \leftarrow 0$
while $i < n$ **do**
  if $(J(p_i) \leq J(\widehat{app}))$ AND $(PLR(p_i) \leq PLR(\widehat{app}))$ AND $(B(\widehat{app} \leq B(p_i))$ **then**
    $OpPaths[x] \leftarrow i$ {obtiene trayectorias factibles que cumplen requerimientos de la applicación}
    $x \leftarrow x + 1$
  end if
  $i \leftarrow i + 1$
end while
$ResultantPath \leftarrow min(D(OpPaths))$ {regresa la trayectoria óptima - menor retardo}

5.1.2. Proceso de comunicación de las trayectorias óptimas basadas en QoS

De acuerdo a la arquitectura SDN, QoSComm reside en el plano de aplicación y se comunica al controlador SDN a través de la Interfaz Norte. Se diseño un proceso para comunicarse con la Interfaz Norte a través de la REST API que proporciona el controlador SDN. La función principal del proceso de comunicación es configurar las trayectorias resultantes del algoritmo de decisión descrito en la Sección 5.1.1, a través de la REST API que proporciona el controlador SDN, como se representa en la Figura 5.3.

Como se mencionó anteriormente, la REST API permite la configuración dinámica de reglas de flujo de datos en la SDN y esta es una de las ventajas que ofrece este tipo de redes.

Una vez que se obtiene la trayectoria óptima, se crean las reglas de flujo de dicha trayectoria para la transferencia de datos. En su configuración, se asigna una mayor prioridad que la que contiene el redireccionamiento por controlador para que

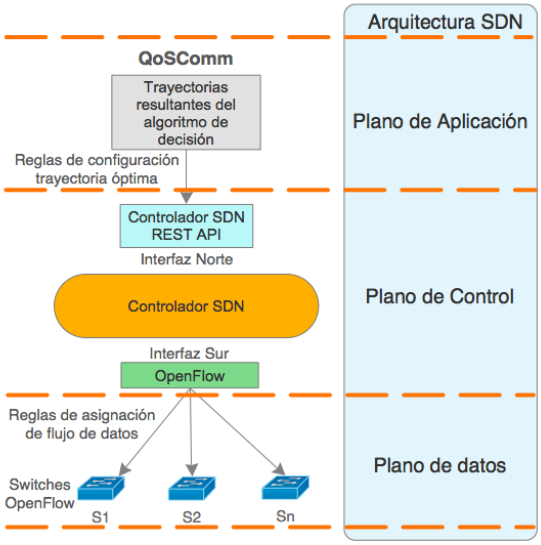

Figura 5.3: QoSComm proceso de comunicación y su referencia en la arquitectura SDN

el switch OF la verifique y en su caso, ejecute primero. En el caso de que dos o más reglas de flujo contengan los mismos valores en el campo de *match* descrito en el Listado 5.1, el switch OF ejecutará primeramente las reglas con mayor prioridad. Esta configuración de reglas se tiene que llevar a cabo en cada uno de los switches OF. Después de crear la regla, se da inicio a la transferencia de datos y cuando ésta finalice, QoSComm elimina la regla de flujo en cada uno de los switches OF. En el caso de que QoSComm no obtenga una trayectoria óptima de acuerdo a los requerimientos de QoS de la aplicación, el controlador SDN es responsable de asignar la trayectoria que utilizará para la transferencia de datos.

Para configurar las reglas de flujo en los switches OF, se crean reglas que cumplan con ciertos criterios como la dirección IP (Internet Protocol por sus siglas en inglés) de los nodos fuente y destino, el número de protocolo IP para el transporte de datos (6 para TCP y 17 para UDP) [39]. También se especifican las acciones que serán ejecutadas por los switches OF, donde se tiene que indicar el número de puerto del switch que será utilizada para la salida del flujo de datos, así como la prioridad que tendrá dicha regla, tal como se describe en el Listado 5.1.

Listado 5.1: OpenDayLight SDN controller XML file for data flow rule configuration parameters

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
<flow-name>flow1</flow-name>
<id>1</id>
<cookie_mask>255</cookie_mask>
<cookie>103</cookie>
<table_id>0</table_id>
<priority>200</priority>
<hard-timeout>0</hard-timeout>
<idle-timeout>0</idle-timeout>
<instructions>
  <instruction>
    <order>0</order>
    <apply-actions>
      <action>
        <order>0</order>
        <output-action>
          <output-node-connector>1</output-node-connector>
          <max-length>60</max-length>
        </output-action>
      </action>
    </apply-actions>
  </instruction>
</instructions>
<match>
  <ethernet-match>
    <ethernet-type><type>2048</type></ethernet-type>
  </ethernet-match>
  <ipv4-source>10.0.0.1/32</ipv4-source>
  <ipv4-destination>10.0.0.6/32</ipv4-destination>
  <ip-match>
    <ip-protocol>6</ip-protocol>
  </ip-match>
</match>
</flow>
```

El proceso de comunicación de QoSComm permite la comunicación con uno o más controladores SDN para realizar la configuración de reglas de flujo de datos en cada uno de los switches OF, de acuerdo a los requerimientos particulares de QoS de cierta aplicación que estén ejecutando trabajos entre centros de datos distribuidos.

La siguiente sección presenta el modelo de simulación así como la ejecución de experimentos considerando la transferencia de datos entre centros de datos distribuidos.

5.2. Modelo de simulación y experimentos

5.2.1. Modelo de simulación

El modelo de simulación utiliza una topología de red donde son requeridos dos o más centros de datos distribuidos conectados por una SDN para realizar la transferencia de flujos de datos entre sus nodos de cómputo. La topología de experimentación está basada en tres capas [40] y es una de las más utilizadas en centros de datos. Dicha topología consiste en la capa de acceso que conecta a los nodos de cómputo (hosts), una capa de agregación o región que conecta los switches de la capa de acceso y una capa raíz que se encuentra en el nivel superior del árbol.

La Figura 5.4 muestra la topología basada en tres capas utilizada para el modelo de simulación. En dicha topología se encuentran dos centros de datos definidos como Dominio A y Dominio B, mismos que se encuentran conectados por un enlace que puede ser el caso de un Proveedor de Servicios de Internet (ISP por sus siglas en inglés). Cada dominio tiene su controlador SDN, por lo que cada uno de ellos solo tiene acceso a los switches de sus respectivos dominios. Los valores de ancho de banda, retardo, fluctuaciones y porcentaje de pérdidas de paquetes son simulados. El listado del código que fue desarrollado en lenguaje Python utilizando el CLI de Mininet, se encuentra en el Anexo A.

QoSComm obtiene las trayectorias óptimas considerando los requerimientos de las aplicaciones y configura dichas trayectorias a través del controlador SDN de cada dominio. Como QoSComm no requiere que los controladores SDN se comuniquen entre ellos, no es necesario agregar un módulo adicional a cada controlador, por lo que se evita realizar modificación alguna a dichos controladores, ya que QoSComm se encarga de comunicarse con cada uno de ellos, además, QoSComm puede adaptarse para comunicarse con controladores de distintos fabricantes. **Esta es una de las principales contribuciones de QoSComm**.

En el Anexo B se encuentra un ejemplo del código desarrollado para configurar las trayectorias a través del controlador de la SDN.

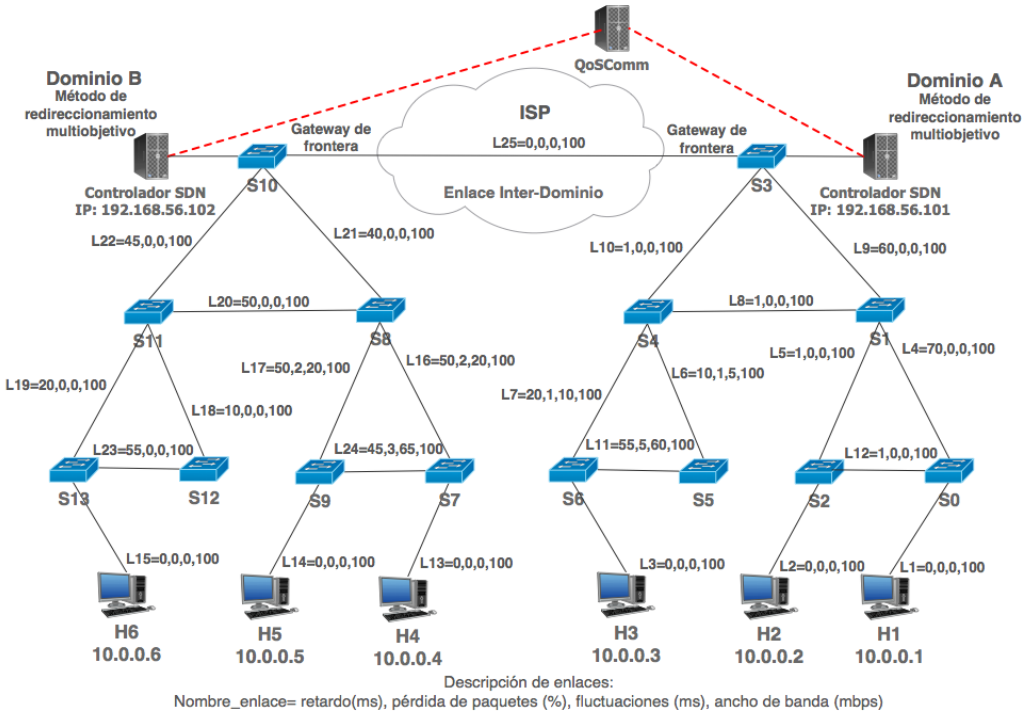

Figura 5.4: Topología de red para modelo de simulación

5.2.2. Diseño del experimento

El algoritmo multiobjetivo de generación de trayectorias óptimas fue implementado en lenguaje de programación Python, así como el script para configuración de trayectorias utilizando la REST API del controlador SDN. Como controlador SDN se utilizó OpenDayLight (ODL) [25] con el módulo RESTCONF habiltado. La topología de red para centros de datos distribuidos fue desarrollada utilizando el simulador SDN Mininet [41].

Para el entorno de simulación, se utilizó una estación de trabajo con procesador Intel Quad core, 16GB de memoria RAM, 512GB de disco duro de estado sólido, con sistema operativo MacOS High Sierra y software de virtualización VirtualBox 5.1. Dentro de VirtualBox se crearon dos máquinas virtuales con sistema operativo Linux Ubuntu 14, una de ellas para el software de simulación Mininet con un controlador ODL instalado y configurado para el Dominio A y la segunda máquina virtual para utilizarse como un controlador externo ODL para el Dominio B. También se configuró una red privada para la comunicación interna de las dos máquinas virtuales.

En la máquina virtual con Mininet, la topología de experimentación descrita en la Sección 5.2.1 fue programada en Lenguaje Python que hace referencia a objetos y componentes de la Interface de Línea de Comandos de Mininet (CLI por sus siglas en inglés). Los enlaces de los switches y nodos de cómputo (hosts) fueron creados con sus respectivos valores de ancho de banda, retardo, fluctuaciones y pérdida de paquetes descritos en la Figura 5.4. En resumen, se simularon dos dominios (A y B) cada uno con su respectivo controlador SDN y estos fueron configurados en el simulador Mininet.

A continuación se lista la metodología utulizada para obtener las trayectorias óptimas considerando los requerimientos de las aplicaciones y su configuración en Mininet:

1. De acuerdo a la topología de experimentación basada en tres capas, convertir cada enlace a un sistema de vectores con los valores de ancho de banda, retardo, fluctuaciones y pérdida de paquetes.

2. Definir las trayectorias para los hosts H1 y H6 (de acuerdo a la topología de experimentación)

3. Realizar la suma de los valores correspondientes de retardo, fluctuaciones y pérdida de paquetes para las trayectorias así como obtener el mínimo de ancho de banda disponible entre las trayectorias del paso 2.

4. Definir los valores para las restricciones de los objetivos con los requerimientos de la aplicación en (ancho de banda, retardo, fluctuaciones y pérdida de paquetes).

5. Buscar dentro de la lista de trayectorias, aquellas que cumplan con las restricciones.

6. De resultado de la búsqueda, obtener la trayectoria con el menor retardo y se clasifica como la trayectoria óptima.

7. Configurar las reglas de flujo de datos en cada uno de los switches OF a través del controlador SDN para que los hosts H1 y H8 utilicen la ruta óptima.

Tabla 5.1: Trayectorias de red y conexión de puertos de los switches OpenFlow

Link ID	SN:PN	DN:PN
1	H1:1	S0:1
2	H2:1	S2:1
3	H3:1	S6:1
4	S1:1	S0:2
5	S2:3	S1:2
6	S5:1	S4:4
7	S4:3	S6:2
8	S1:3	S4:1
9	S1:4	S3:1
10	S4:2	S3:2
11	S5:2	S6:3
12	S2:2	S0:3
13	H4:1	S7:1
14	H5:1	S9:1
15	H6:1	S13:1
16	S8:1	S7:3
17	S8:2	S9:3
18	S11:3	S12:1
19	S11:4	S13:2
20	S11:1	S8:3
21	S8:4	S10:1
22	S11:2	S10:2
23	S12:2	S13:3
24	S7:2	S9:2
25	S3:3	S10:3

Nomenclatura:
Link ID= Identificador del Enlace
SN= Nodo Fuente
PN= Número de Puerto
DN= Nodo Destino

Tabla 5.2: Trayectorias disponibles entre los Host 1 y Host 6 en el modelo de simulación

Path ID	Trayectoria de Red	Retardo (ms)	Saltos
1	S0,S1,S3,S10,S11,S13	195	6
2	S0,S1,S4,S3,S10,S11,S13	137	7
3	S0,S2,S1,S3,S10,S11,S13	127	7
4	S0,S2,S1,S4,S3,S10,S11,S13	69	8
5	S0,S1,S3,S10,S11,S12,S13	240	7
6	S0,S1,S4,S3,S10,S11,S12,S13	182	8
7	S0,S2,S1,S3, S10,S11,S12,S13	172	8
8	S0,S2,S1,S4,S3,S10,S11,S12,S13	114	9
9	S0,S1,S3,S10,S8,S11,S13	240	7
10	S0,S1,S4,S3,S10,S8,S11,S13	182	8
11	S0,S2,S1,S3,S10,S8,S11,S13	172	8
12	S0,S2,S1,S4,S3,S10, S8,S11,S13	114	9
13	S0,S1,S3,S10,S8,S11,S12,S13	185	8
14	S0,S1,S4,S3,S10,S8,S11,S12,S13	227	9
15	S0,S2,S1,S3,S10,S8,S11,S12,S13	217	9
16	S0,S2,S1,S4,S3,S10,S8,S11,S12,S13	159	10

Considerando valores de:
Ancho de banda disponible: 100 mbps
Pérdida de paquetes: 0 %
Fluctuaciones: 0 ms

Cuando se ejecuta la simulación, Mininet crea los enlaces de red conectado los puertos de cada uno de los switches OF. La Tabla 5.1 muestra los enlaces de red y la conexión de los puertos de cada switch.

El experimento consiste en la medición de la transferencia de datos entre los hosts H1 y H6 comparando el redirecionamiento por default que realiza en controlador contra el método de redireccionamiento QoSComm de la presente investigación. En la primera fase se utilizaron las aplicaciones de IPERF [42] y D-ITG [43] para realizar pruebas de desempeño de la red. En una segunda fase, se utiliza una aplicación de cómputo paralelo MPI para medir el tiempo de terminación de sus trabajos. La Tabla 5.2 muestra las trayectorias disponibles entre los hosts H1 y H6.

A continuación se presentan los pasos que se definieron para la ejecución de las pruebas:

1. Programar la topología de experimentación en simulador Mininet:

2. Desarrollar la topología de red con dos dominios (A y B) un conectar cada dominio con su respectivo controlador SDN.

3. Ejecutar pruebas de desempeño de la red del host H1 al host H6 utilizando IPERF y D-ITG (TCP y UDP) con el redireccionamiento de flujos de datos por Controlador:

   a) Obtener las métricas para la primer fase: Obtener el número de MBytes y Mbps transferidos en un periodo específico de tiempo.

4. Ejecutar pruebas de desempeño de aplicación HPC del host H1 al host H6 con el redireccionamiento de flujos de datos por Controlador:

   a) Obtener las métricas para la segunda fase: Obtener el tiempo de terminación de la aplicación MPI (en segundos).

5. Repetir los pasos 3 y 4 utilizando el redireccionamiento de flujos de datos basado en QoSComm.

6. Obtener los resultados.

5.3. Evaluación del desempeño

La evaluación se enfoca en el desempeño de las transferencias de flujos de datos entre dos hosts, cada uno hospedado en un centro de datos. En la primer fase, se utilizaron dos aplicaciones para probar el desempeño. IPERF y D-ITG fueron utilizadas para generar tráfico TCP y UDP entre los dos hosts, ya que su comportamiento sería similar a la de una aplicación HPC-BigData transfiriendo datos entre ellos. En el caso particular de D-ITG, fue utilizado para generar flujos de datos simultáneos entre los hosts, variando el támaño de los paquetes en cada transferencia, con la intención de simular el comportamiento de una aplicación transfiriendo datos en Internet. En la segunda fase, se configuró un cluster de cómputo basado en MPI en el simulador Mininet para probar el desempeño de este tipo de aplicaciones HPC. Se utilizó a una

aplicación MPI para transferir mensajes entre los nodos y medir el tiempo de terminación de sus trabajos. En ambas fases de prueba, se comparó en redireccionamiento de flujos de datos del Controlador SDN contra el método de QoSComm.

5.3.1. Primera fase: Desempeño de la red

IPERF

Se utilizó a IPERF para probar el desempeño de la red entre los hosts H1 y H6, aplicando el redireccionamiento de flujos de datos default del controlador y el método de QoSComm. IPERF fue configurado con los valores por default y se ejecutaron las pruebas considerando un intervalo de tiempo de 120 segundos para cada transferencia de datos. El servidor de IPERF se ejecutó en H6 y el cliente en H1. Los experimentos se ejecutaron 40 veces para flujos de datos por medio de UDP y luego se repitieron las pruebas para TCP.

Los resultados de IPERF se muestran en promedio de 10 grupos. Las figuras 5.5 y 5.6 y muestran los resultados utilizando TCP. Las figuras 5.7 y 5.8 muestran los resultados utilizando UDP. Como se puede observar, el redireccionamiento de flujos de datos por QoSComm obtuvo una mayor transferencia de datos en el mismo periodo de tiempo que el redireccionamiento default del controlador. Esto se debe a que el controlador utilizó la trayectoria 1 (Tabla 5.2: S0,S1,S3,S10,S11,S13), la cual tiene 6 saltos (hops por su término en inglés) para llegar a H6, contra 8 saltos de la trayectoria 4, sin embargo, cuenta con mayor retardo que la trayectoria 4 (195ms contra 69ms), que fue la calculada y configurada por QoSComm.

D-ITG

D-ITG fue utilizado para enviar flujos de datos simultáneos entre los hosts H1 y H6 aplicando los redireccionamientos default del controlador y por el método de QoSComm en el modelo de simulación. Se creó un script para transferir tres flujos de datos simultáneos cada uno con distinta tasa de paquetes (pps -C) y con 512 bytes como tamaño constante en el área de datos (payload por su término en inglés). En H6 se ejecutó el servidor de recepción de D-ITG. H1 fue utilizado para ejecutar el

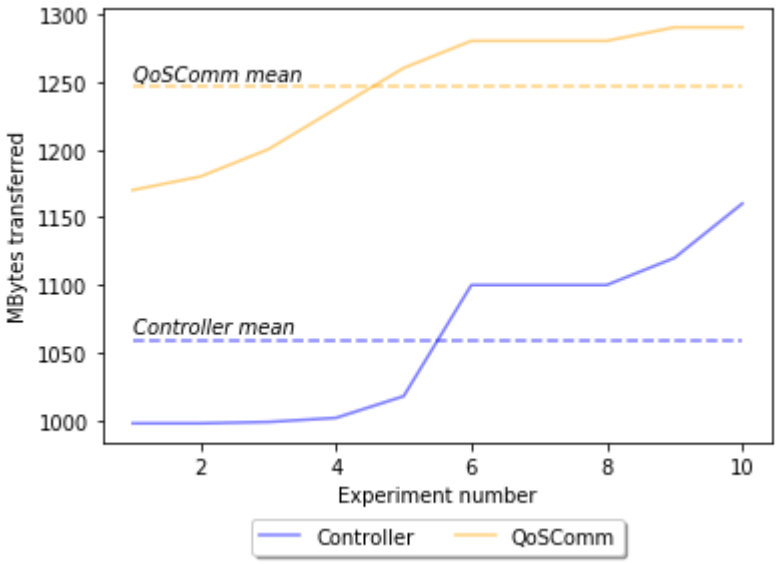

Figura 5.5: IPERF TCP comparativa de transferencia de datos (MBytes)

QoSComm PROMEDIO: 1246, STDEV: 47.18. Controlador default PROMEDIO: 1059.5, STDEV: 62.30. Intervalo de tiempo: 120 segundos.

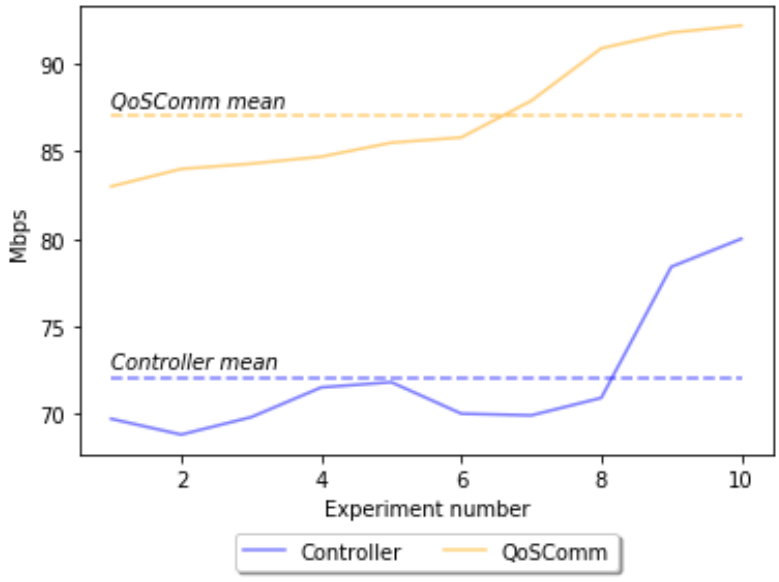

Figura 5.6: IPERF TCP comparativa de transferencia de datos (Mbps)

QoSComm PROMEDIO: 87.01, STDEV: 3.45. Controlador default PROMEDIO: 72.08, STDEV: 3.87. Intervalo de tiempo: 120 segundos.

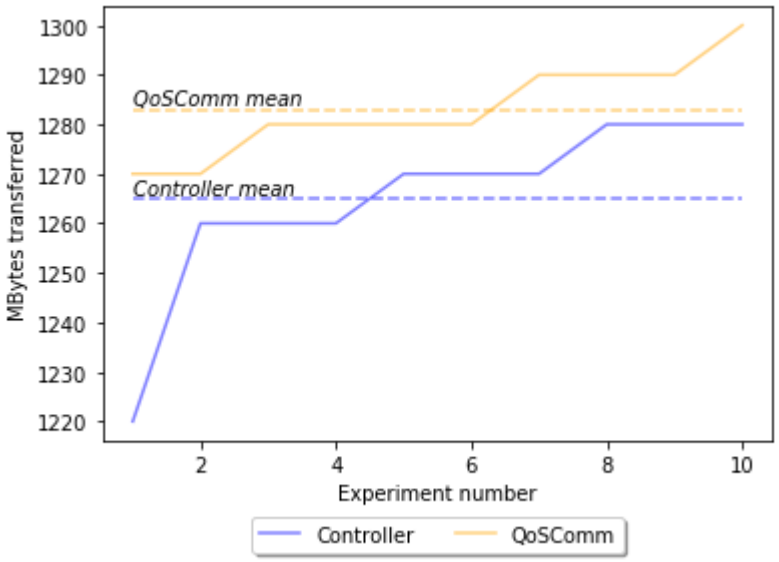

Figura 5.7: IPERF UDP comparativa de transferencia de datos (MBytes)

QoSComm PROMEDIO: 1283, STDEV: 9.48. Controlador default PROMEDIO: 265, STDEV: 17.79. Intervalo de tiempo: 120 segundos.

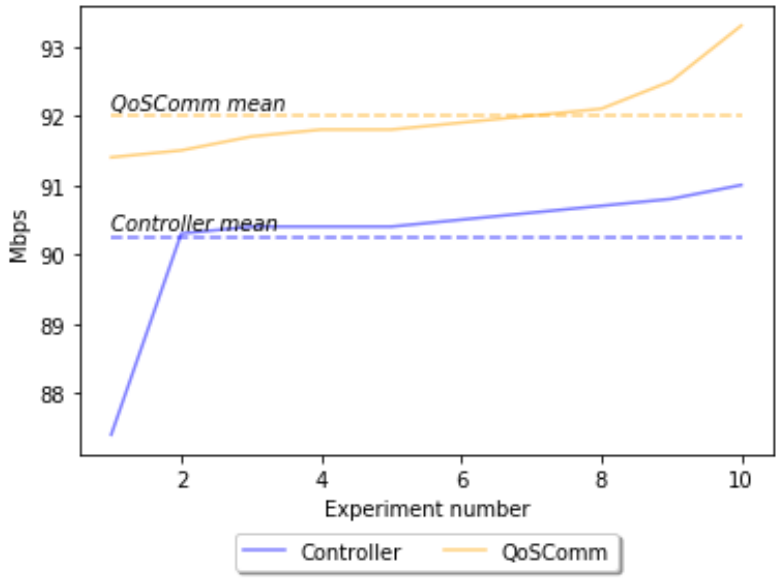

Figura 5.8: IPERF UDP comparativa de transferencia de datos (Mbps)

QoSComm PROMEDIO: 92, STDEV: 0.55. Controlador default PROMEDIO: 90.25, STDEV: 1.02. Intervalo de tiempo: 120 segundos.

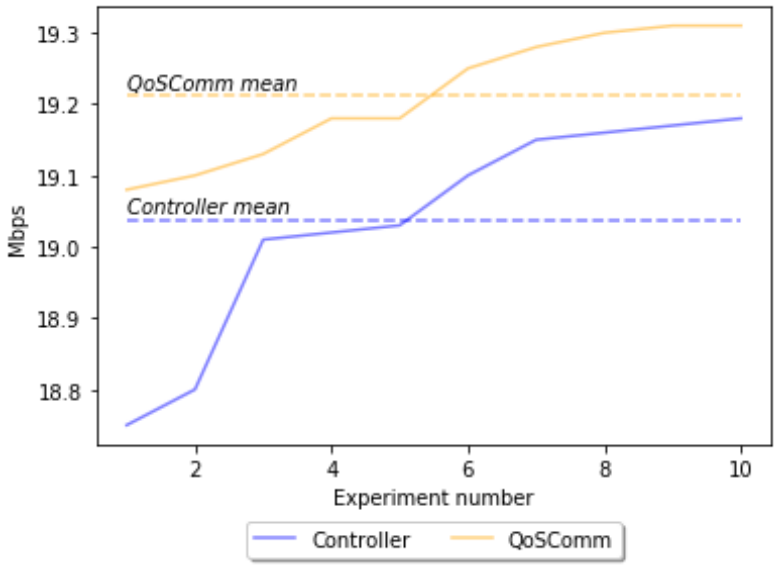

Figura 5.9: D-ITG comparativa de transferencia de datos (MBytes)

Tres flujos de datos UDP simultáneos. QoSComm PROMEDIO: 287.95 STDEV: 1.19. Controlador default PROMEDIO: 284.95, STDEV: 2.35. Intervalo de tiempo: 120 segundos.

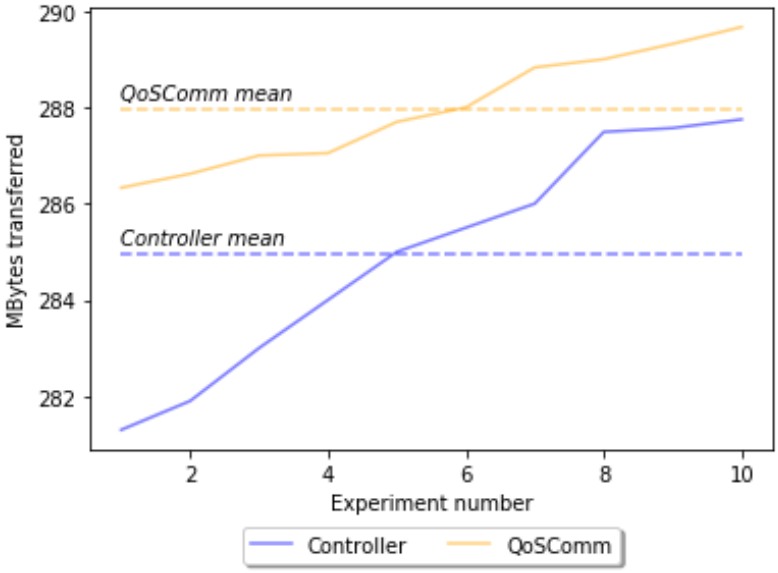

Figura 5.10: D-ITG comparativa de transferencia de datos (Mbps)

Tres flujos de datos UDP simultáneos. QoSComm PROMEDIO: 19.12 STDEV: 0.9. Controlador default PROMEDIO: 19.03, STDEV: 0.15. Intervalo de tiempo: 120 segundos.

servidor de log de D-ITG y también como cliente para transferir datos a H6. Durante las pruebas, se transfirieron flujos de datos del host H1 al H6 por 120 segundos. Las figuras 5.9 y 5.10 muestran las los resultados de las transferencias de D-ITG los cuales son la suma de Mbps y Mbytes de las tres transferencias simultáneas para cada experimento, respectivamente.

En las pruebas con D-ITG se observó que el método de redireccionamiento por QoSComm también obtuvo una mayor cantidad de datos transferidos, alrededor de 2-3 %, con respecto al default del controlador. QoSComm utilizó la trayectoria 4 (S0,S2,S1,S4,S3,S10,S11,S13: 69ms), la que tiene menor retardo y el controlador utilizó la trayectoria 1 (S0,S1,S3,S10,S11,S13: 195ms), con menos saltos pero con mayor retardo entre los hosts H1 y H6.

5.3.2. Segunda fase: Desempeño de la aplicación HPC

Pruebas con aplicación MPI

Como se mencionó en la sección anterior, se implementó un cluster MPI configurando el paquete MPICH [44] en la máquina virtual del simulador Mininet. Se utilizó una aplicación MPI para medir la transferencia de mensajes entre los hosts H1 y H6 (nodos de cómputo). La aplicación MPI proporciona comunicación punto a punto para un número dado de tareas MPI y utiliza TCP como protocolo de transporte. El objetivo de la prueba es medir el tiempo de terminación de la aplicación la cual ejecutará tareas entre los nodos de cómputo H1 y H6 utilizando los métodos de redireccionamiento de flujos por default del Controlador y QoSComm.

La aplicación MPI funciona de la siguiente manera: Se ejecuta la aplicación MPI y se crean dos tareas, una en cada nodo de cómputo. Cada tarea envía y recibe 100000 bytes entre los dos nodos, y lo hace 50 veces. Después, incrementa por 100000 bytes y repite el proceso de envío y recepción hasta alcanzar los 1000000 bytes. Dicha aplicación fue ejecutada 40 veces para cada uno de los métodos de redireccionamiento y se midió el tiempo de terminación.

Utilizando la misma topología de simulación, los resultados muestran que el tiempo de terminación de la aplicación MPI obtiene una mejora al utilizar la estrategia de redireccionamiento de flujos de datos QoSComm en vez de solo utilizar la default

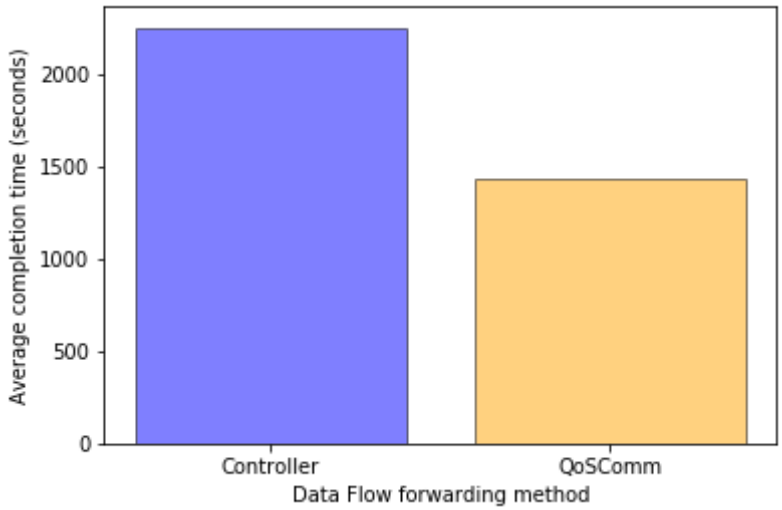

Figura 5.11: Aplicación MPI: Comparativa de redireccionamiento por Controlador vs QoSComm

Resultados del tiempo de terminación. QoSComm PROMEDIO: 1434.1, STDEV: 2.18. Controlador default PROMEDIO: 2251.6, STDEV: 3.09. Tiempo en segundos.

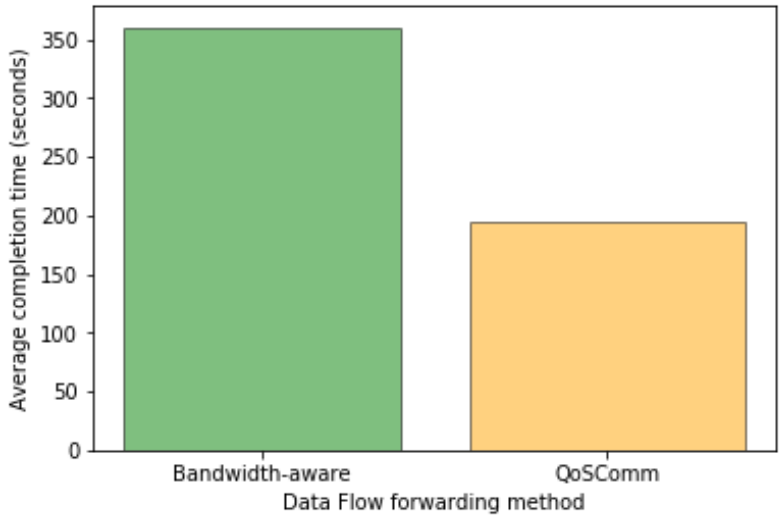

Figura 5.12: Aplicación MPI: Comparativa de redireccionamiento por Bandwidth-aware vs QoSComm

Resultados del tiempo de terminación. QoSComm PROMEDIO: 194.6, STDEV: 2.01. Bandwidth-aware PROMEDIO: 360.1, STDEV: 2.60. Tiempo en segundos.

del controlador, tal como se puede observar en la Figura 5.11. Como se mencionó anteriormente, QoSComm utiliza la trayectoria 4 (S0,S2,S1,S4,S3,S10,S11,S13) que tiene 69ms de retardo punto a punto y la aplicación terminó su trabajo en 1434 segundos (en promedio). El controlador utilizó la trayectoria 1 (S0,S1,S3,S10,S11,S13), la cual tiene 195ms de retardo punto a punto y le tomó 2251 segundos (en promedio) completar el trabajo de la aplicación. Se puede observar una diferencia de 817 segundos al utilizar el redireccionamiento de flujos de datos por QoSComm en de usar la default que proporciona el Controlador.

Para un escenario de redireccionamiento considerando la trayectoria con mayor ancho de banda disponible, que denominaremos bandwitdh-aware, se modificaron los valores de los parámetros de cada enlace en la topología de red. Los valores de retardo se eliminaron en todos los enlaces y solo se configuraron con 10ms de retardo los enlaces 4, 9, 19 y 22 referidos en la Tabla 5.1. También se cambió la la tasa de transmisión de 100 a 80 mbps en los enlaces 5 y 12 de la Tabla 5.1.

Cuando se ejecutó la simulación, el Controlador tomó la trayectoria 1 (Tabla 5.2) para la transferencia de datos entre los hosts H1 y H6. Para esta simulación, la trayectoria 1 Tabla 5.2 tiene 100mbps de tasa de transmisión y 40ms de retardo punto a punto. QoSComm calculó las trayectorias y seleccionó y configuró la que obtuvo menor retardo entre los nodos de cómputo. En este caso, la trayectoria 16 (Tabla 5.2) tiene 80mbps para transferencia de datos y no tiene retardos punto a punto. La Figura 5.12 muestra un promedio del tiempo de terminación comparando el redireccionamiento por bandwidth-aware contra QoSComm. Los resultados muestran que para esta aplicación MPI, la propuesta de redireccionamiento de flujos de datos por el método de QoSComm mejora considerablemente el tiempo de terminación de dicha aplicación, logrando 35 % menos que al utilizar el redireccionamiento por controlador.

En base a estos resultados, se puede clasificar a esta aplicación MPI como sensible al tiempo por lo que, de acuerdo a los resultados de las pruebas, el retardo punto a punto se convierte en un factor crítico a considerar en escenarios de centros de datos distribuidos, y no es suficiente contar con mayor ancho de banda, tal como se observó en dichos resultados.

Considerando lo anterior, QoSComm superó a los trabajos descritos en la Tabla

4.1 al considerar los requerimientos de QoS de las aplicaciones HPC-BigData en base a cuatro parámetros de red para la selección de trayectorias óptimas, y tomando en cuenta una arquitectura de centros de datos distribuidos, donde el parámetro de retardo se convierte en un factor muy importante a considerar y no es suficiente con solo controlar el ancho de banda. Además, QoSComm puede comunicarse con los diferentes controladores de la SDN de cada centro de datos (o dominio) descubriendo dinámicamente la topología de red en operación de cada uno de estos centros, explotando los mecanismos que proveen las redes definidas por software para beneficio de este tipo de aplicaciones, por lo que se cumple con el objetivo 1.3 de la presente tesis.

Capítulo 6

Conclusiones

En la presente investigación se analizaron diversos trabajos en donde las aplicaciones HPC y BigData hacen uso de la red definida por software para la toma de decisión en sus procesos de planificación de tareas y trabajos. Si bien, en la mayoría de los trabajos se logra una mejora en el desempeño de la aplicación, considerando algunos parámetros de la red, existieron casos donde se afectó su tiempo de terminación o respuesta. Como parte de la revisión de dichas propuestas, en ninguna de las propuestas consultadas se consideraron políticas de calidad de servicio que contemplen requerimientos de las aplicaciones y condiciones de la red. En este aspecto, se identificaron una serie de oportunidades y retos relacionados a proporcionar calidad de servicio en la SDN y cómo pueden contribuir a mejorar el desempeño de las aplicaciones.

Los trabajos desarrollados hasta el momento que han incursionado en el tema de QoS en la SDN proveen una ventana de oportunidades para mejorar el desempeño de aplicaciones como HPC y BigData. Estos trabajos han resuelto de manera parcial algunas necesidades que pudieran funcionar para este tipo de aplicaciones, sin embargo, como resultado de esta investigación, se identificaron áreas de oportunidad para el desarrollo de nuevos temas de investigación relacionados a una calidad de servicio dinámica y especializada para aplicaciones de cómputo de altas prestaciones, buscando minimizar sus tiempos de terminación y optimizar los recursos de cómputo de los centros de datos. En el Captítulo 4, se cumple con el objetivo específico 1 de la tesis.

Considerando lo anterior, en la presente investigación se desarrolló un modelo de QoS para la distribución de flujos de datos denominado QoSComm, que proporciona calidad de servicio desde el punto de vista de red, enfocado a aplicaciones HPC-BigData, especificamente en el proceso de envío de trabajos a clusters de cómputo hospedados en centros de datos distribuidos, logrando mejorar los tiempos de terminación de sus trabajos, tal como lo muestran los resultados de los experimentos realizados en el capítulo anterior, cumpliendo así con los objetivos específicos 2 y 3 de la presente tesis.

Las aplicaciones de cómputo de alto desempeño que requieren de transferir datos entre nodos de cómputo hospedados en centros de datos distribuidos conectados por una SDN, pueden beneficiarse en su tiempo de terminación al considerar trayectorias que cumplan sus requerimientos de calidad de servicio en el proceso de asignación de reglas de flujo de datos en la red, y no solo utilizar el redireccionamiento tradicional del controlador o solo considerar trayectorias con mayor ancho de banda.

Como trabajo futuro, se propone una mejora al diseño de QoSComm para que incorpore un servicio web, del tipo Restful, que permita a las aplicaciones comunicarle requerimientos adicionales para la ejecución de sus trabajos, como seguridad y auto-provisionamiento de recursos como almacenamiento, entre otros, aprovechando las ventajas que ofrecen los componentes de los centros de datos definidos por sofware, descritos en el Capítulo 2. Otro de los aspectos a considerar como trabajo futuro, es la combinación de la selección de trayectorias basadas en QoS y un esquema de redireccionamiento seguro para aplicaciones HPC-BigData que contengan datos sensibles.

Referencias

[1] P. Chupala, K. Ichikawa, H. Lida, N. Kessaraphong, P. Uthayopas, S. Date, H. Yamanaka, and E. Kawai. Application-Oriented Bandwidth and Latency Aware Routing with OpenFlow Network. In *Proceedings of the IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014.

[2] Peng Qin, Bin Dai, Beniong Huang, and Guan Xu. Bandwidth-Aware Scheduling with SDN in Hadoop: A New Trend for Big Data. *IEEE Systems Journal*, pages 2337–2344, 2017.

[3] M. Veiga, C. Rose, K. Katrinis, and H. Franke. Pythia: Faster Big Data in Motion through Predictive Software-Defined Network Optimization at Runtime. In *IEEE 28th International Parallel & Distributed Processing Symposium*, 2014.

[4] K. Govindarajan, K. Meng, H. Ong, and W. Tat. Realizing the Quality of Service (QoS) in Software-Defined Networking (SDN) Based Cloud Infrastructure. In *Proceedings of the 2nd International Conference on Information and Communication Technology (ICoICT)*, 2015.

[5] H. Egilmez, S. Dane, K. Bagci, and A. Tekalp. OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks. In *Signal & Information Processing Association Annual Summit and Conference*, 2012.

[6] M.A. Karaman, B. Gorkemli, S. Tatlicioglu, M. Komurcuoglu, and O. Karakaya. Quality of Service Control and Resource Priorization with Software Defined Networking. In *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[7] H. Owens and A. Durresi. Video over Software-Defined Networking (VSDN). In *16th International Conference on Network-Based Information Systems*, 2013.

[8] S. Tomovic, N. Prasad, and I. Radusinovic. SDN control framework for QoS provisioning. In *IEEE 22nd Telecommunications Forum*, 2014.

[9] M. Marchese. *QoS over heterogeneous networks*. John Wiley & Sons, 2007.

[10] H. Farhady, H. Lee, and A Nakao. Software-Defined Networking: A survey. *Elsevier Computer Networks*, 81:79–95, 2015.

[11] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004.

[12] K. Brown, J. Domke, and S Matsuoka. Hardware-Centric Analysis of Network Performance for MPI Applications. In *IEEE 21st International Conference on Parallel and Distributed Systems*, 2015.

[13] J. Sloan. *High Performance Linux Clusters*. O'Really, 2004.

[14] Apache Hadoop. `https://hadoop.apache.org`. Último acceso: 25-mayo-2019.

[15] Lizhe W., Jie Tao, H. Marten, A. Streit, S.U. Khan, J. Kolodziej, and D Chen. MapReduce Across Distributed Clusters for Data-intensive Applications. In *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, 2012.

[16] M. Carlson, A. Yoder, L. Schoeb, D. Deel, C. Pratt, C. Lionetti, and D. Voigt. *Software Defined Storage*. Storage Networking Industry Association, 2015.

[17] M. Portnoy. *Virtualization Essentials*. Wiley and Sons, 2016.

[18] Medrano-Jaimes F., Lozano-Rizk J.E., Castaneda-Avila S., and Rivera-Rodriguez R. Use of Containers for High-Performance Computing. *Communications in Computer and Information Science (Springer)*, 948:24–32, 2019.

[19] C. Rewer, A. Tchernykh, J. Cortes-Mendoza, R. Rivera-Rodriguez, J. Lozano-Rizk, A. Avetisyan, Z. Du, G. Radchenko, and E. Concepcion-Morales. Energy Consumption and Quality of Service Optimization in Containerized Cloud Computing. In *Ivannikov Ispras Open Conference (ISPRAS)*, 2019.

[20] T. Adufu, J. Choi, and Y. Kim. Is container-based technology a winner for high performance scientific applications? In *7th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2015.

[21] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. SDSecurity: A Software Defined Security Experimental Framework. In *IEEE International Conference con Communication Workshop*, 2015.

[22] W. Stallings. Software-Defined Networks and OpenFlow. *The Internet Protocol Journal*, 16, 2013.

[23] OpenFlow. Open Networking Foundation. `https://www.opennetworking.org`. Último acceso: 25-abril-2019.

[24] A. Garcia, C. Rodriguez, C. Calderon, and F. Casmartino. Controladores SDN, elementos para su selección y evaluación. *Revista Telem@tica*, 13:10–20, 2014.

[25] OpenDayLight Project. `https://www.opendaylight.org`. Último acceso: 25-abril-2019.

[26] S. Deshmukh, J.V. Aghav, and R Chakravarthy. Job Classification for MapReduce Scheduler in Heterogeneous Environment. In *Cloud and Ubiquitous Computing and Emerging Technologies (CUBE)*, 2013.

[27] Y. Watashiba, K. Kido, S. Date, H. Abe, K. Ichikawa, H. Yamanaka, E. Kawai, and H. Takemura. Prototyping and evaluation of a network-aware Job Management System on a cluster system. In *19th IEEE International Conference on Networks (ICON)*, 2013.

[28] P. Makpaisit, K. Ichikawa, and P. Uthayopas. MPI Reduce Algorithm for OpenFlow-Enabled Network. In *Proceedings of the 15th International Symposium on Communications and Information Technologies (ISCIT)*, 2015.

[29] J. Huang, L. Xu, M. Zeng, C. Xing, Q. Duan, and Y. Yan. Hybrid Scheduling for Quality of Service Guarantee in Software Defined Networks to support Multimedia Cloud Services. In *IEEE International Conference on Services Computing*, 2015.

[30] H. Alkaff, I. Gupta, and L. Leslie. Cross-Layer Scheduling in Cloud Systems. In *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, 2015.

[31] Apache Storm. `https://storm.apache.org/`. Último acceso: 25-mayo-2019.

[32] H. Owens and A. Durresi. Explicit Routing in Software-Defined Networking (ERSDN): Addressing Controller Scalability. In *IEEE International Conference on Network-Based Information Systems*, 2015.

[33] O. Younis and S. Fahmy. Constraint-based Routing in the Internet: Basic Principles and Recent Research. *IEEE Communications Surveys*, 5:2–13, 2003.

[34] M. Tajiki, B. Akbari, M. Shojafar, S. Ghasemi, M. Barazandeh, N. Mokari, L. Chiaraviglio, and M. Zink. CECT: computationally efficient congestion-avoidance and traffic engineering in software-defined cloud data centers. *Cluster Computing*, pages 1881–1897, 2018.

[35] A. Shah, W. Wu, Q. Lu, L. Zhangb, S. Sasidharan, P. DeMar, C. Guokc, J. Macauleyc, E. Pouyoulc, J. Kimd, and S. Noh. AmoebaNet: An SDN-enabled network service for big data science. *Journal of Network and Computer Applications*, 119:70–82, 2018.

[36] E. Cosio-Velazquez. *Modelado de una arquitectura de red definida por software (SDN) para el aprovisionamiento de recursos utilizando Cross-Layer-Design (CLD) Tesis.* CICESE, 2017.

[37] M. Emmerich and A. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17:585–609, 2018.

[38] G. Pantuza-Junior. A multi-objective approach to the scheduling problem with workers allocation. *Gest. Prod.*, 23:132–145, 2016.

[39] Assigned Internet Protocol Numbers List. `https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml`. Último acceso: 25-mayo-2019.

[40] R. Hwang, H. Tseng, and Y. Tang. Design of SDN-enabled Cloud Data Center. In *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 2015.

[41] Mininet SDN Simulator. `http://www.mininet.org`. Último acceso: 25-abril-2019.

[42] Network performance tool. `https://iperf.fr`. Último acceso: 25-abril-2019.

[43] A. Botta, A. Dainotti, and A. Pescape. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks (Elsevier)*, 56:3531–3547, 2012.

[44] MPICH Sitio web oficial. `http://www.mpich.org`. Último acceso: 25-abril-2019.

Glosario

API	Application Programming Interface.
CPU	Central Processing Unit.
D-ITG	Distributed Internet Traffic Generator.
HPC	High Performance Computing.
HPCA	High Performance Computing Applications.
MPI	Message Passing Interface.
OFS	OpenFlow Switch.
QoS	Quality of Service.
RAM	Random Access Memory.
SDN	Software Defined Network.

Anexos

Anexo A

Código de topología de simulación en Mininet

Ejemplo de topología de simulación -centros de datos distribuidos

```
#!/usr/bin/python

"""
Programacion de topologia para datacenter basada en modelo
    de 3 capas
Topologia No. AB-1
"""
from mininet.net import Mininet
from mininet.link import TCLink
from mininet.node import OVSSwitch, Controller,
    RemoteController
from mininet.topolib import TreeTopo
from mininet.topo import Topo
from mininet.log import setLogLevel
from mininet.cli import CLI

#definir controladores     -variables globales
```

```
c0= RemoteController('c0', '127.0.0.1', 6633)     #ODL
c1= RemoteController('c1', '192.168.56.102', 6633) #ODL

# definir los switches que atendara cada controlador:  c0
   del s0 al s6,  c1 del s7 al s13
cmap={'s0': c0,'s1': c0, 's2': c0, 's3': c0, 's4': c0, 's5':
    c0, 's6': c0, 's7' : c1, 's8': c1, 's9':c1, 's10':c1, '
   s11':c1,'s12':c1, 's13':c1}

class DataCenterA(Topo):

def __init__(self, **opts):
"""Create custom topo."""

# Initialize topology
# It uses the constructor for the Topo cloass
super(DataCenterA, self).__init__(**opts)

# Agregar hosts para la simulacion
h1 = self.addHost('h1')
h2 = self.addHost('h2')
h3 = self.addHost('h3')
h4 = self.addHost('h4')
h5 = self.addHost('h5')
h6 = self.addHost('h6')

# Agregar switches Dominio A
s0 = self.addSwitch('s0', dpid="0000000000000001")
s1 = self.addSwitch('s1', dpid="0000000000000002")
s2 = self.addSwitch('s2', dpid="0000000000000003")
s3 = self.addSwitch('s3', dpid="0000000000000004")
s4 = self.addSwitch('s4', dpid="0000000000000005")
```

```
s5 = self.addSwitch('s5', dpid="0000000000000006")
s6 = self.addSwitch('s6', dpid="0000000000000007")

# Agregar switches Dominio B
s7 = self.addSwitch('s7', dpid="0000000000000008")
s8 = self.addSwitch('s8', dpid="0000000000000009")
s9 = self.addSwitch('s9', dpid="0000000000000010")
s10 = self.addSwitch('s10', dpid="0000000000000011")
s11 = self.addSwitch('s11', dpid="0000000000000012")
s12 = self.addSwitch('s12', dpid="0000000000000013")
s13 = self.addSwitch('s13', dpid="0000000000000014")

# Hosts del Dominio A, conectados a Switch#
self.addLink(h1, s0, bw=100, max_queue_size=1000, use_htb=
    True)
self.addLink(h2, s2, bw=100, max_queue_size=1000, use_htb=
    True)
self.addLink(h3, s6, bw=100, max_queue_size=1000, use_htb=
    True)

# Hosts del Dominio B, conectados a Switch#
self.addLink(h4, s7, bw=100, max_queue_size=1000, use_htb=
    True)
self.addLink(h5, s9, bw=100, max_queue_size=1000, use_htb=
    True)
self.addLink(h6, s13, bw=100, max_queue_size=1000, use_htb=
    True)

 # Agregar enlaces para switches Dominio A
# Nomenclatura retraso, perdida de paquetes, fluctuaciones,
    ancho de banda
```

```
self.addLink(s0, s1, bw=100, delay='70ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s0, s2, bw=100, delay='1ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s1, s2, bw=100, delay='1ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s1, s4, bw=100, delay='1ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s1, s3, bw=100, delay='60ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s4, s3, bw=100, delay='1ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s4, s6, bw=100, delay='20ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s4, s5, bw=100, delay='1ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s5, s6, bw=100, delay='55ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)

 self.addLink(s7, s9, bw=100, delay='45ms', loss=0, jitter='
     0ms', max_queue_size=1000, use_htb=True)
self.addLink(s7, s8, bw=100, delay='50ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s9, s8, bw=100, delay='50ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)
self.addLink(s8, s11, bw=100, delay='50ms', loss=0, jitter='
    0ms', max_queue_size=1000, use_htb=True)
self.addLink(s8, s10, bw=100, delay='40ms', loss=0, jitter='
    0ms', max_queue_size=1000, use_htb=True)
self.addLink(s11, s10, bw=100, delay='45ms', loss=0, jitter=
    '0ms', max_queue_size=1000, use_htb=True)
```

```
self.addLink(s11, s12, bw=100, delay='10ms', loss=0, jitter=
    '0ms', max_queue_size=1000, use_htb=True)
self.addLink(s11, s13, bw=100, delay='20ms', loss=0, jitter=
    '0ms', max_queue_size=1000, use_htb=True)
self.addLink(s13, s12, bw=100, delay='55ms', loss=0, jitter=
    '0ms', max_queue_size=1000, use_htb=True)

self.addLink(s3, s10, bw=100, delay='0ms', loss=0, jitter='0
    ms', max_queue_size=1000, use_htb=True)

class MultiSwitch( OVSSwitch ):
"Switch Custom que conecta a switches controlados por
    distintos controladores "
def start(self, controllers):
return OVSSwitch.start(self, [cmap[self.name]])
def run():
# Ejecutar simulacion de red virtual, controller=None es
    para que no utilizar un controlador local y solo los dos
    controladores remotos
net = Mininet(topo=DataCenterA(), link=TCLink, switch=
    MultiSwitch, controller=None)
net.addController(c0)
net.addController(c1)
net.start()
CLI(net)
net.stop()

# if the script is run directly (sudo custom/optical.py):
if __name__ == '__main__':
setLogLevel('info')
run()
```

Anexo B

Códigos para configuración de reglas de flujo

Ejemplo de scripts en Python para configuración de reglas de flujo en switches OF.

```
#!/usr/bin/python
import os

#para eliminar un flujo condigurado desde QoSComm en un
    switch
os.system('curl -X DELETE -H "Content-Type: application/json
    " -H "Accept: application/json" --user admin:admin http
    ://192.168.56.102:8181/restconf/config/opendaylight-
    inventory:nodes/node/openflow:17/table/0/flow/1')

#para configurar un flujo a traves de ODL en un switch
# configura Ruta 1 protocolo TCP de acuerdo al calculo de
    ruta de la topologia topo-AB-1M
# controlador ODL Dominio A (192.168.56.101)
```

```
os.system('curl -X PUT -d @mP2_H6-1 -H "Content-Type:
    application/xml" -H "Accept: application/xml" --user admin
    :admin http://192.168.56.101:8181/restconf/config/
    opendaylight-inventory:nodes/node/openflow:1/table/0/flow
    /1')

os.system('curl -X PUT -d @mP4_H6-1 -H "Content-Type:
    application/xml" -H "Accept: application/xml" --user admin
    :admin http://192.168.56.101:8181/restconf/config/
    opendaylight-inventory:nodes/node/openflow:2/table/0/flow
    /1')

os.system('curl -X PUT -d @mP3_H6-1 -H "Content-Type:
    application/xml" -H "Accept: application/xml" --user admin
    :admin http://192.168.56.101:8181/restconf/config/
    opendaylight-inventory:nodes/node/openflow:4/table/0/flow
    /1')

#controlador ODL Dominio B (192.168.56.102)
os.system('curl -X PUT -d @mP2_H6-1 -H "Content-Type:
    application/xml" -H "Accept: application/xml" --user admin
    :admin http://192.168.56.102:8181/restconf/config/
    opendaylight-inventory:nodes/node/openflow:17/table/0/flow
    /1')

os.system('curl -X PUT -d @mP4_H6-1 -H "Content-Type:
    application/xml" -H "Accept: application/xml" --user admin
    :admin http://192.168.56.102:8181/restconf/config/
    opendaylight-inventory:nodes/node/openflow:18/table/0/flow
    /1')
```

```
os.system('curl -X PUT -d @mP1_H6-1 -H "Content-Type:
    application/xml" -H "Accept: application/xml" --user admin
    :admin http://192.168.56.102:8181/restconf/config/
    opendaylight-inventory:nodes/node/openflow:20/table/0/flow
    /1')

#Para consultar flujo configurados en la RESTCONF de ODL
os.system('curl -X GET -H "Content-Type: application/json" -
    H "Accept: application/json" --user admin:admin http
    ://192.168.56.102:8181/restconf/config/opendaylight-
    inventory:nodes/node/openflow:20/table/0/flow/1')
```

Ejemplo de archivos XML para configuración de reglas de flujo a través del controlador ODL. Archivo mP2H6-1

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
<strict>false</strict>
<flow-name>flow1</flow-name>
<id>1</id>
<cookie_mask>255</cookie_mask>
<cookie>103</cookie>
<table_id>0</table_id>
<priority>1</priority>
<hard-timeout>1800</hard-timeout>
<idle-timeout>1800</idle-timeout>
<installHw>true</installHw>
<instructions>
 <instruction>
  <order>0</order>
  <apply-actions>
  <action>
   <order>0</order>
```

```
    <output-action>
     <output-node-connector>2</output-node-connector>
     <max-length>60</max-length>
    </output-action>
   </action>
   </apply-actions>
  </instruction>
</instructions>
<match>
 <ip-match><ip-protocol>4</ip-protocol></ip-match>
 <ethernet-match>
  <ethernet-type>
   <type>2048</type>
  </ethernet-type>
 </ethernet-match>
 <ipv4-source>10.0.0.1/8</ipv4-source>
 <ipv4-destination>10.0.0.6/8</ipv4-destination>
</match>
</flow>
```

Contribuciones Académicas

A) Software Defined Data Center for High Performance Computing Applications

Se redactó una versión corta del artículo y fue sometida al 10th International Supercomputing Conference in Mexico (ISUM 2019). Dicha versión fue aceptada y presentada en el Congreso el 28 de marzo en la sesión de Ponencias. La versión extendida se sometió el 20 de abril. El artículo fue aceptado el 15 de mayo del 2019, carta del Editor y será publicado en la revista internacional Communications in Computing and Information Science (CCIS) de Springer a principios del 2020.

B) QoSComm: A Data Flow Allocation Strategy Among SDN-Based Data Centers For IoT Big Data Analytics

Artículo sometido a la revista: International Journal of Distributed Sensor Networks (IJDSN) de SAGE Publishing el 24 de mayo del 2019. Del 29 de mayo y hasta el 10 de junio estuvo en revisión de editor y del 11 de junio a la fecha su estatus es "en asignación de revisores".

C) QoS-Aware Data Forwarding Scheme in Software-Defined Networking Based on Genetic-MOCELL optimization

Artículo en elaboración. Colaboración con investigadores del CICESE y UABC. Se planea someter a la revista Computer Networks de Elsevier a finales del mes de agosto del 2019.

D) Participación en comité de tesis

- Participación como Co-Asesor del trabajo "Análisis de la Factibilidad para la Transformación de un Centro de Datos Tradicional a Uno Definido por Software", por Verónica Rico Rodríguez, estudiante de la Maestría en Gestión de Tecnologías de la Información y Comunicaciones, FCAyS-UABC.

- Participación como miembro del comité de tesis del trabajo "Enrutamiento de flujos para trayectorias seguras utilizando redes definidas por software (SDN)" por Jose Enrique Gonzalez Trejo, estudiante de la Maestría en Ciencias en Electrónica y Telecomunicaciones del CICESE.

- Participación como miembro del comité de tesis del trabajo "Modelado de una arquitectura de red definida por software (SDN) para el aprovisionamiento de recursos utilizando Cross-Layer-Design (CLD)" por Ernesto Cosío Velázquez, estudiante de la Maestría en Ciencias en Electrónica y Telecomunicaciones del CICESE. Obtuvo el grado en Febrero del 2017.

E) Participación en congresos y reuniones

- Participación como ponente del tema "Software Defined Data Center for High Performance Computing Applications" en Congreso Internacional ISUM2019, Monterrey, Nuevo León, Marzo del 2019.

- Participación como ponente del tema "Simulación de Topologías de Red para Centros de Datos Distribuidos Basados en Redes Definidas por Software" como parte de la presentación de avances de investigación en la Reunión Anual de la Red Mexicana de Supercómputo, Red Temática del CONACYT, en las instalaciones de la Universidad de Guadalajara, Guadalajara, Jalisco, Octubre del 2018.

- Participación como ponente del tema "Redes Definidas por Software para el Cómputo de Altas Prestaciones" como parte de la presentación de avances de investigación en la Reunión Anual de la Red Mexicana de Supercómputo, Red Temática del CONACYT, en las instalaciones del CICESE Ensenada, B.C., Noviembre del 2017.